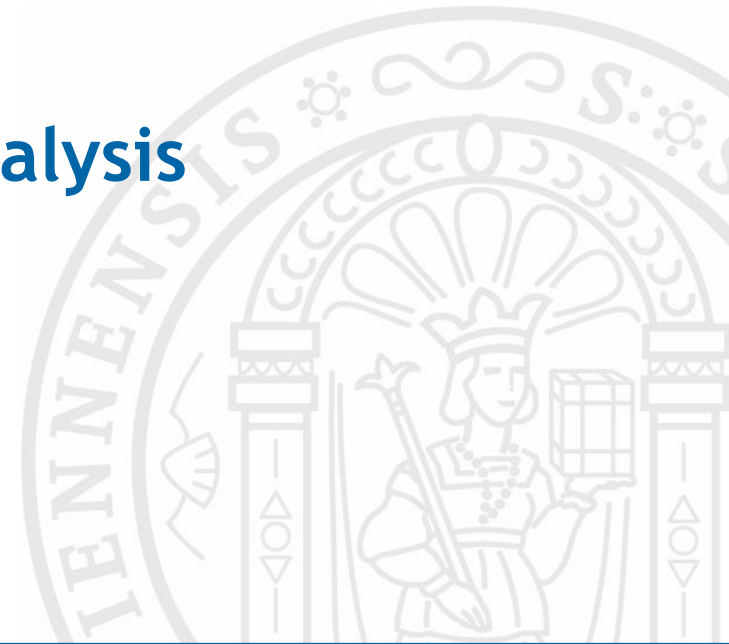


NEXT GENERATION ENTERPRISE MODELLING IN THE AGE OF INTERNET OF THINGS - NEMO 2016 -

Smart City Exercises

Exercise II: Smart City Analysis

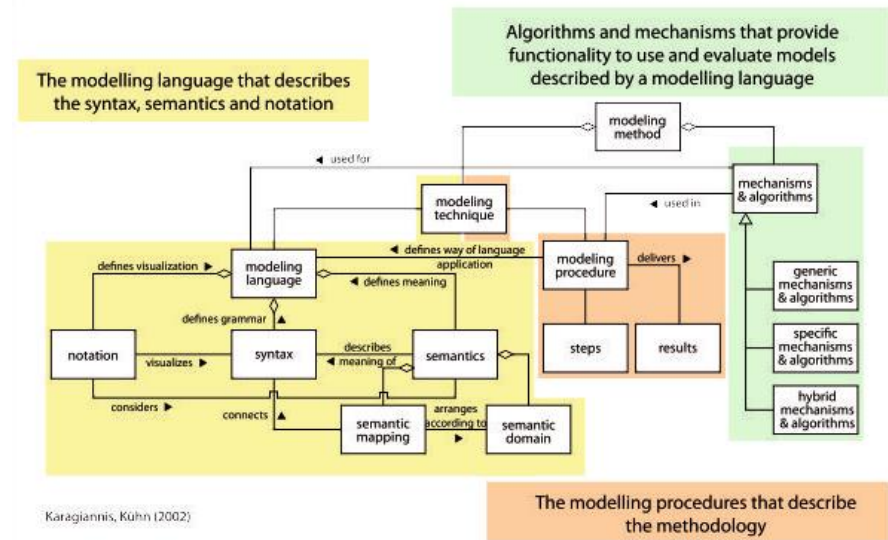


INTRODUCTION



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**

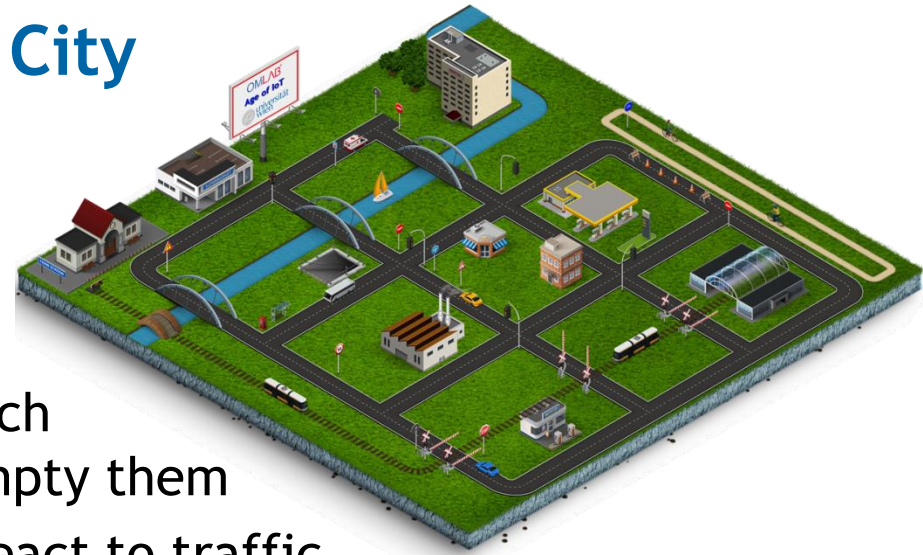


Karagiannis, Kuhn (2002)

Mechanisms and Algorithms - Analysis

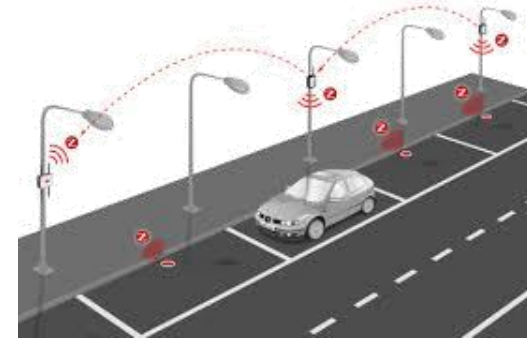
- Analysis assists in utilizing the complex relationships within a model or between multiple models
- Manual model analysis can be costly
- An analysis technique is chosen for a certain purpose
 - Analysis supports the decision making process
 - Analysis assists in understanding, visualizing and communicating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

- 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, ...)



www.psfk.com



Emergency Scenario: Block Public Traffic (animated)

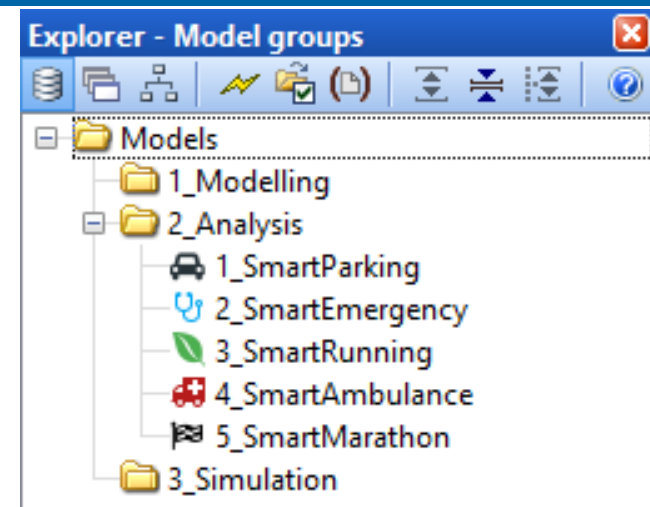


Emergency Scenario: Medical Response (animated)



11

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

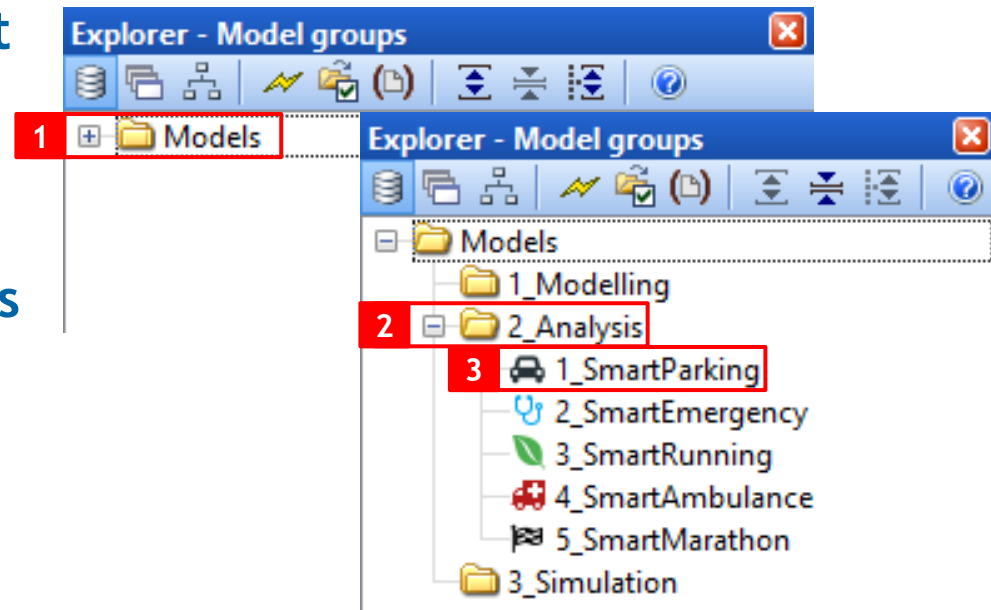
4. Pre-defined analysis queries

- 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



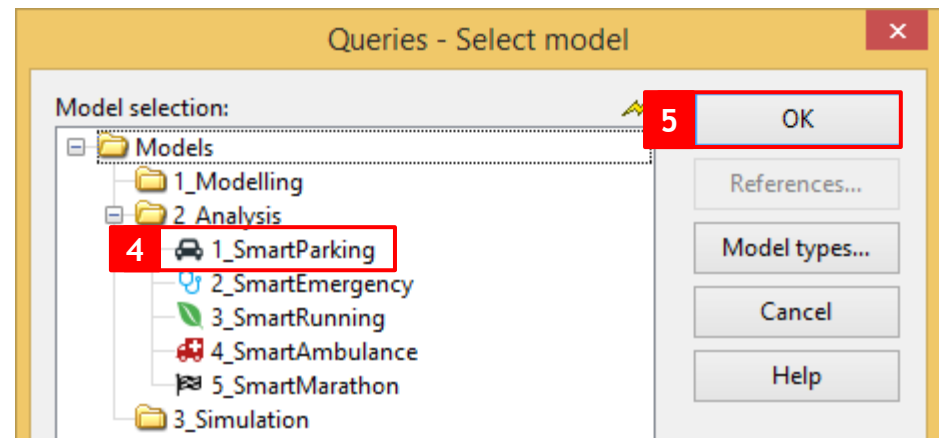
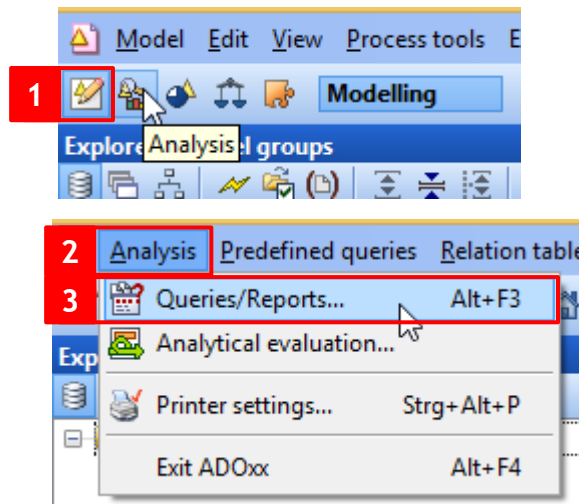
- The city of Vienna wants to change its smart traffic signs for the Eurovision Song Contest. You are in charge! The first step is to find the relevant signs via queries
 - Open the **Modelling Toolkit**
 - > Log in
 - Open the **1_SmartParking** model (doubleclick) within the model group **2_Analysis**



15

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)

Queries

Query scope

☐ Queries on models (model attributes)

☒ Queries on model contents

Standardised queries

Query:

1 Get all objects of class...

Get all objects of class ... with attribute ...

Get object ... of class ...

Get all objects connected with the object ... of class ... with the relation ...

Get all connectors of relation ...

Get all connectors of relation ... with attribute ...

Input field

Get all objects of 2 Traffic_Signs

Add 3 Evaluate

User defined queries

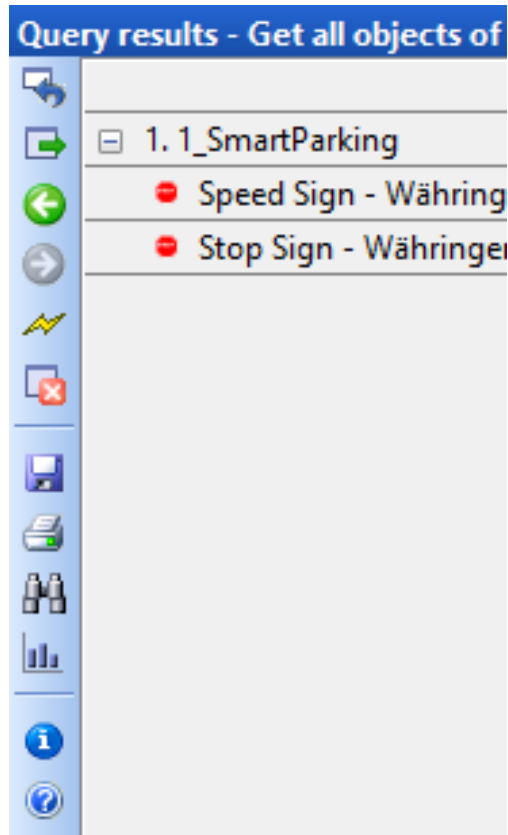
AND OR DIFF Clear







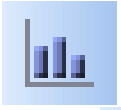


☒ Show attributes in columns

< Back Execute Attributes... Model info... Cancel Help

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



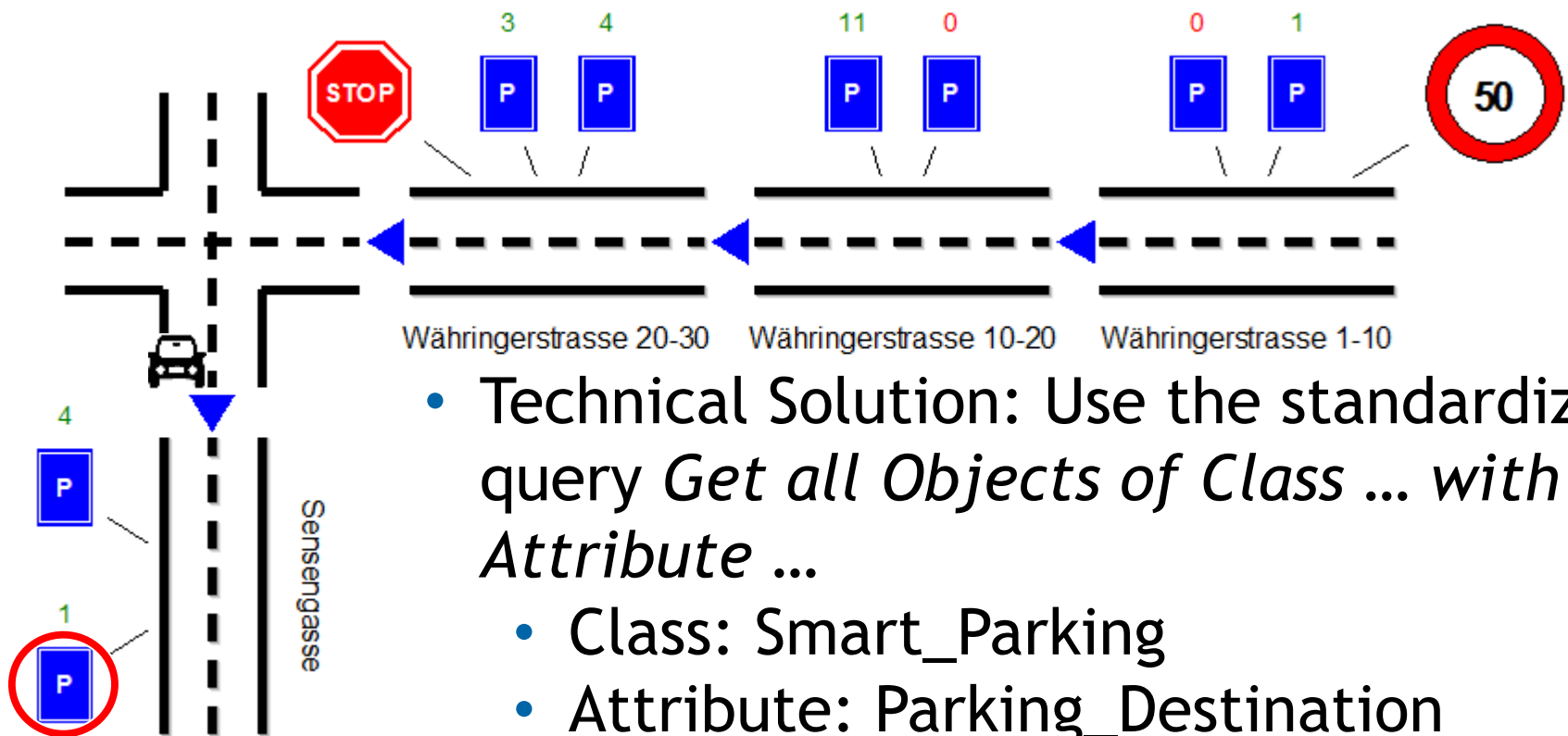
www.theparkerapp.com

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

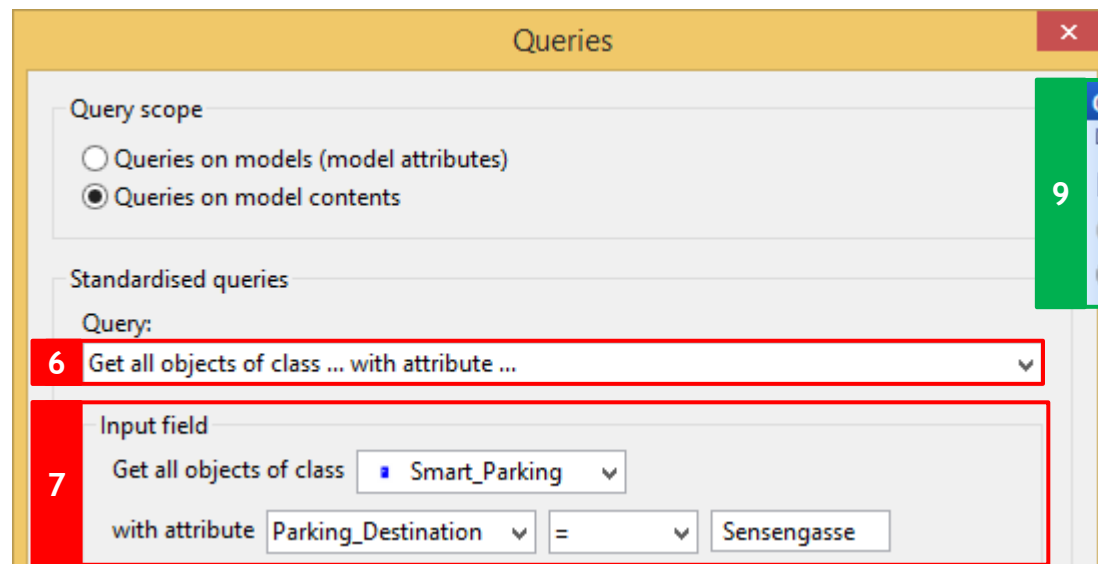
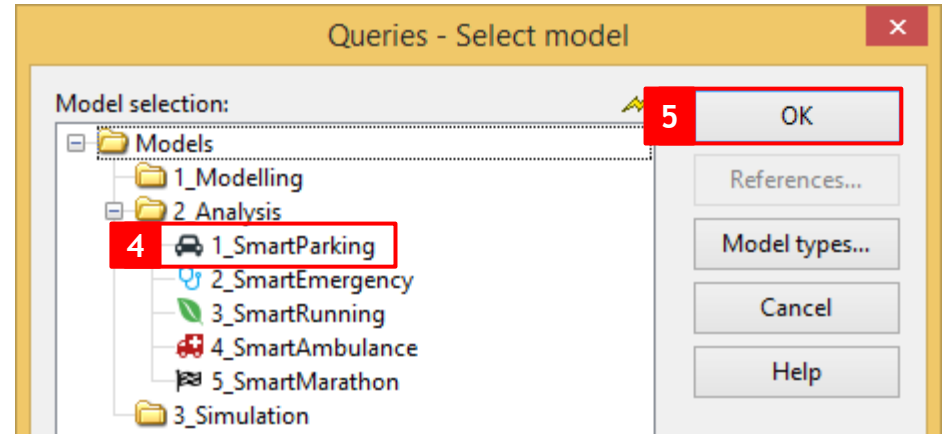
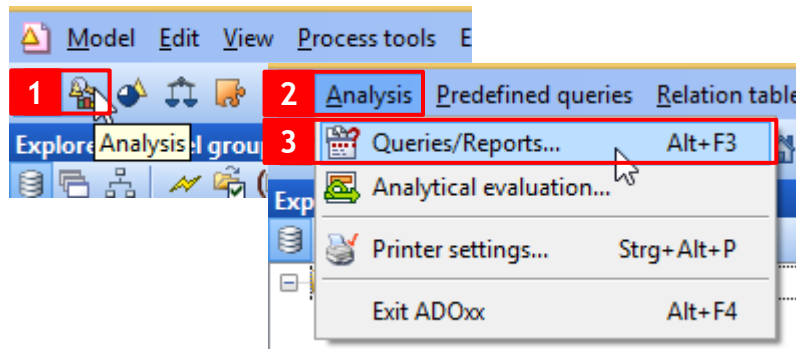
20

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



Demo 1: Solution



9. Query results - Get all objects of class "Smart_Parking" with attribute

	Parking_Destination
1. 1_SmartParking	
Smart Parking Spot - Sensengasse 14	Sensengasse
Smart Parking Spot - Sensengasse 2	Sensengasse



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

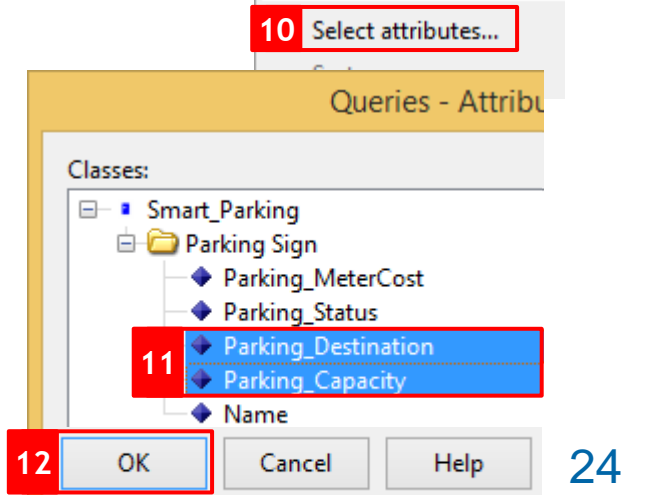
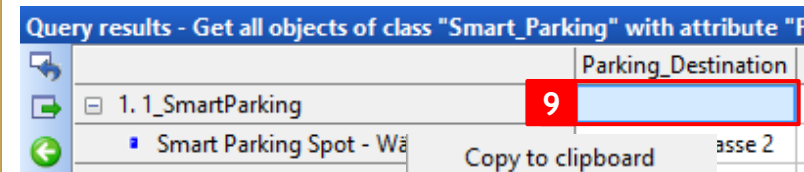
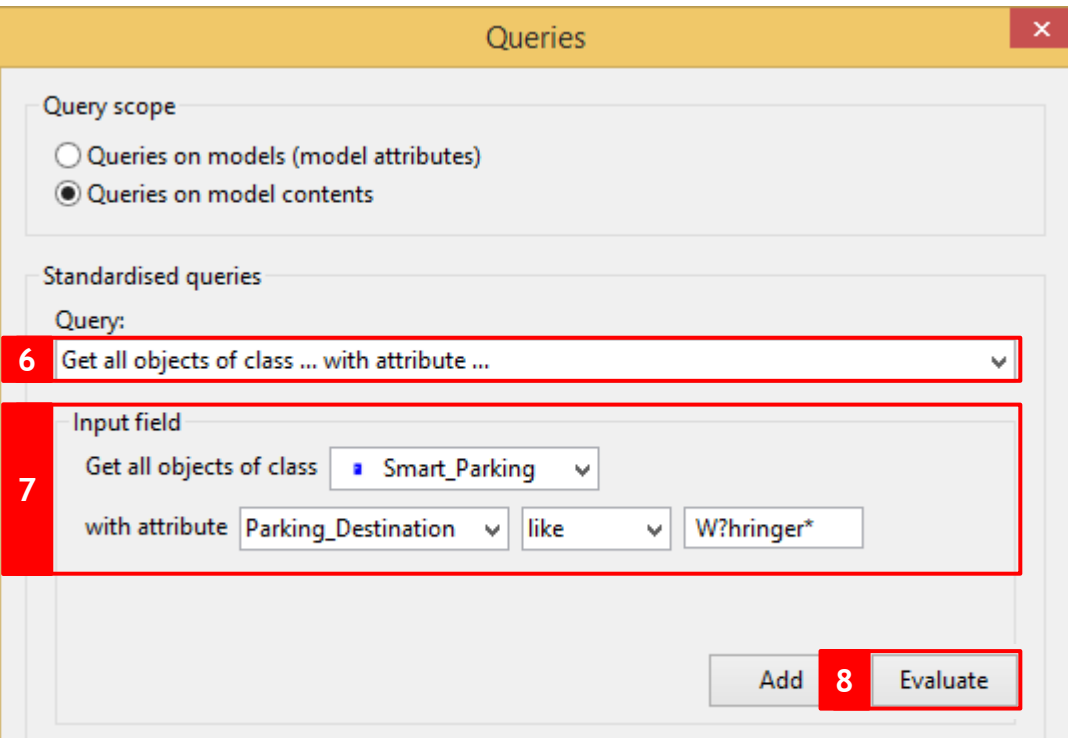
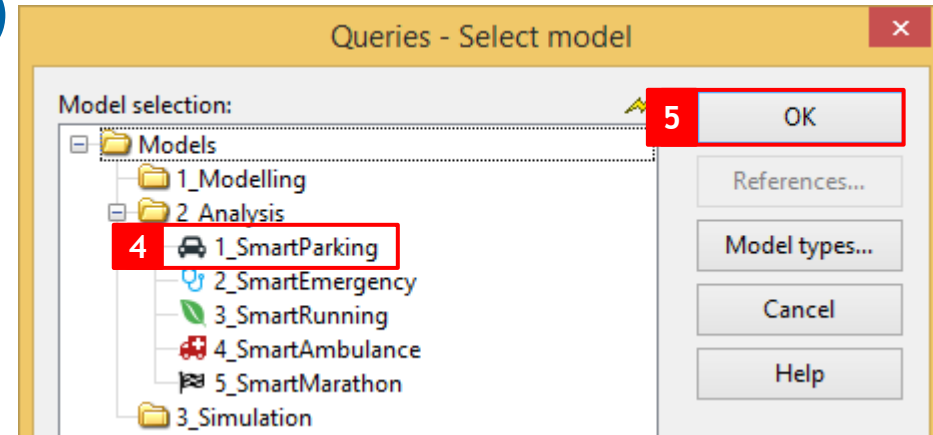
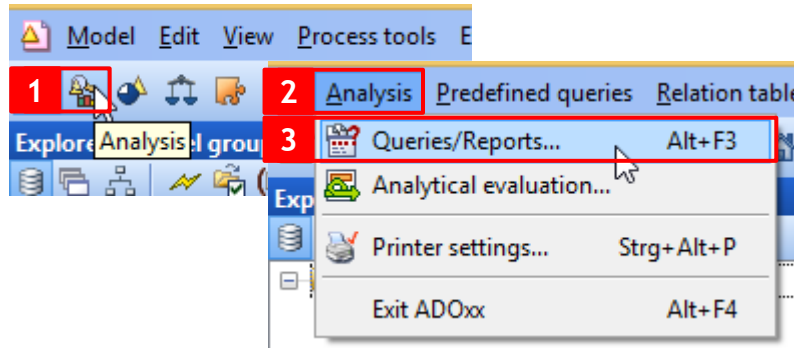
- You need a parking spot in *Währingerstrasse*. Find all parking spots in *Währingerstrasse* and display their attributes in the results window
 - Hint:
 - > Use the *like* operator to compare attributes and '?' or '*' as wildcard
 - > If you cannot find the 'ä' key you can use a wildcard

The diagram illustrates a street layout for Währingerstrasse with three segments: Währingerstrasse 20-30, Währingerstrasse 10-20, and Währingerstrasse 1-10. Each segment has two parking spots marked with a blue 'P' in a circle. The first segment also has a red octagonal 'STOP' sign. A car is shown driving on the right side of the street. A red arrow points from the car towards the query results window. A 'Sensel' sensor is located at the bottom left of the diagram. The query results window displays the following data:

Query results - Get all objects of class "Smart_Parking" with attribute "Pa	
	Parking_Destination
1. 1_SmartParking	
Smart Parking Spot - Währingerstrasse 1	Währingerstrasse 2
Smart Parking Spot - Währingerstrasse 2	Währingerstrasse 8
Smart Parking Spot - Währingerstrasse 3	Währingerstrasse 16
Smart Parking Spot - Währingerstrasse 4	Währingerstrasse 12
Smart Parking Spot - Währingerstrasse 5	Währingerstrasse 20
Smart Parking Spot - Währingerstrasse 6	Währingerstrasse 22

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

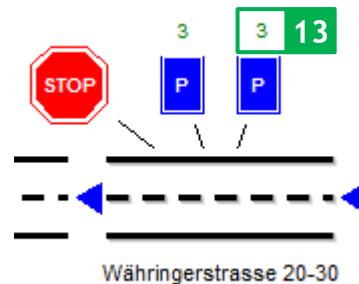
Hands-On 1: Solution (1/2)



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	Währingerstrasse 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

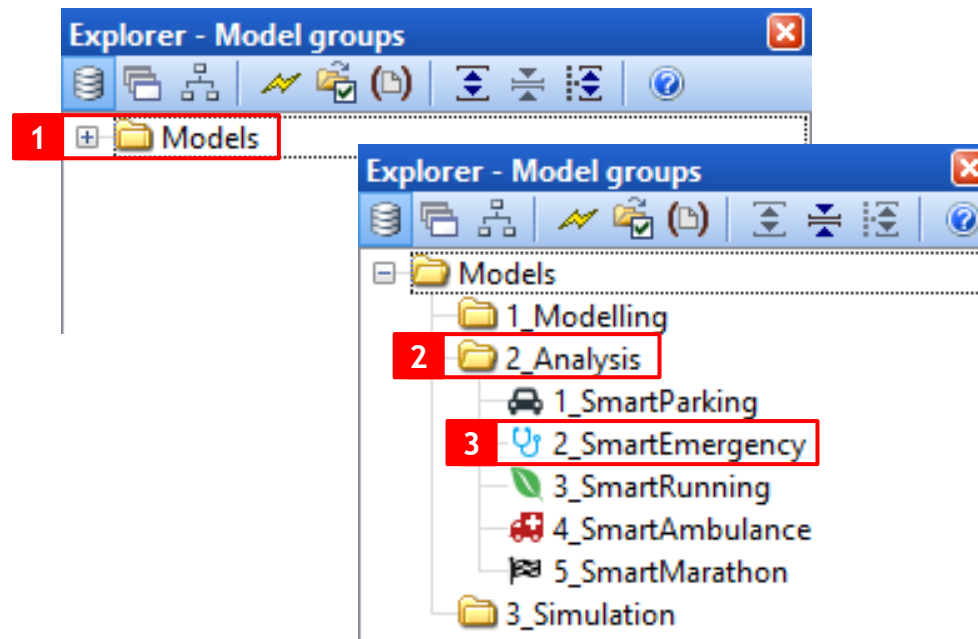
- Basics: Relations and relationship operators, AQL
- Demo: AQL, transitive relations | Smart Traffic Planning
- Hands-on: AQL, transitive relations | Smart Ambulance Navigation

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

Query scope

☐ Queries on models (model attributes)

☒ Queries on model contents

Standardised queries

Query:

Get all objects connected with the object ... of class ... with the relation ...

Get all connectors of relation ...

Get all connectors of relation ... with attribute ...

1

Get all objects connected with the object ... of class ... with the relation ...

Input field

Get all objects connected with the object

of class ☐ Area

with the relation

Add Evaluate

2

AND OR DIFF Clear

4

Execute

3

Get all objects connected with the object ... of class ... with the relation ...

Get all connectors of relation ...

Get all connectors of relation ... with attribute ...

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**

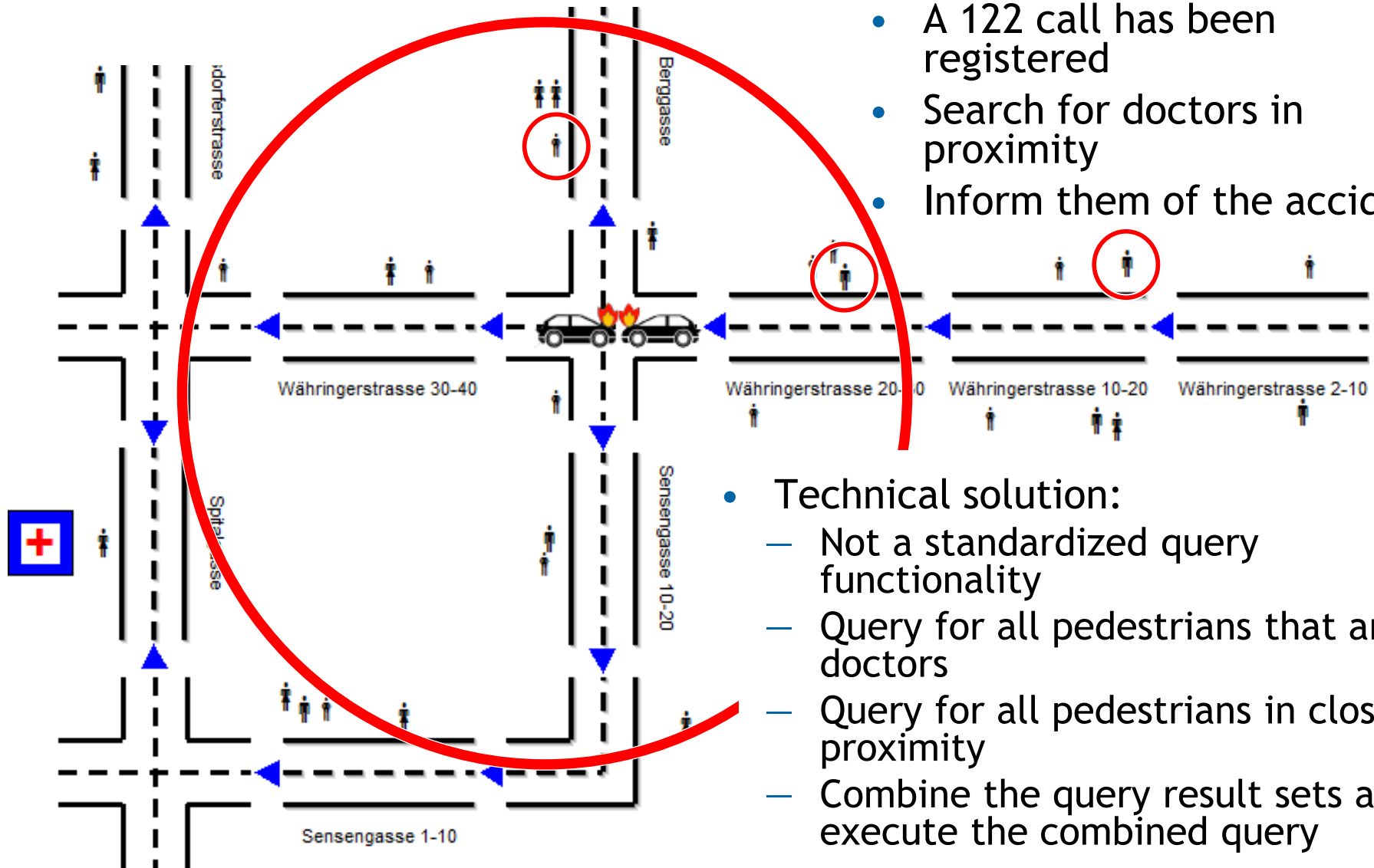
28

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)

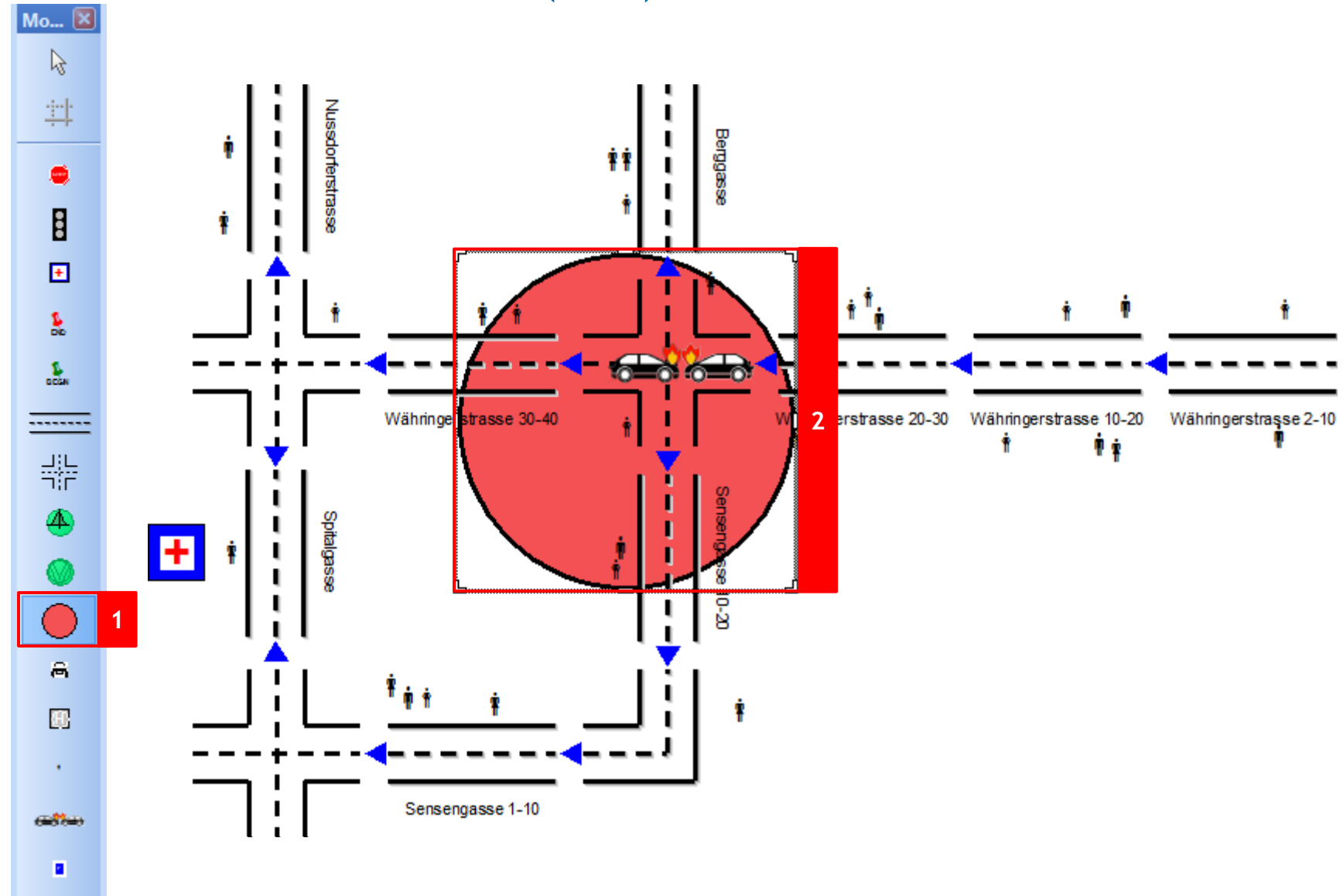


- A 122 call has been registered
- Search for doctors in proximity
- Inform them of the accident

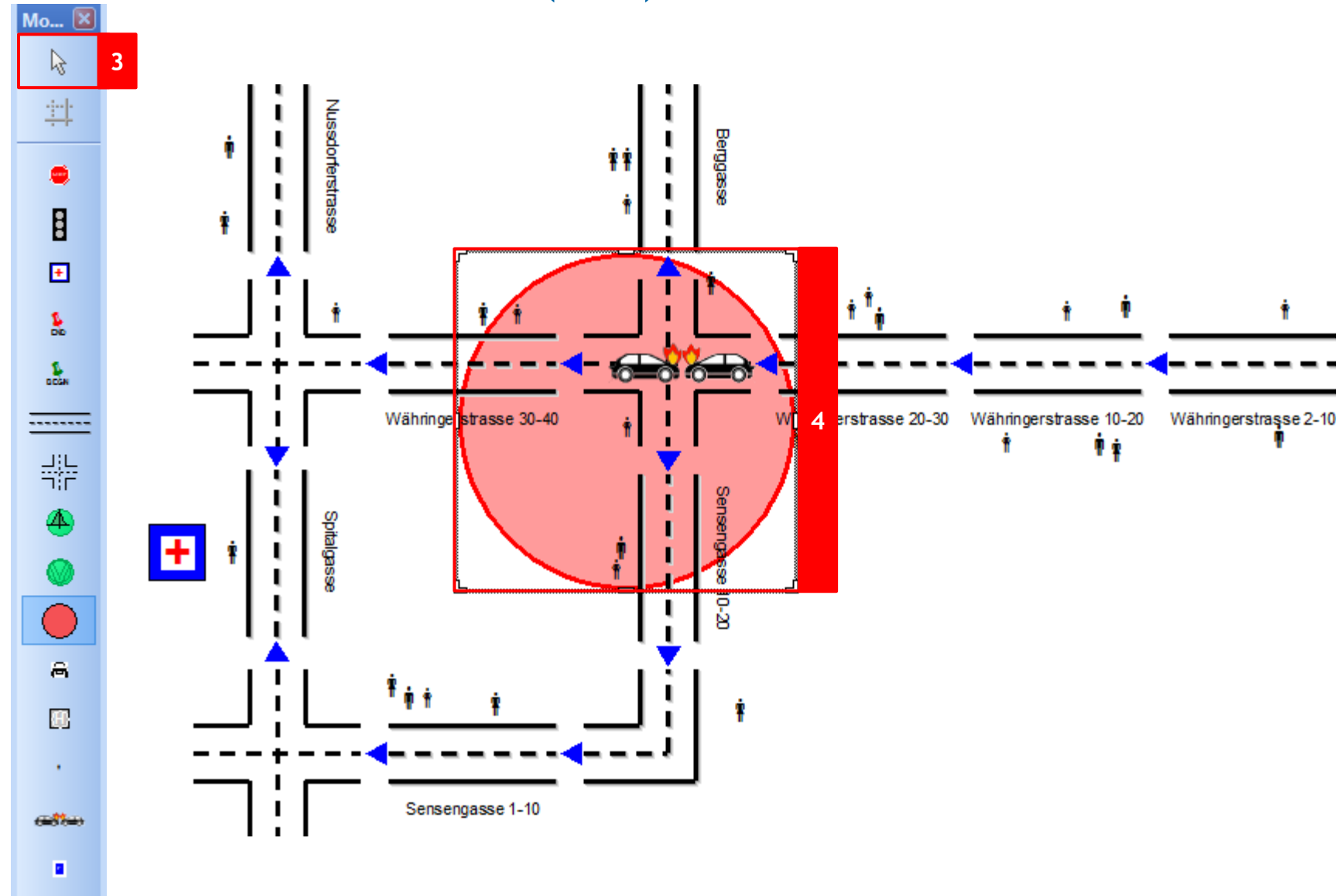
- Technical solution:
 - Not a standardized query functionality
 - Query for all pedestrians that are doctors
 - Query for all pedestrians in close proximity
 - Combine the query result sets and execute the combined query

30

Demo 2: Solution (1/5)



Demo 2: Solution (2/5)



Demo 2: Solution (3/5)

The screenshot displays the OMLAB simulation environment. On the left is a toolbar with various icons for navigation and object placement. The main area shows a street intersection with labels for 'Nussdorferstrasse', 'Berggasse', 'Währingerstrasse 30-40', 'Währingerstrasse 20-30', 'Währingerstrasse 10-20', 'Währingerstrasse 2-10', 'Spitalgasse', and 'Sensengasse'. A red circle highlights a fire incident at the intersection, with a red rectangle indicating the area of interest. A red box with the number '3' is next to the mouse cursor icon in the toolbar. A red box with the number '4' is next to the red rectangle. A red box with the number '5' is next to the 'Name' field in the form, which contains 'Call 1001'. A red box with the number '6' is next to the 'Type' field in the form, which contains 'Emergency'. A red box with the number '7' is next to the 'Description' field in the form. The form is titled 'Area-26845 (Area)'.

Mo... x

3

Nussdorferstrasse

Berggasse

Währingerstrasse 30-40

Währingerstrasse 20-30

Währingerstrasse 10-20

Währingerstrasse 2-10

Spitalgasse

Sensengasse

4

Area-26845 (Area)

7 x

Name:

5 Call 1001

Type:

6 Emergency

Form

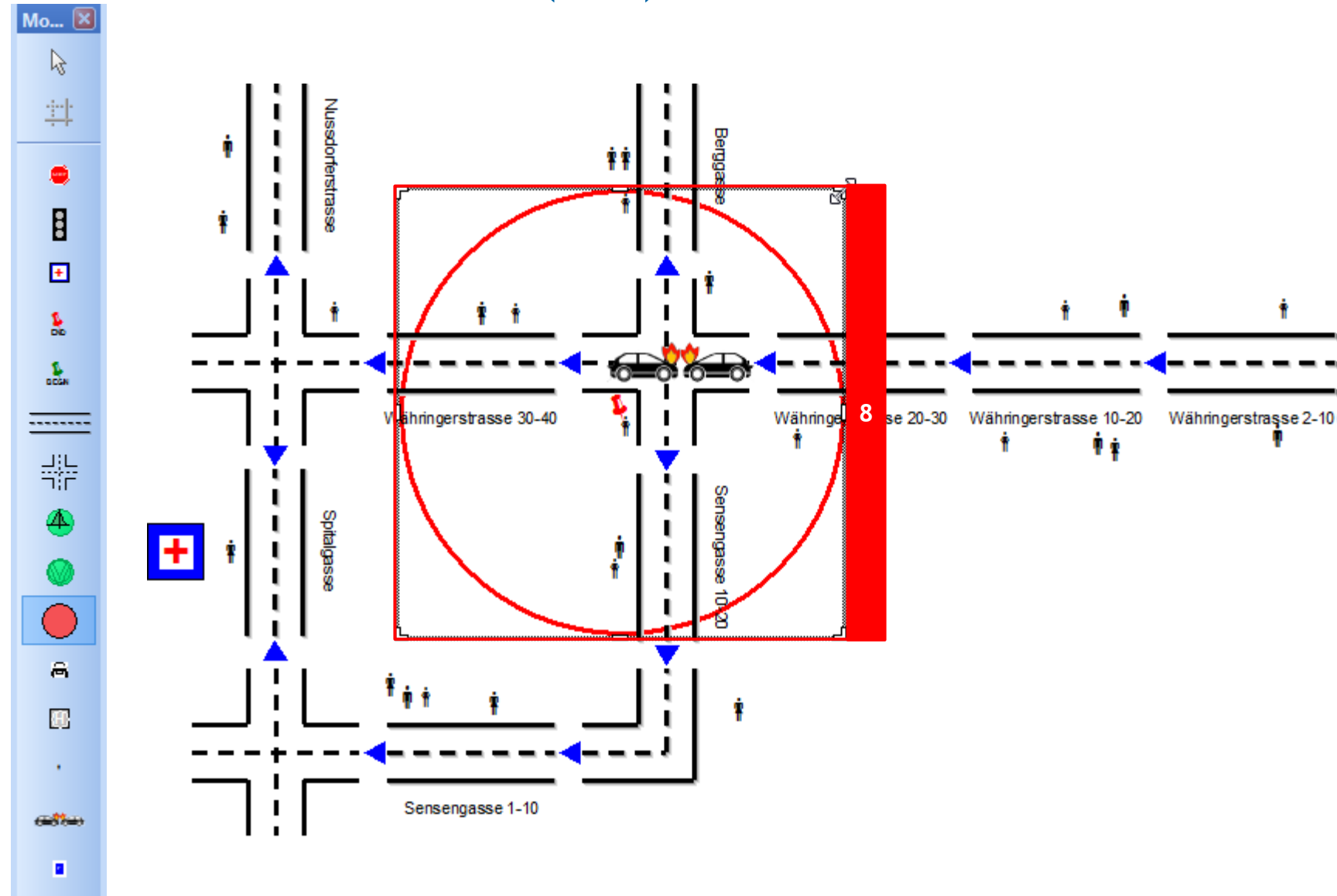
● Ellipse

○ Rectangle

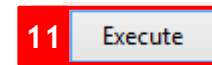
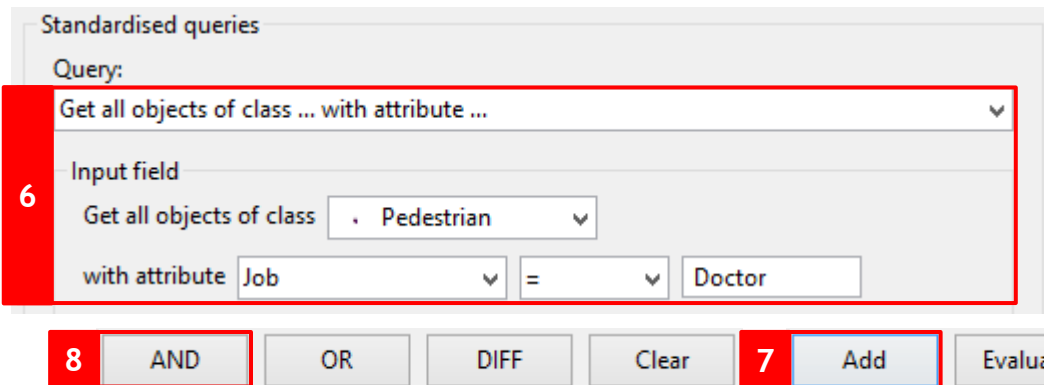
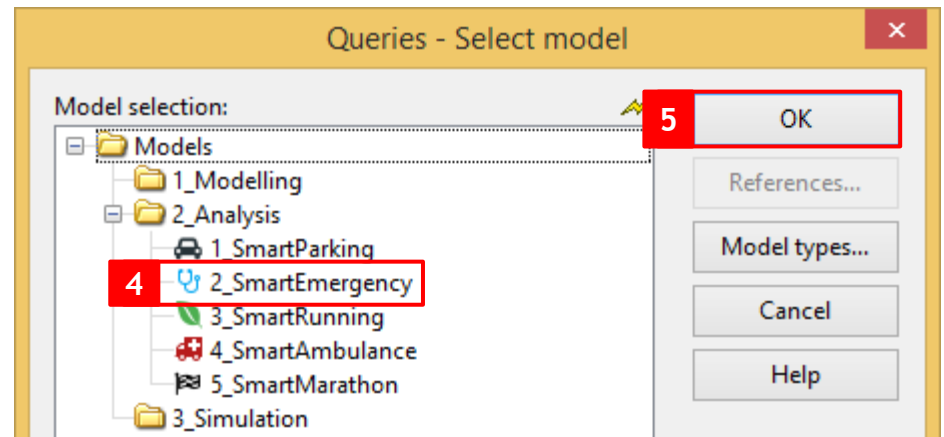
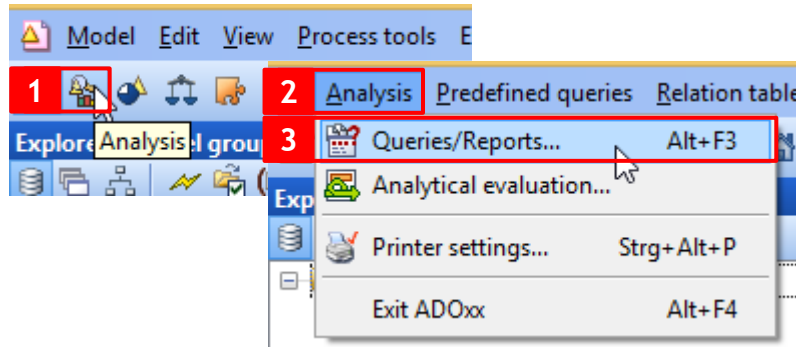
Description

33

Demo 2: Solution (4/5)



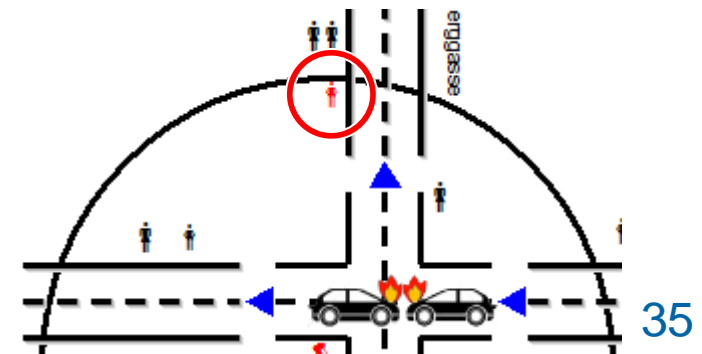
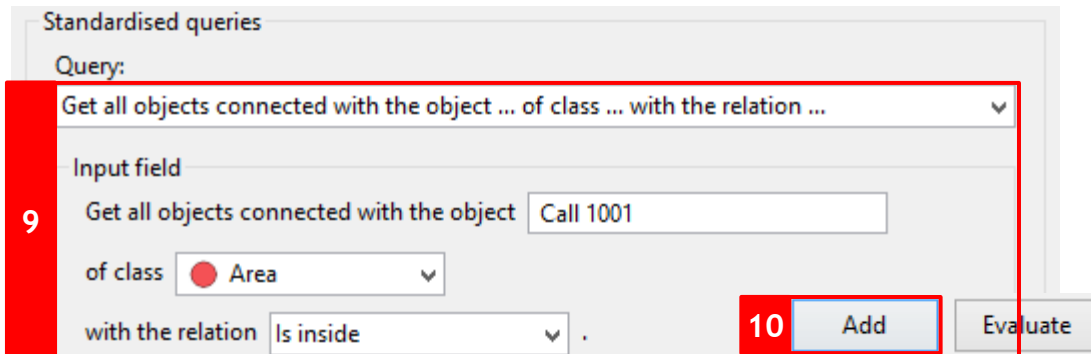
Demo 2: Solution (5/5)



Query results - (<"Pedestrian">[?"Job" = "Doctor"]) AND ((!"Call 1001"

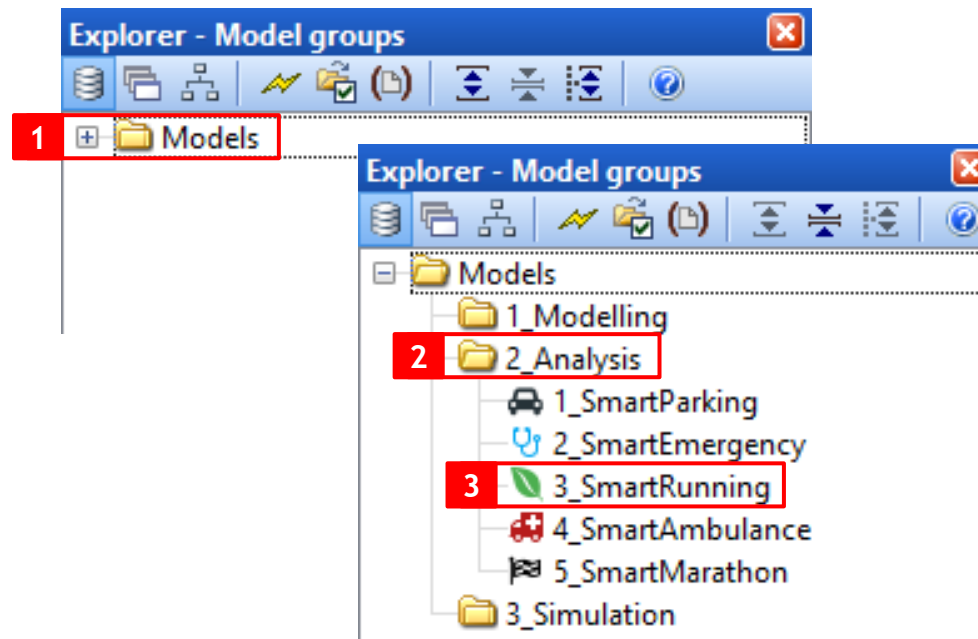
	Name	Job	Contact
1. 2_SmartEmergency			
• Johanna Plato	Johanna Plato	Doctor	43-1-4277-789 01

12

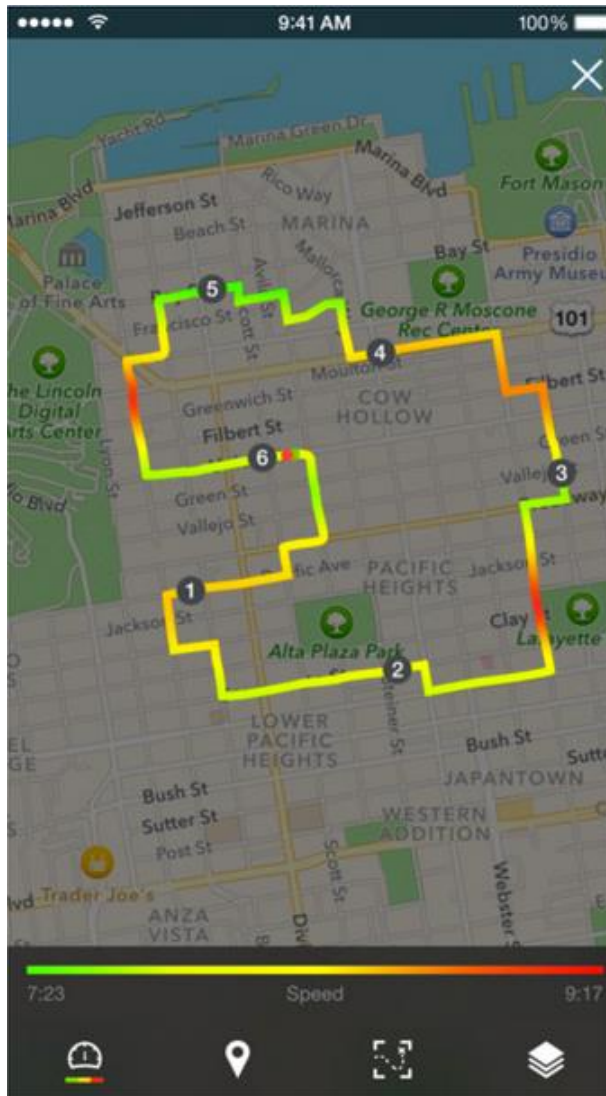


Combination of standardized queries using set operators

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



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

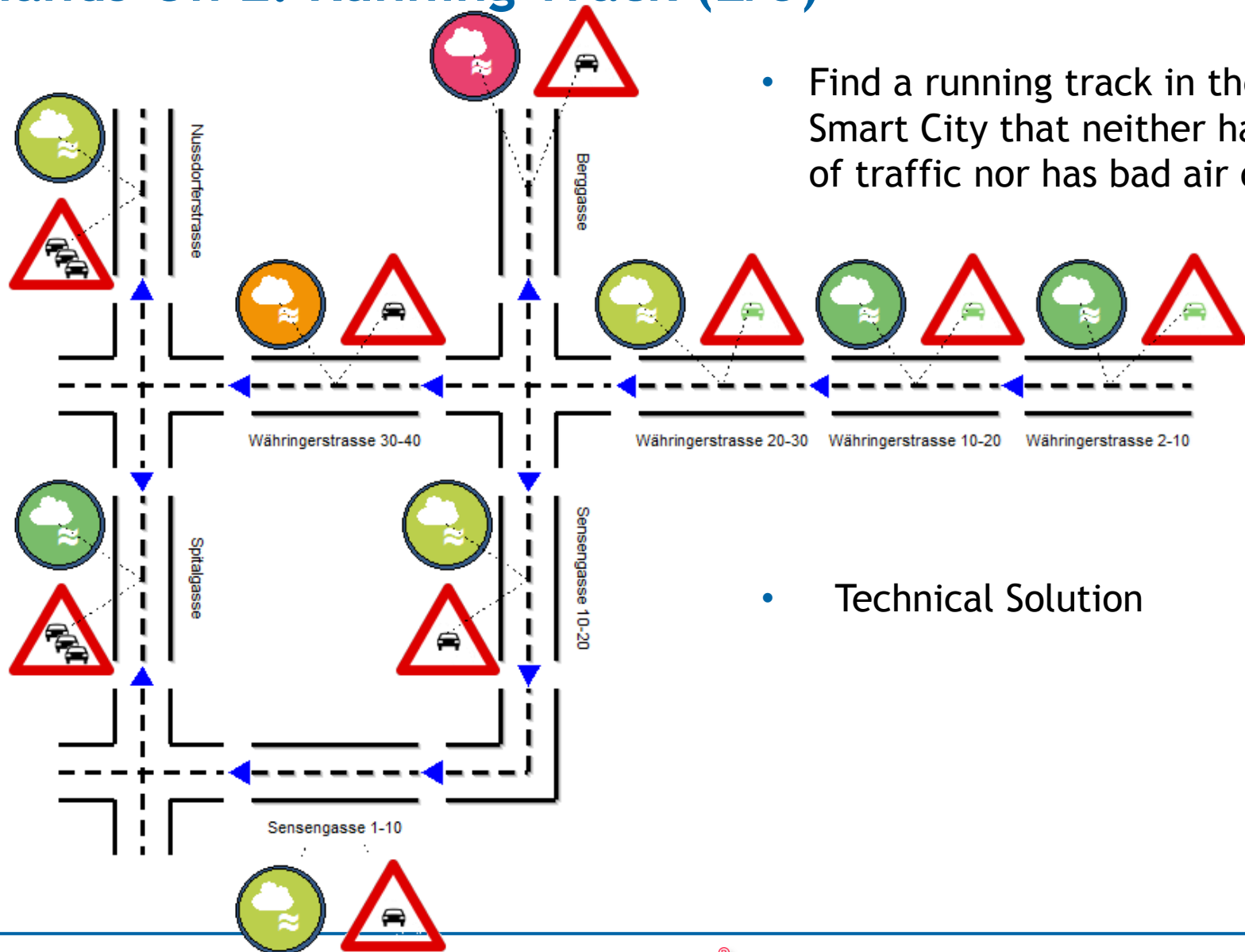


www.runtastic.com

- 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.

37

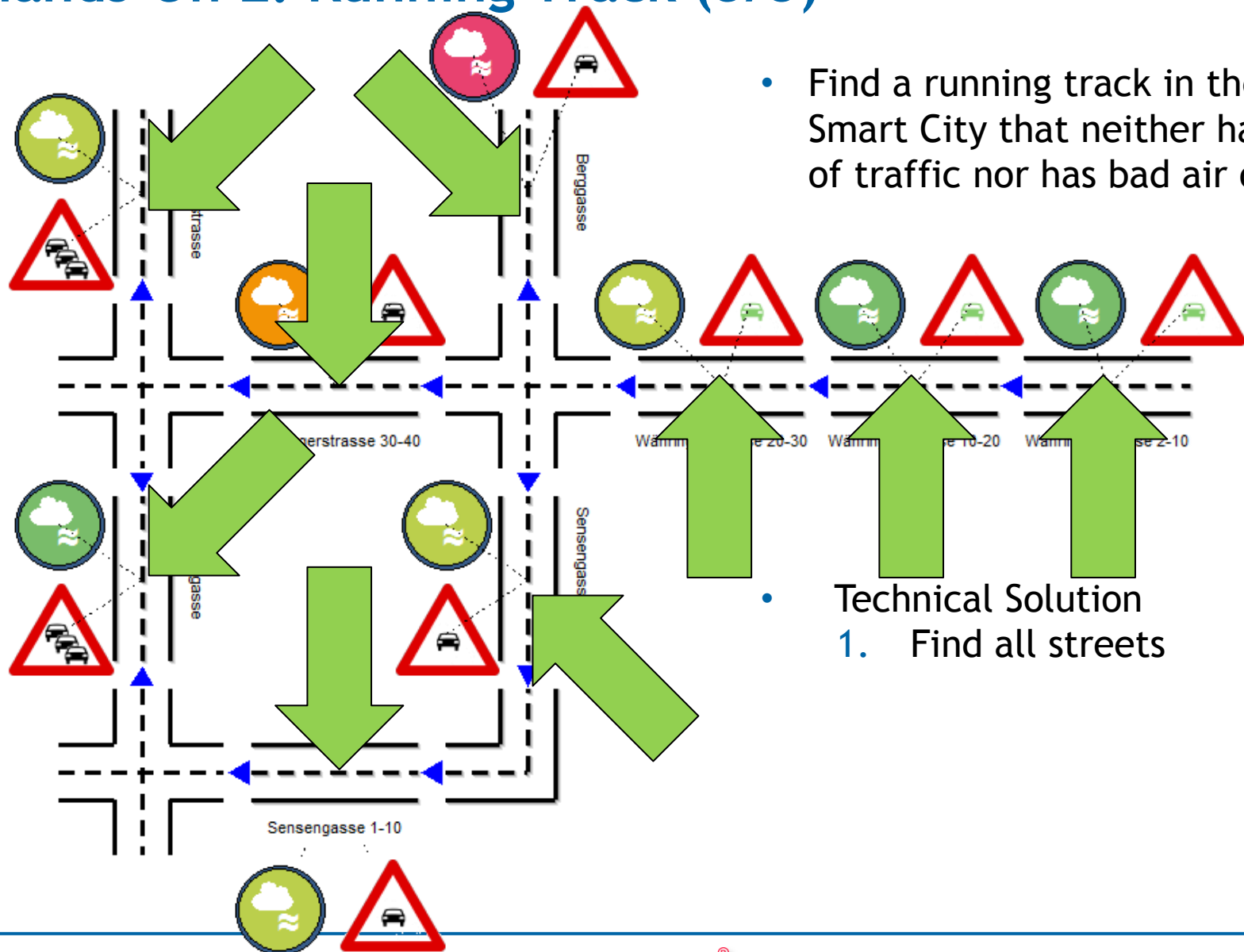
Hands-On 2: Running Track (2/5)



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

- Technical Solution

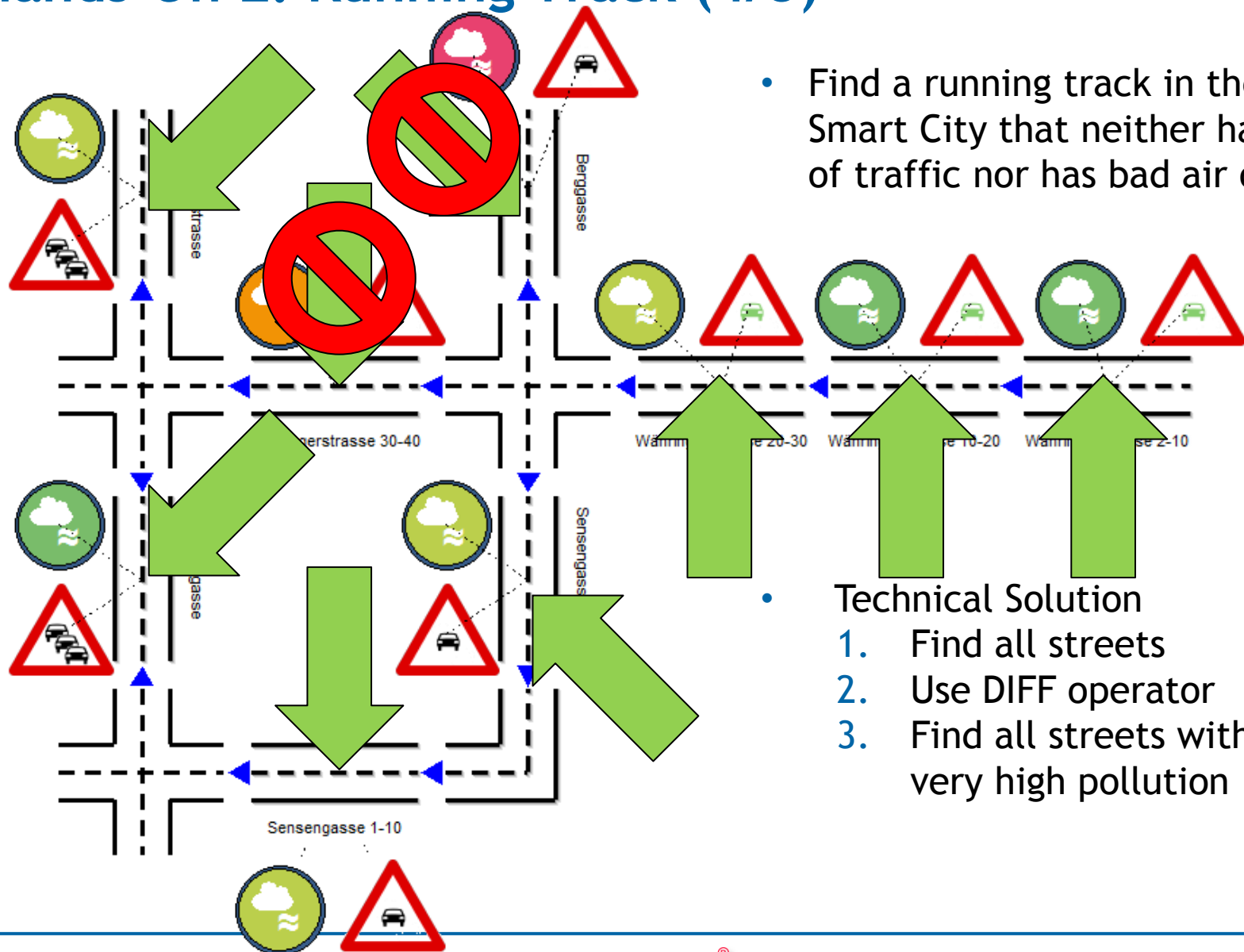
Hands-On 2: Running Track (3/5)



- 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

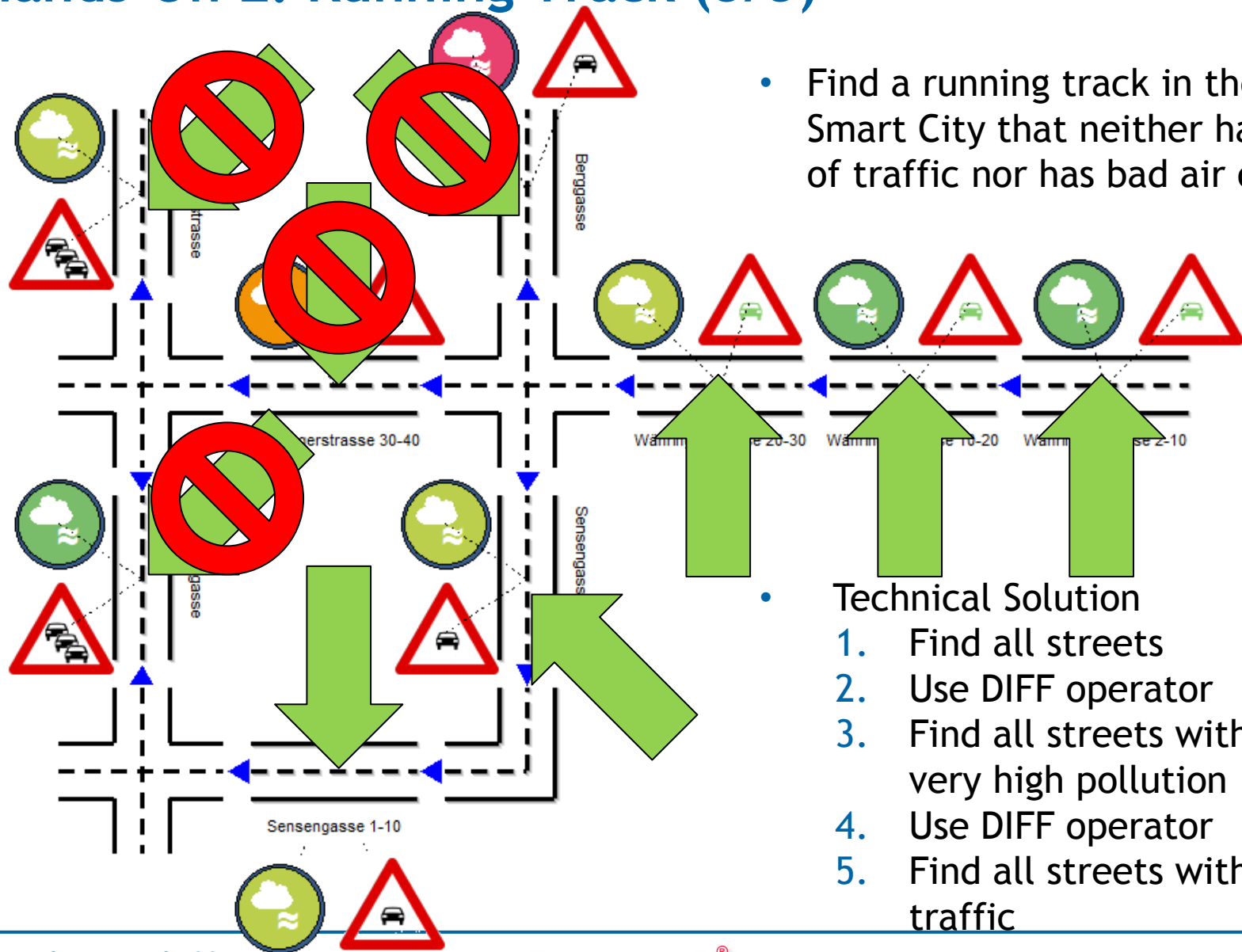
Hands-On 2: Running Track (4/5)



- 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

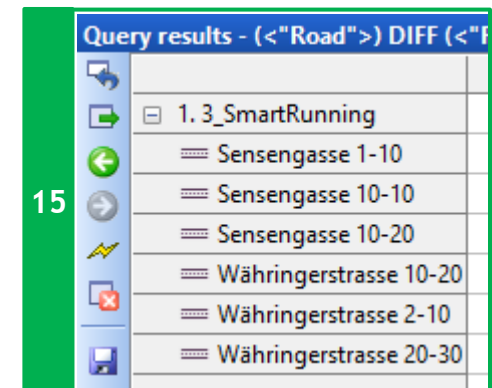
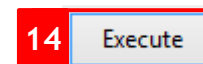
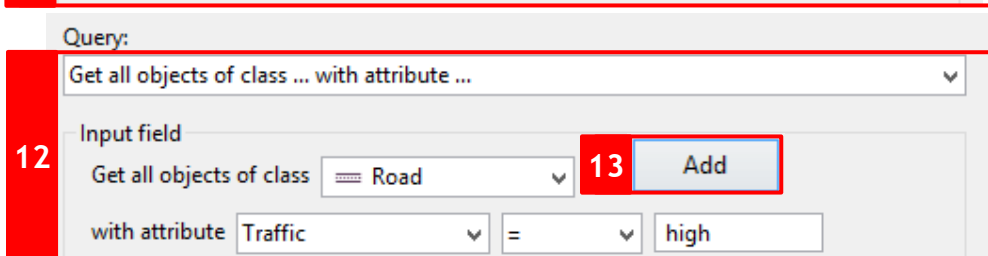
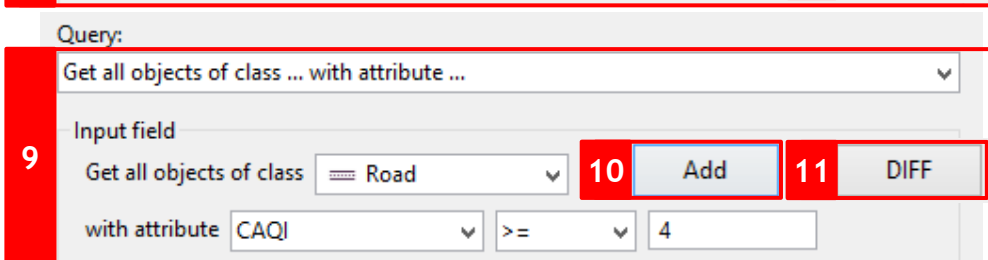
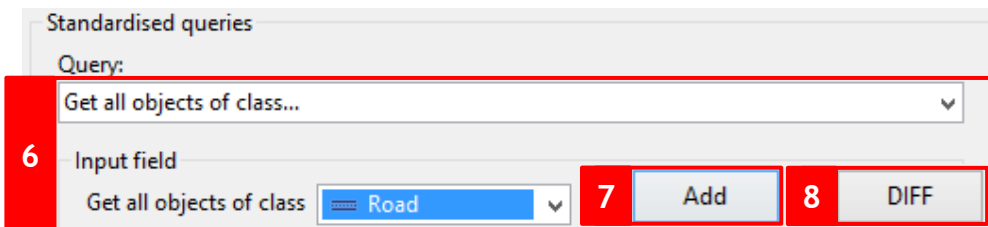
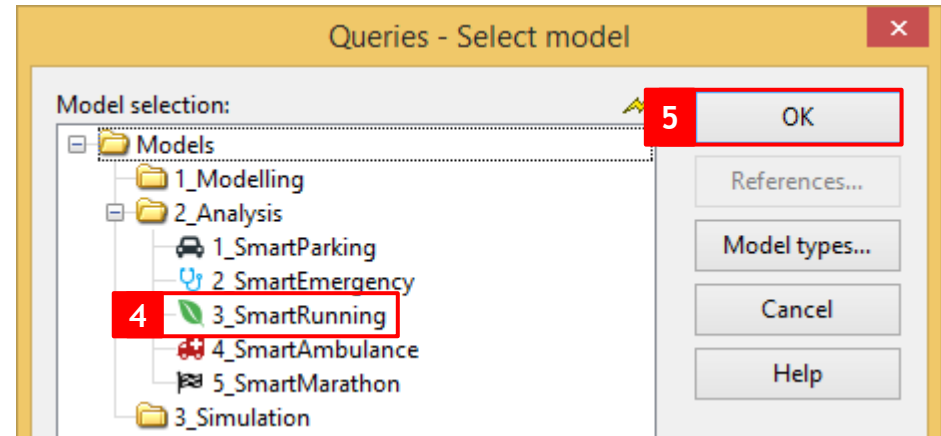
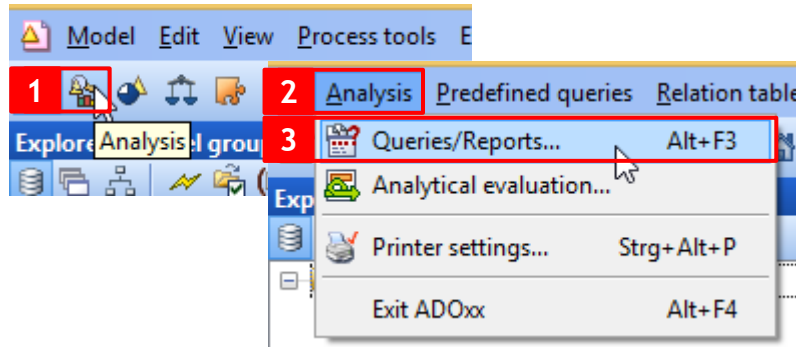
Hands-On 2: Running Track (5/5)



- 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

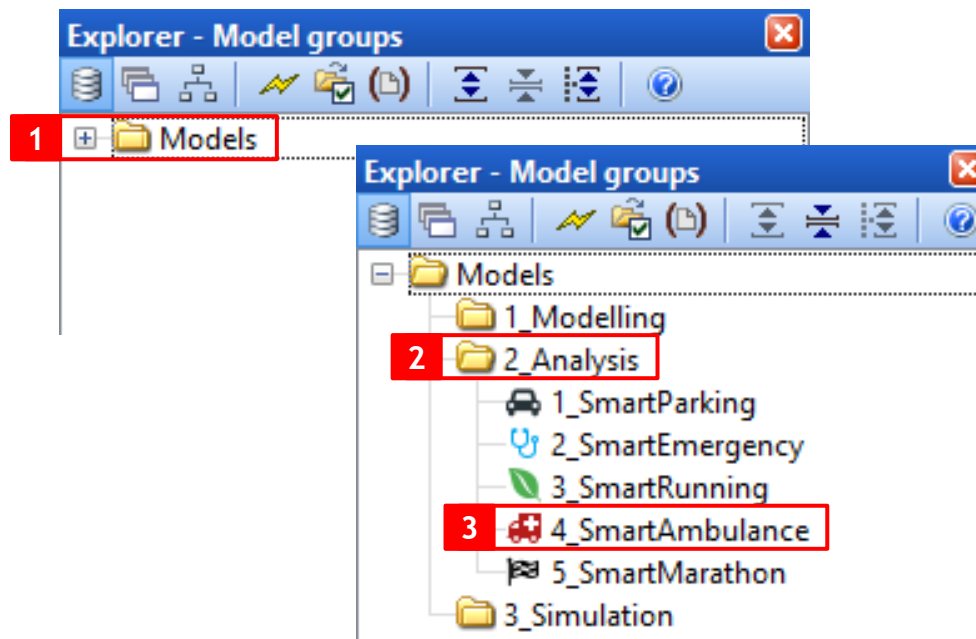
- Basics: [Relations and relationship operators](#), [AQL](#)
- Demo: [AQL, transitive relations](#) | Smart Traffic Planning
- Hands-on: [AQL, transitive relations](#) | Smart Ambulance Navigation

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 screenshot shows a software interface for creating queries. A dropdown menu is open, displaying several query options. The option 'Get all connectors of relation ...' is highlighted with a red box and a red '1'. Below the dropdown, the 'Input field' section shows 'Get all connectors of relation' followed by a dropdown menu with 'SmartCity_Relati' selected. At the bottom, a row of buttons includes '< Back', 'Execute' (highlighted with a red box and a red '2'), 'Attributes...', 'Model info...', 'Cancel', and 'Help'.

Standardised queries

Query:

1 Get all connectors of relation ...

Input field

Get all connectors of relation — SmartCity_Relati

< Back 2 Execute Attributes... Model info... Cancel Help

Get all objects of class...

Get all objects of class ... with attribute ...

Get object ... of class ...

Get all objects connected with the object ... of class ... with the relation ...

1 Get all connectors of relation ...

Get all connectors of relation ... with attribute ...

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

The screenshot shows a table titled 'Query results - Get all connectors of relation "SmartCity_Relati"'. The table has two columns: a connector ID and a description. The results are as follows:

Connector ID	Description
1.4_SmartAmbulance	
Speed-Sign-Sg	Sensengasse 10-20
Speed_Signs-Ws-1	Währingerstrasse 2-10
Speed_Signs-Ws-2	Währingerstrasse 10-20
Stop-Sign-Ws	Währingerstrasse 20-30
Traffic_Lights-14564-19800	Sensengasse 1-10

3

Basics: User-defined Queries & AQL

Queries

Query scope

☐ Queries on models (model attributes)

☒ Queries on model contents

User defined queries

1

```
{("Währingerstrasse":"Traffic_Signs")<-"SmartCity_Relation") OR  
{("Währingerstrasse":"Traffic_Signs")->"SmartCity_Relation")
```

AND OR DIFF Clear

☒ Show attributes in columns

< Back 2 Execute Attributes... Model info... Cancel Help

- 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

Standardised queries

Query:

Get all objects of class ... with attribute ...

Input field

Get all objects of class Traffic_Signs with attribute Type = Under_Const

User defined queries

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

- The result contains all objects connected with construction sites

Query results - (<"Traffic_Signs">["Type" = "Under_Construction"])->"SmartCity_Relation"

1.4_SmartAmbulance
Währingerstrasse 20-30

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

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




1.4_SmartAmbulance
Währingerstrasse 10-20
Währingerstrasse 2-10

User defined queries

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

- The result lists all signs on preceding streets

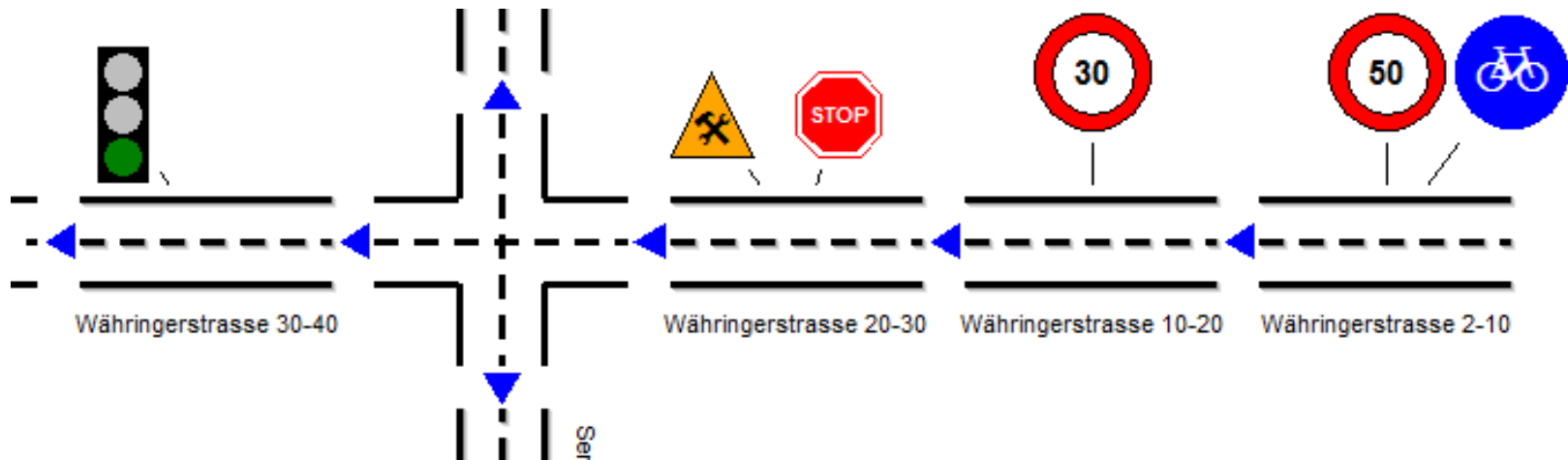
Query results - (<"Traffic_Signs">[?"Type" = "Under_Construction"])->"SmartCity_Relation"<<-"Subsequent"<-"SmartCity_Relation"



1.4_SmartAmbulance
Speed_Signs-Ws-1
Speed_Signs-Ws-2
Traffic_Signs-19836

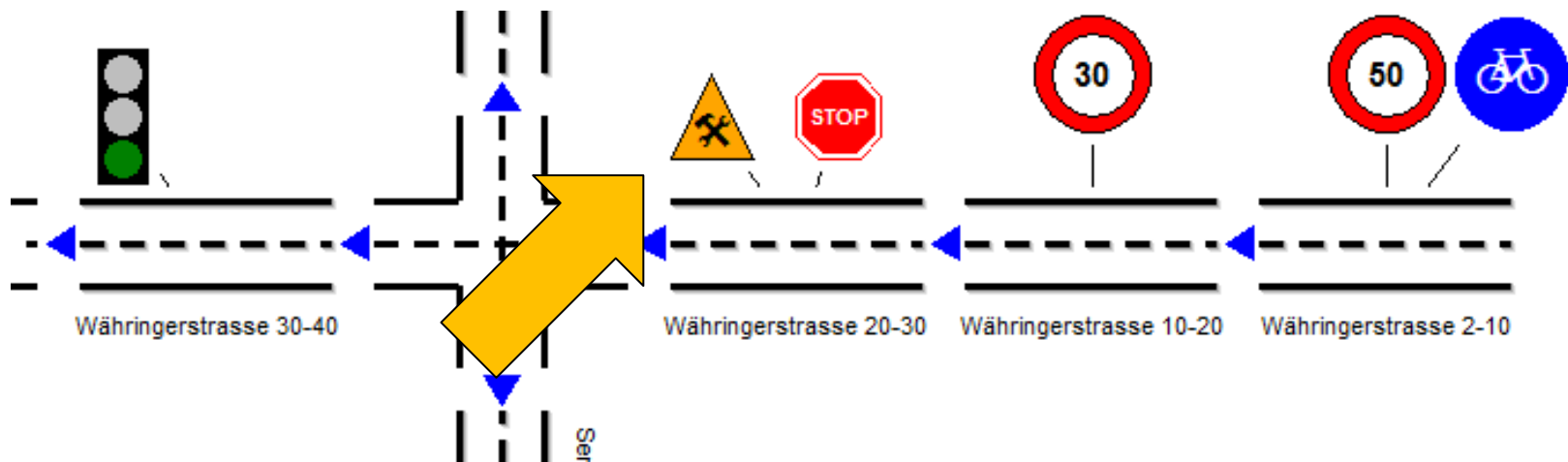
Demo 3: Construction Sites (1/4)

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



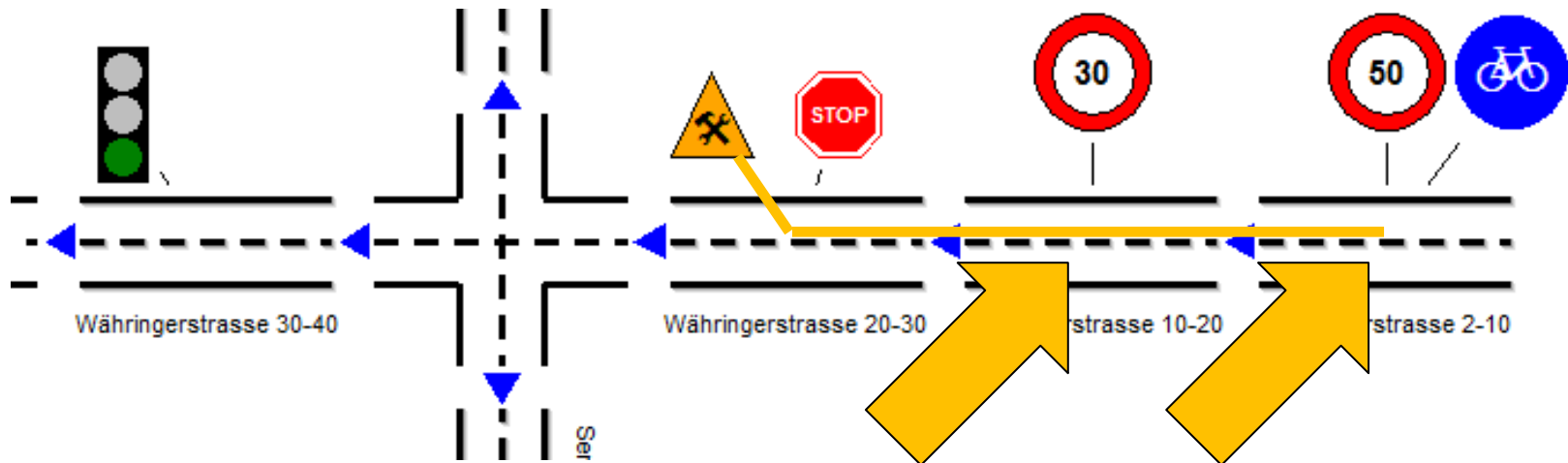
Demo 3: Construction Sites (2/4)

- 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



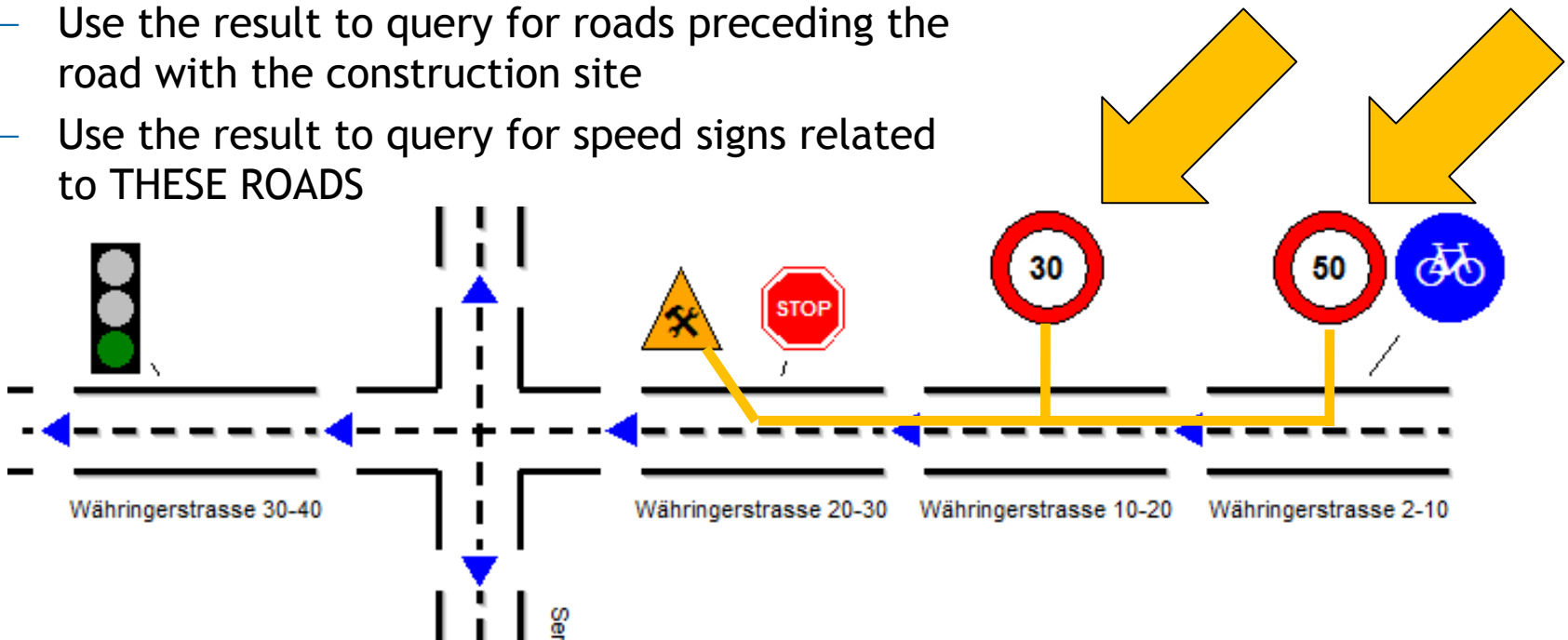
Demo 3: Construction Sites (3/4)

- 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

Standardised queries

Query:

1 Get all objects of class ... with attribute ...

Input field

Get all objects of class ⚠ Traffic_Signs

with attribute Type = Under_Const

1 Add Evaluate

User defined queries

1 (<"Traffic_Signs">[?"Type" = "Under_Construction"])

2 -> "SmartCity_Relation"

3 << - "Subsequent"

4 <- "SmartCity_Relation"

< Back 5 Execute Attributes... Model info... Cancel Help

- 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

6

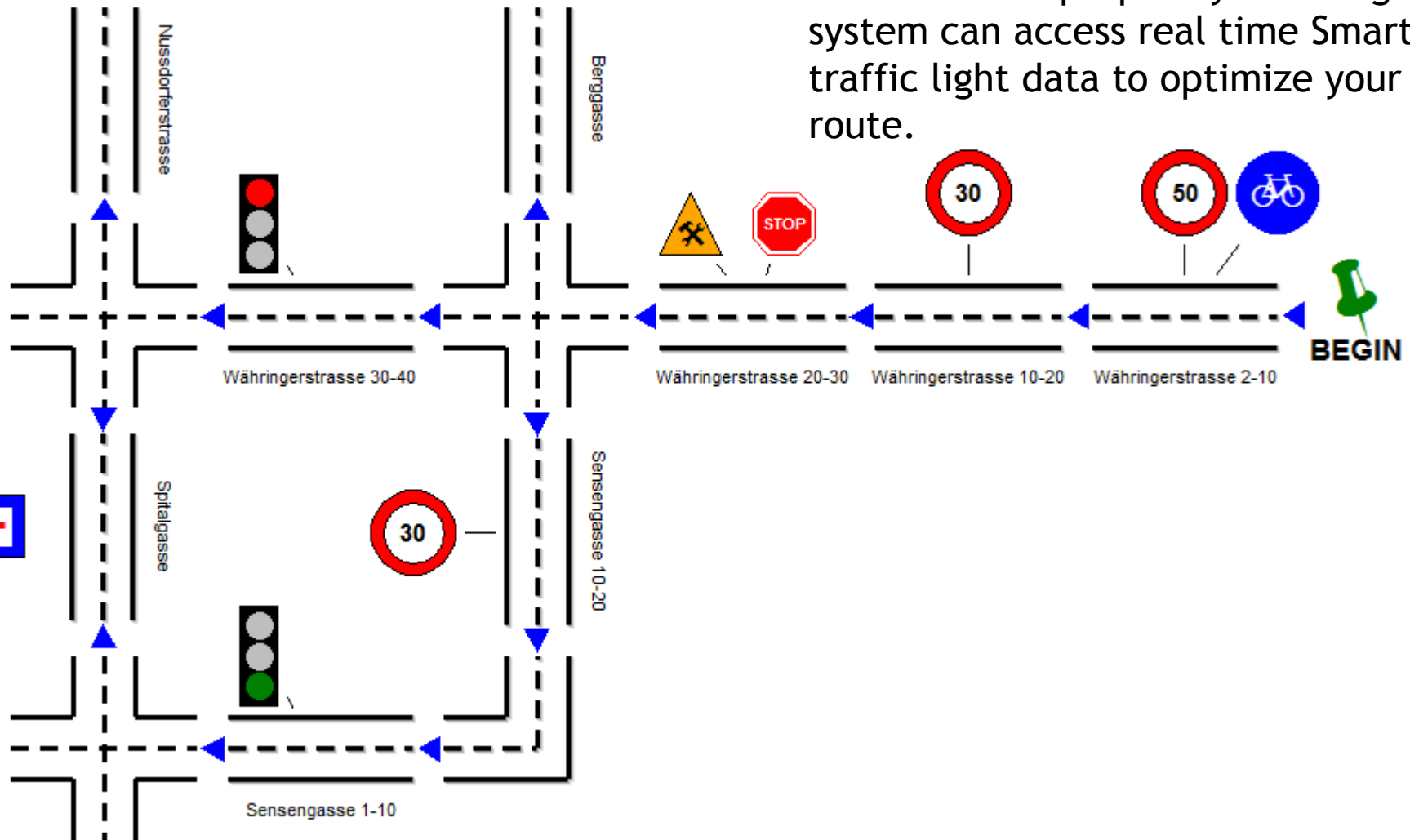
Query results - (<"Traffic_Signs">[?"Type" = "Under_Construction"])

	Type	Max Speed
1. 4_SmartAmbulance		
⚠ Speed_Signs-Ws-1	Speed_Limit	50
⚠ Speed_Signs-Ws-2	Speed_Limit	30

53

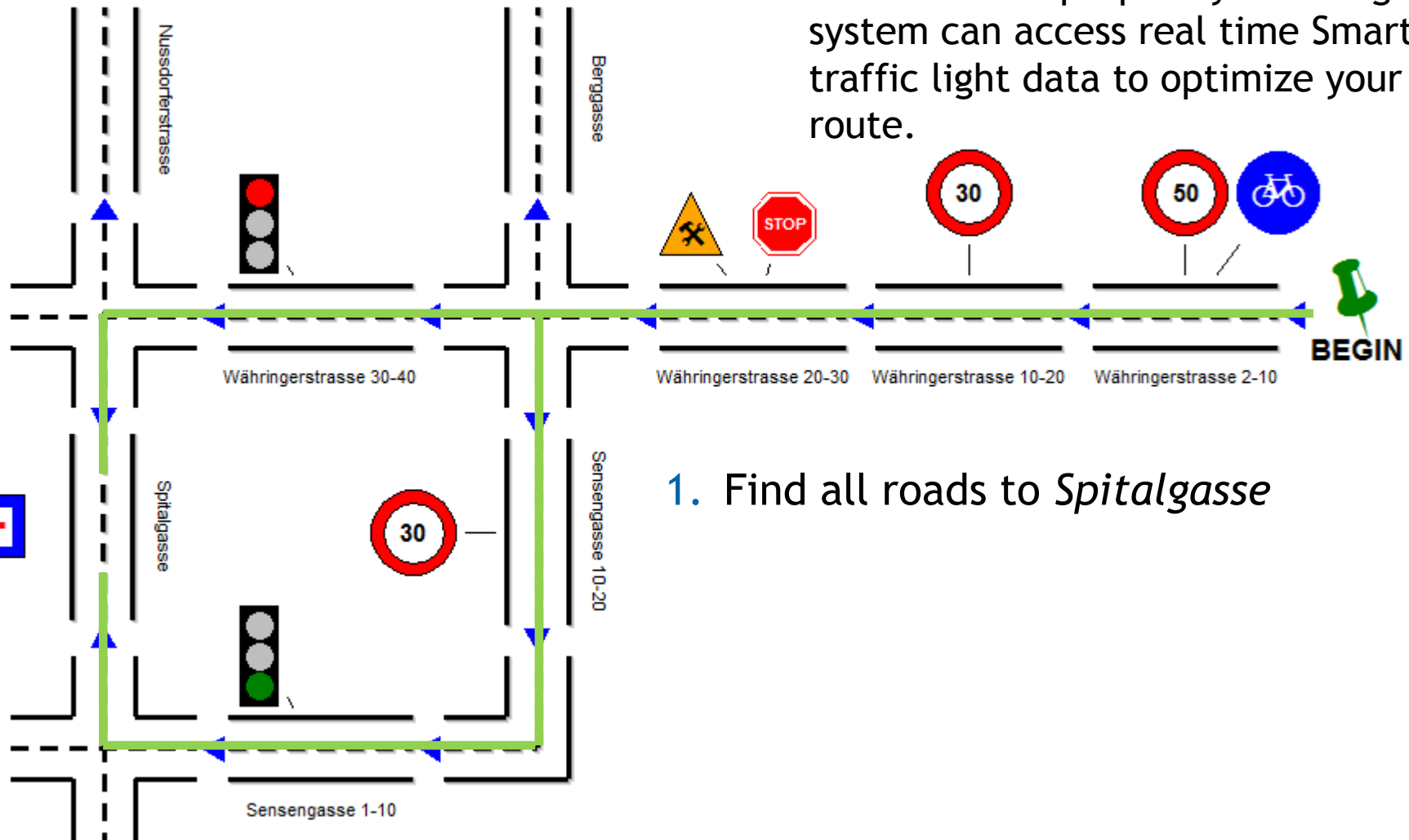
Hands-On 3: Emergency Service in a Smart City (1/5)

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.



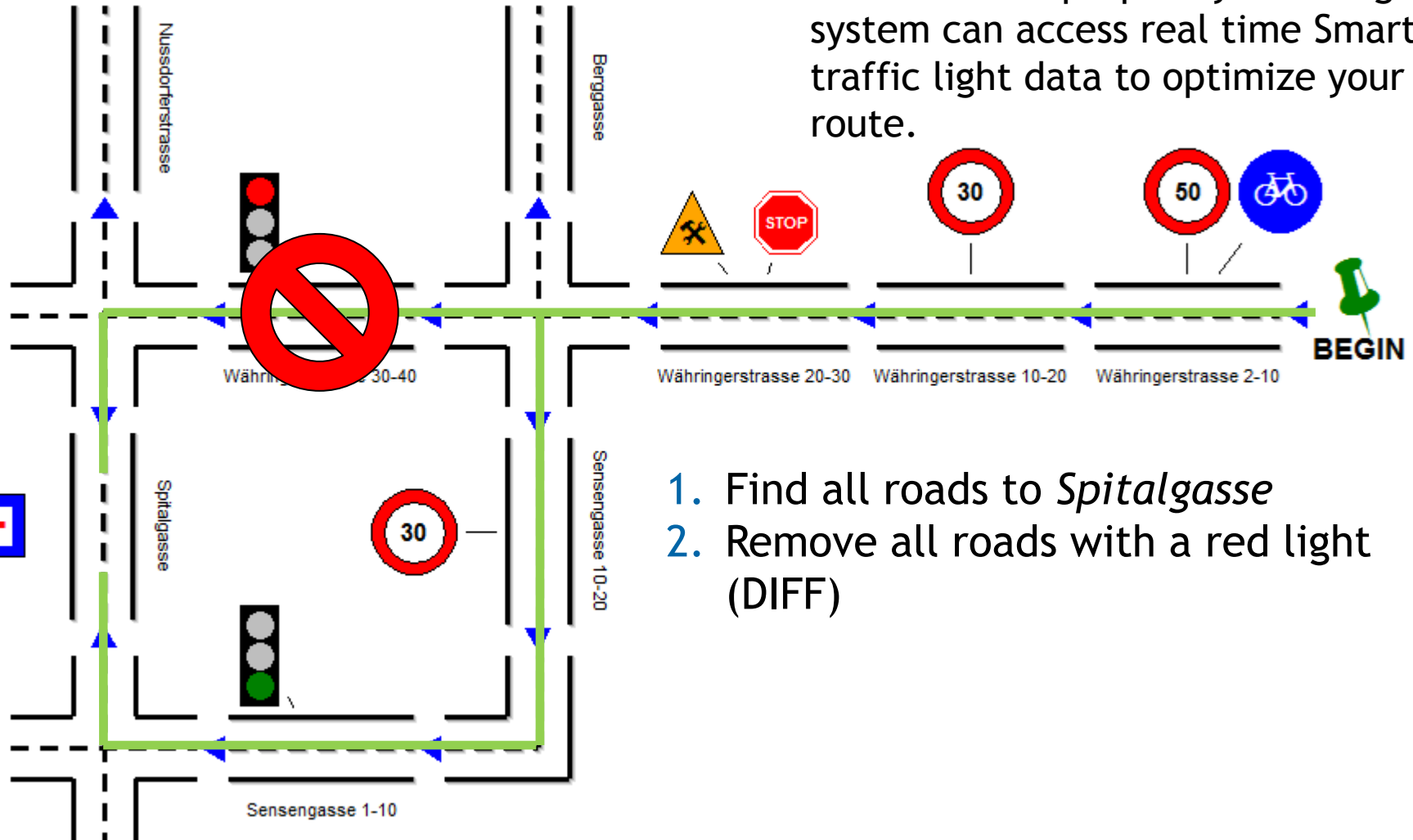
Hands-On 3: Emergency Service in a Smart City (2/5)

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.



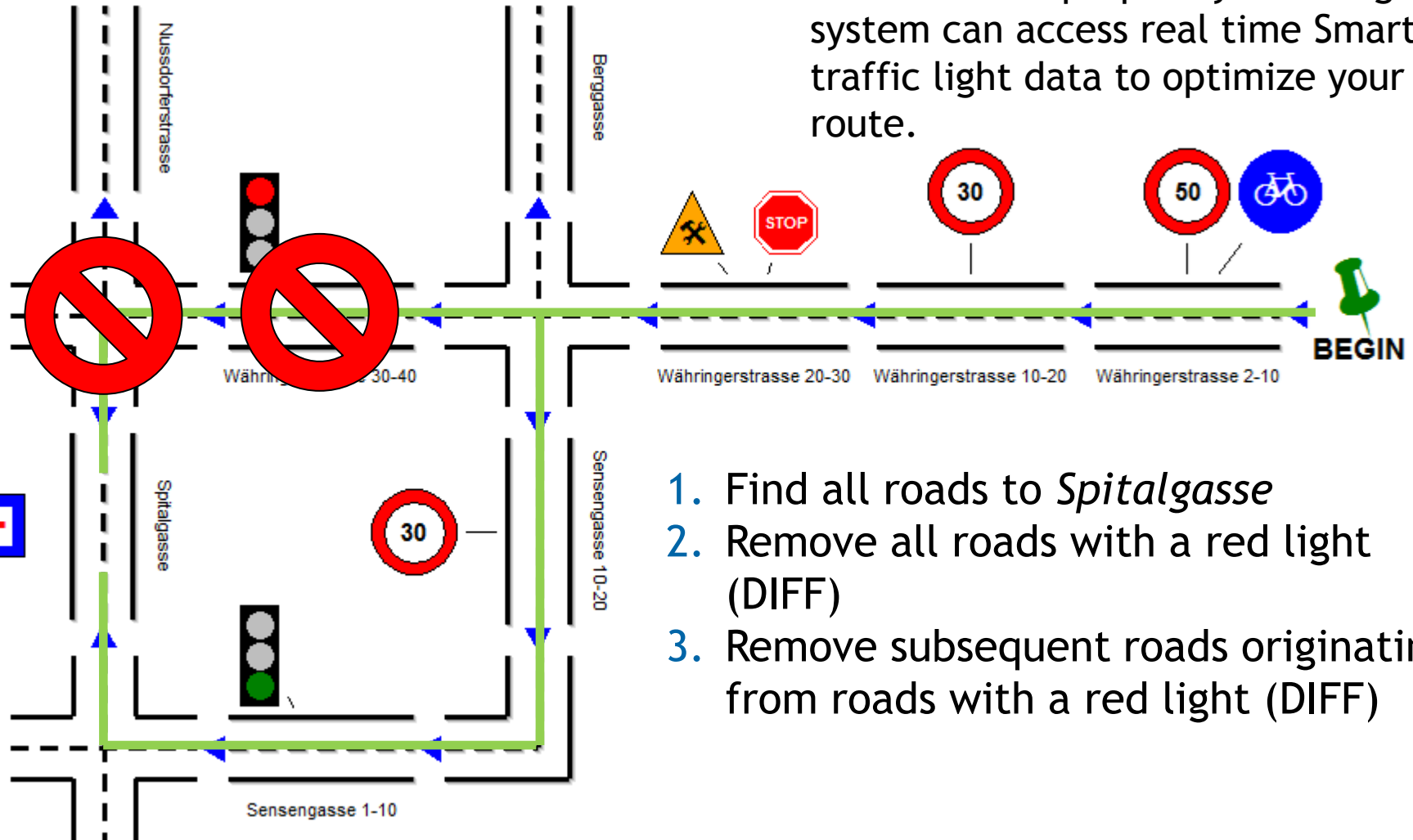
Hands-On 3: Emergency Service in a Smart City (3/5)

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.



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

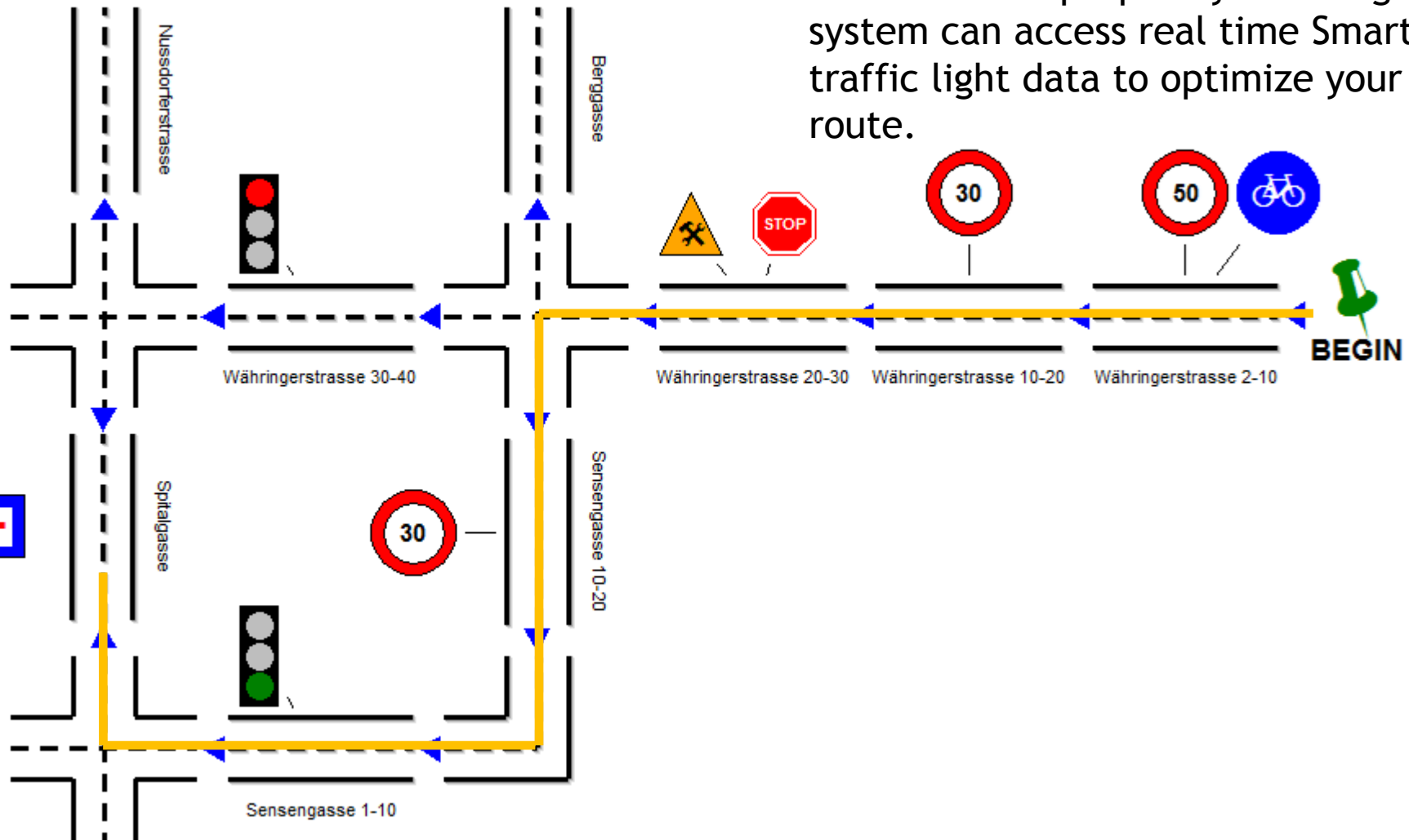
You transport an injured person in an ambulance. The condition is not critical, but you want to reduce waiting time. For this purpose your navigation system can access real time Smart City traffic light data to optimize your route.



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
 - Find all roads to *Spitalgasse*
 - Remove all roads with a red light (DIFF)
 - Remove subsequent roads originating from roads with a red light (DIFF)

4 Execute

5

Query results - (<"Road">[?"Name" = "Spitalgasse"]

	Name
1. 4_SmartAmbulance	
CrossRoad-12628	
CrossRoad-14570	
Sensengasse 1-10	Sensengasse 1-10
Sensengasse 10-10	Sensengasse 10-10
Sensengasse 10-20	Sensengasse 10-20
Währingerstrasse 10-20	Währingerstrasse
Währingerstrasse 2-10	Währingerstrasse
Währingerstrasse 20-30	Währingerstrasse

User defined queries

```
1 (<"Road">[?"Name" = "Spitalgasse"]<-"Subsequent") DIFF
2 (<"Traffic_Lights">[?"Color" = "Red"]->"SmartCity_Relation") DIFF
3 (<"Traffic_Lights">[?"Color" = "Red"]->"SmartCity_Relation"->>"Subsequent")
```

AND OR DIFF Clear

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

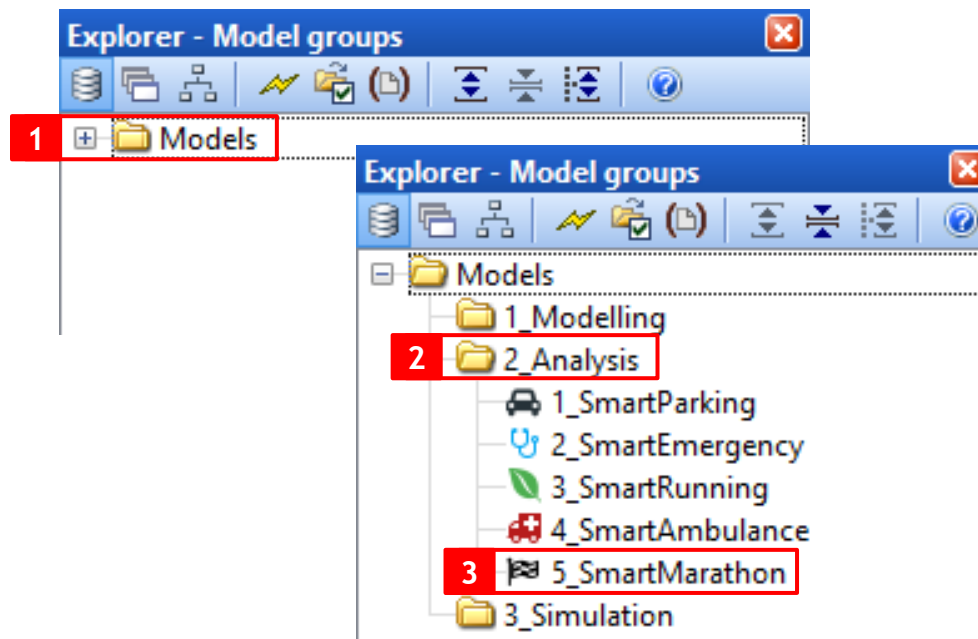
- Basics: Relations and relationship operators, AQL
- Demo: AQL, transitive relations | Smart Traffic Planning
- Hands-on: AQL, transitive relations | Smart Ambulance Navigation

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

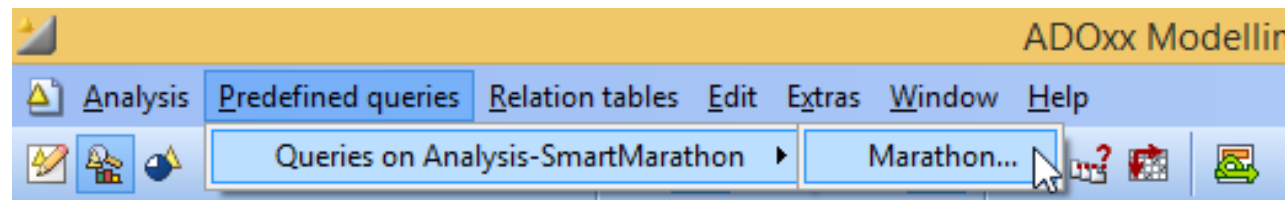
Predefined analysis queries

- 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

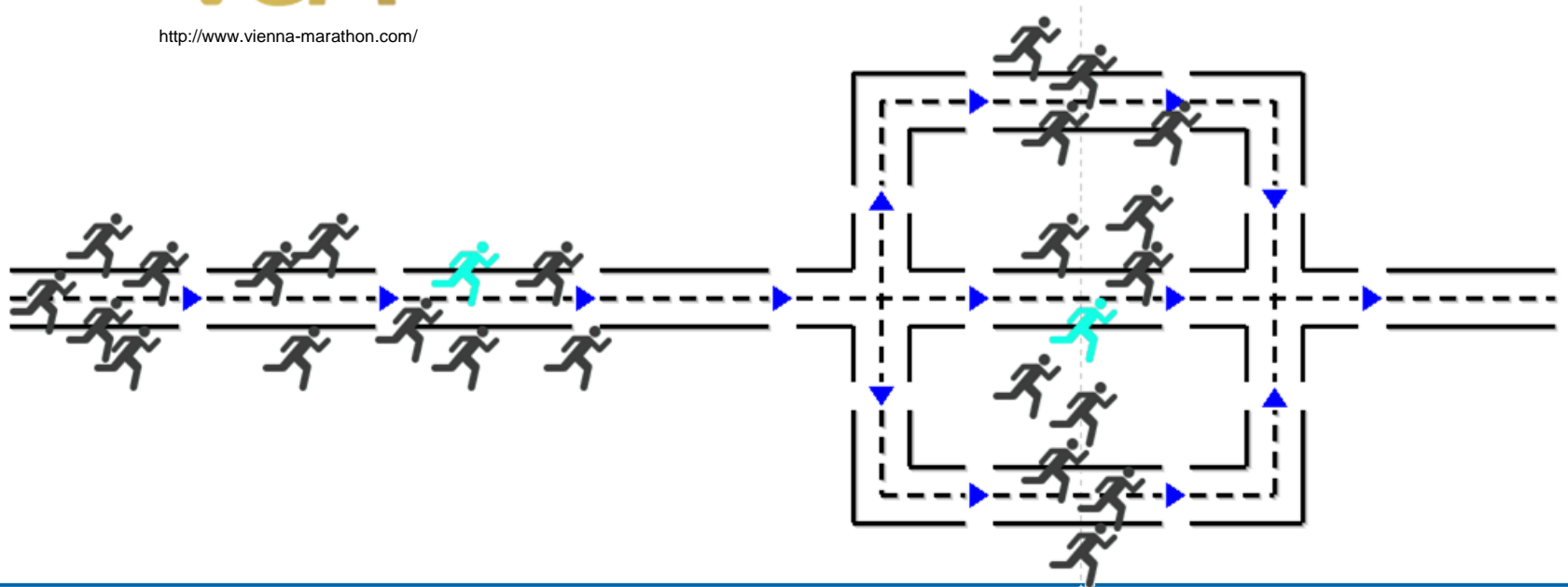
- Relay marathon hand overs are an uncomfortable situation
 - Lots of people waiting
 - No fixed time
 - Runners are exhausted
- Goal: A tool for the waiting person that can track your currently running teammate
 - Once you found your partner, you can pass along the tool (e.g., on a smartphone or smartwatch) 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/>

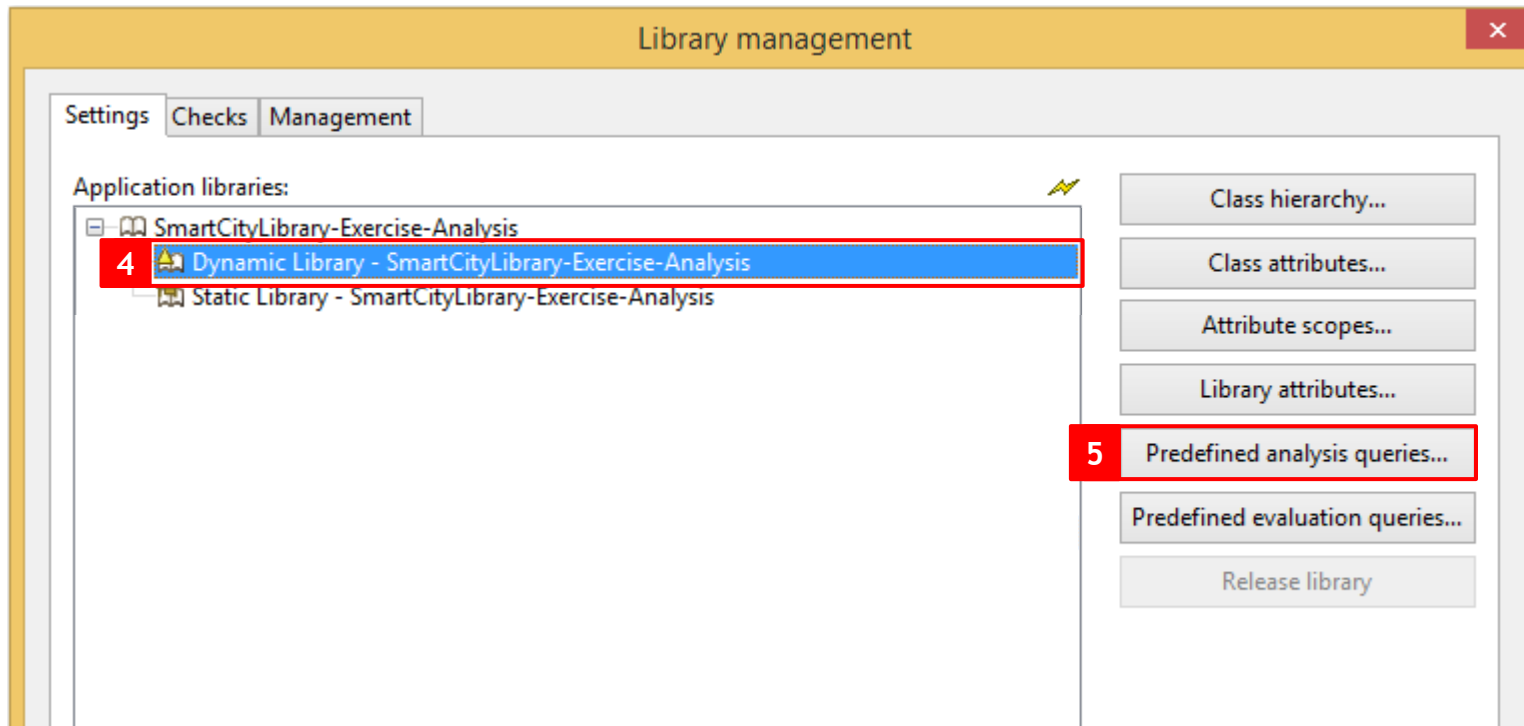
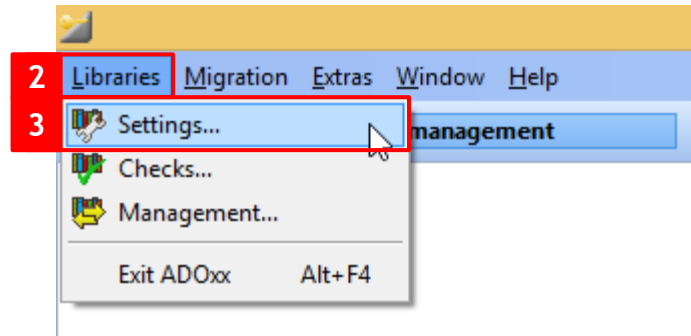
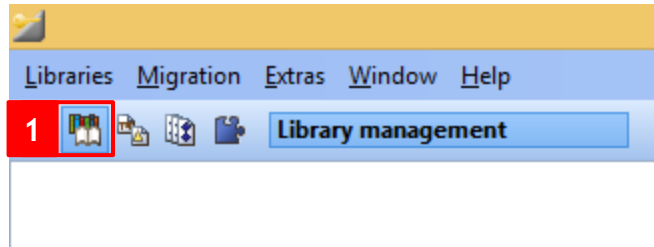


64

Demo 4: Solution

- **Design** the query in the **Development Toolkit**
 - Create new menu entry
 - Create input fields
 - Customize AQL expressions
 - Parameterize 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

The screenshot illustrates the process of adding a new menu item in the 'Dynamic Library - SmartCityLibrary-Exercise-Analysis - Edit queries' window. The steps are numbered 1 through 12:

1. The 'Menu items' tab is selected.
2. The 'New...' button is highlighted.
3. The 'Insert new element' dialog is open, showing 'Menu item' as the new element.
4. The 'Position to insert' section shows 'As last element' selected.
5. The 'OK' button in the 'Insert new element' dialog is highlighted.
6. The 'Edit text field' dialog is open, showing 'Menu item' as the text.
7. The 'Marathon' text is entered into the 'Menu item' field.
8. The 'Model type' dropdown is open, showing 'Analysis-SmartMarathon' selected.
9. The 'OK' button in the 'Edit text field' dialog is highlighted.
10. The 'Edit text field' dialog is open, showing 'Query' as the text.
11. The 'Relay' text is entered into the 'Query' field.
12. The 'OK' button in the 'Edit text field' dialog is highlighted.

67

Demo 4: Solution

The image shows a sequence of five screenshots from a software application titled "Dynamic Library - SmartCityLibrary-Exercise-Analysis - Edit queries".

- Screenshot 13:** The "Menu items" tab is selected. A tree view shows a folder "Marathon" containing a sub-item "1. Relay". The "1. Relay" item is highlighted with a red box and the number 13.
- Screenshot 14:** The "Input fields" tab is selected. The "1. Query: Relay" section shows an "Elements:" area. A context menu is open over this area, with the "New..." option highlighted by a red box and the number 15.
- Screenshot 16:** The "Insert new element" dialog is open. The "New element:" dropdown menu is set to "Text", highlighted with a red box and the number 16.
- Screenshot 17:** The "Insert new element" dialog is shown with the "OK" button highlighted by a red box and the number 17.
- Screenshot 18:** The "Edit text field" dialog is open. The "Text field:" input contains the text "Name", highlighted with a red box and the number 18.
- Screenshot 19:** The "Edit text field" dialog is shown with the "OK" button highlighted by a red box and the number 19.

68

Demo 4: Solution

The screenshot illustrates the process of adding a new input field in the 'Dynamic Library - SmartCityLibrary-Exercise-Analysis - Edit queries' window. The window has three tabs: 'Menu items', 'Input fields', and 'AQL expressions'. The 'Input fields' tab is active, showing a tree structure under '1. Query: Relay' with an 'Elements' list containing 'Text' and 'Name'. A context menu is open over the 'Text' element, with the 'New...' option highlighted. A red box labeled '21' is around the 'New...' option. A red box labeled '20' is around the 'Input fields' tab. A red box labeled '22' is around the 'Edit field' option in the 'Insert new element' dialog. A red box labeled '23' is around the 'OK' button in the 'Insert new element' dialog. A red box labeled '24' is around the 'Field length' input field in the 'Edit field' dialog, which has the value '60'. A red box labeled '25' is around the 'OK' button in the 'Edit field' dialog. The 'Edit field' dialog also shows 'Input type' set to 'String' and 'Field length' set to '60'.

Demo 4: Solution

The screenshot shows the 'Dynamic Library - SmartCityLibrary-Exercise-Analysis - Edit queries' window. The 'AQL expressions' tab is active. A context menu is open over the 'AQL elements' list, with 'New...' selected (27). The 'Insert new element' dialog is open, with 'AQL part' selected in the 'New element:' dropdown (28). The 'Edit query part' dialog is open, showing the 'Standardised queries' list with 'Get all objects of class ... with attribute ...' selected (30). The 'Input field' section shows 'Get all objects of class' with a dropdown set to 'Runner', 'with attribute' with a dropdown set to 'Name', and a dropdown set to 'like' followed by a text field containing 'dummy' (31). The 'OK' button in the 'Edit query part' dialog is highlighted (31). The 'OK' button in the 'Insert new element' dialog is also highlighted (29).

Dynamic Library - SmartCityLibrary-Exercise-Analysis - Edit queries

Menu items Input f 26 AQL expressions Result attributes

1. Query: Relay

AQL elements:

AQL expression:

27 New...

28 AQL part

29 OK

30 Query: Get all objects of class ... with attribute ...

31 OK

Input field

Get all objects of class Runner

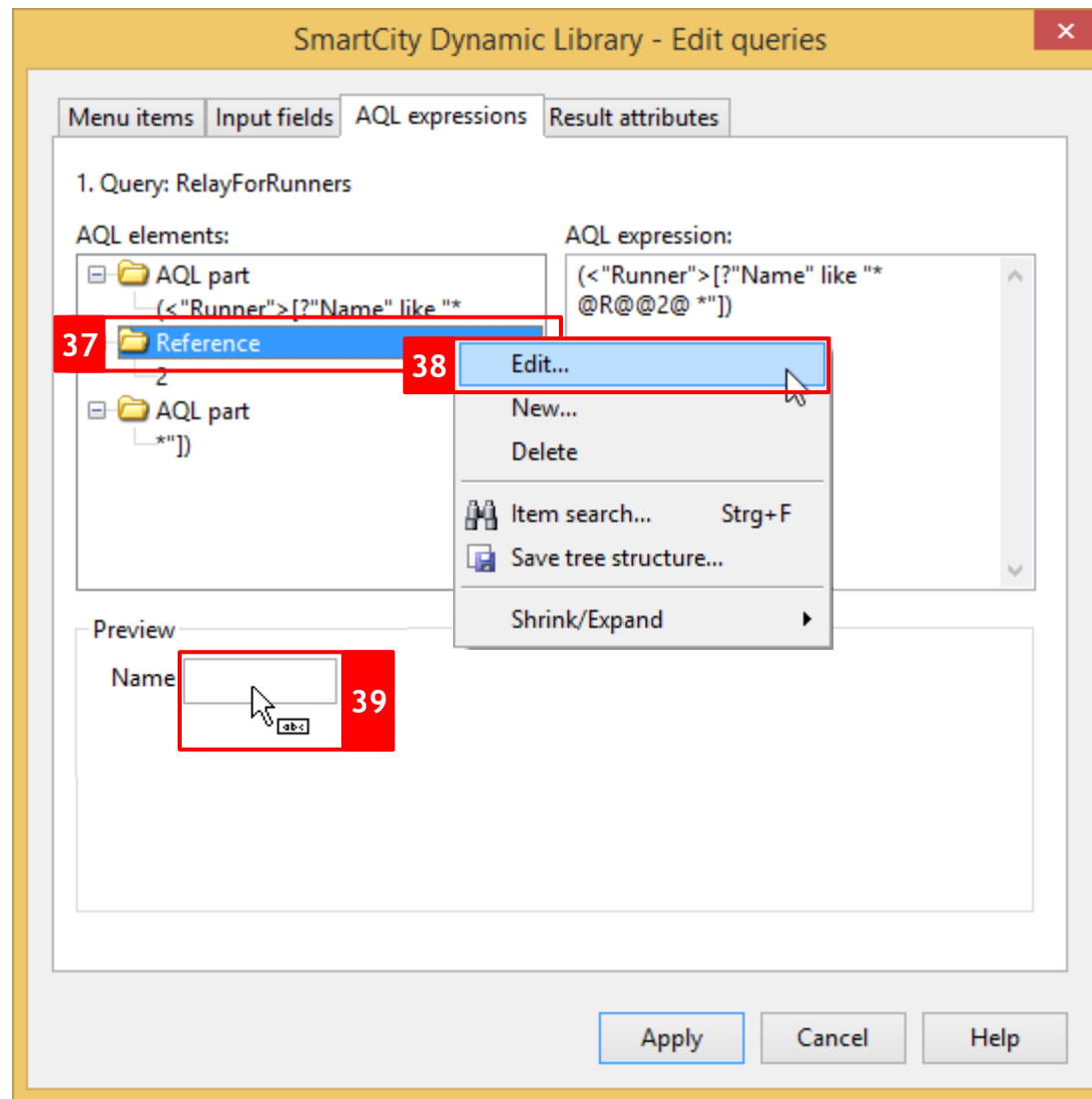
with attribute Name like dummy

OK Cancel Help 70

Demo 4: Solution

The screenshot displays the 'Dynamic Library - SmartCityLibrary-Exercise-Analysis - Edit queries' window. The 'AQL expressions' tab is active, showing a query named '1. Query: Relay' with the expression `(<"Runner">[?"Name" like "dummy"])`. A context menu is open over the AQL elements, with the 'New...' option highlighted (32). The 'Insert new element' dialog is open, showing 'Reference' as the new element (33). The 'Position to insert' section has 'As last element' selected. The 'AQL - Create reference' dialog is open, showing the expression `(<"Runner">[?"Name" like "dummy"])` with a cursor at the end (35). The 'Create' button is highlighted (36). The 'OK' button in the 'Insert new element' dialog is highlighted (34).

Demo 4: Solution



Demo 4: Solution

The screenshot displays the 'Dynamic Library - SmartCityLibrary-Exercise-Analysis' window. The 'Menu items' tab is active, and the 'Result attributes' section is highlighted with a red box and the number 40. A red box with the number 41 highlights the 'New...' button in the 'Classes' section. A red box with the number 42 highlights the 'Attribute' option in the 'Insert new element' dialog. A red box with the number 43 highlights the 'OK' button in the same dialog. A red box with the number 44 highlights the 'Attribute selection' dialog, which shows a list of attributes. A red box with the number 45 highlights the 'OK' button in the 'Attribute selection' dialog. A red box with the number 46 highlights the 'Apply' button at the bottom of the main window.

Dynamic Library - SmartCityLibrary-Exercise-Analysis

Menu items Input fields AQL express 40 Result attributes

1. Query: PartnerFinder

Classes:

Edit... 41 New... Delete

Item search... Strg+F

Save tree structure...

Shrink/Expand

Relations:

Insert new element 44

New element: 42 Attribute

Position to insert

☐ Before the selected element

☐ After the selected element

☒ As last element

43 OK Cancel

Attribute selection 45

Attribute:

- SmartCity
- Simulation-Smart Marathon
- SmartCityAnalysis
- Analysis-SmartParking
- Analysis-SmartEmergency
- Analysis-SmartTrack
- Analysis-SmartAmbulance
- Analysis-SmartMarathon
 - Traffic_Signs
 - Traffic_Lights
 - Buildings
 - Road
 - CrossRoad
 - Car
 - Heliport
 - Logo
 - RelaySection
 - Traffic
 - Runner
 - Description
 - Name
 - Color
 - Number
 - Km
 - RelaySection
 - AssignedRelaySection
- Simulation-Smart Marathon Drink S
- Simulation-Smart Marathon Static I

46 Apply Cancel Help

Demo 4: Solution

The screenshot illustrates the steps to execute a query in the OMLAB software. The steps are numbered 47 through 54:

- 47: Click on the 'Queries on SmartCity' menu item.
- 48: Click on the 'Marathon...' sub-menu item.
- 49: Select the '5_SmartMarathon' model in the 'Model selection' dialog.
- 50: Click the 'OK' button in the 'Model selection' dialog.
- 51: Select the 'Relay' query in the 'Marathon - Queries' dialog.
- 52: Enter 'Max Weber' in the 'Name' field of the 'Query specification' section.
- 53: Click the 'Execute' button.
- 54: View the 'Query results - RelayForOrganizers' table.

The 'Query results - RelayForOrganizers' table displays the following data:

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

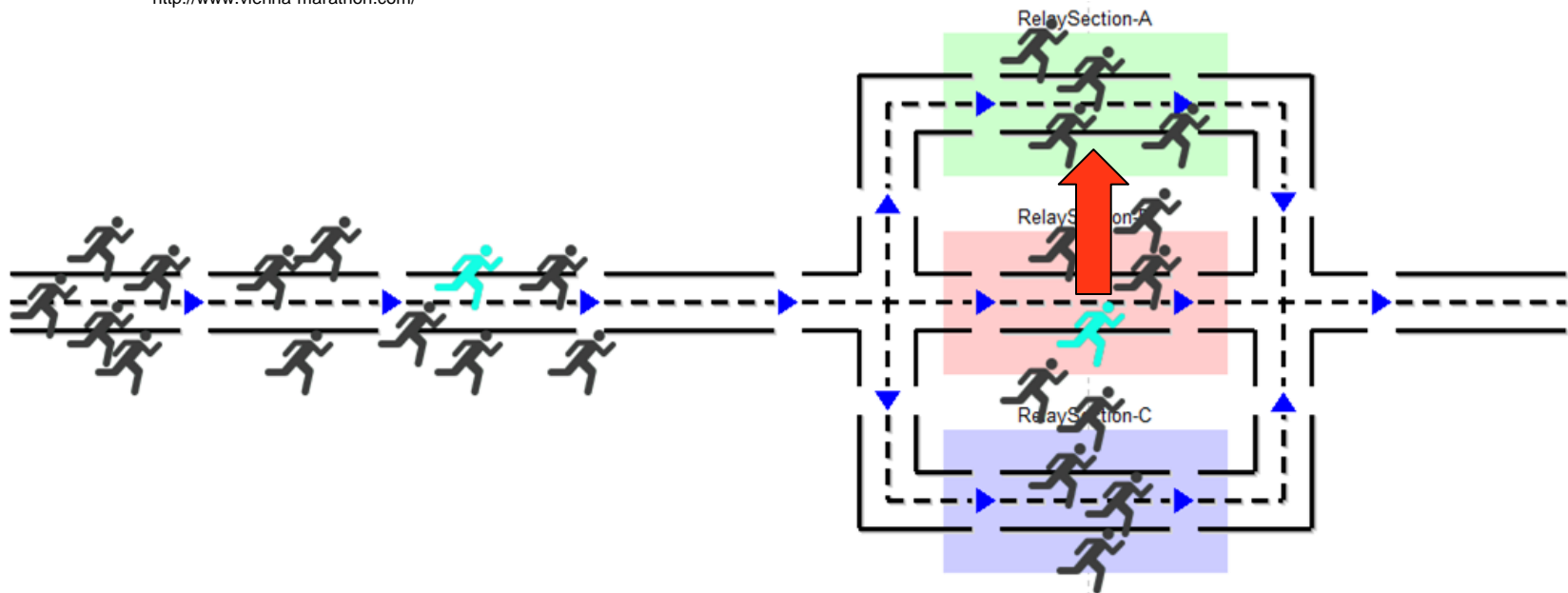
74

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



75

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

```
((({ "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" } <-  
  "Is inside") OR ({ "RelaySection-C": "RelaySection" } -> "Is inside"))
```

AND

```
(<"Runner">[?"AssignedRelaySection" !=  
  "C"]]))
```

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:

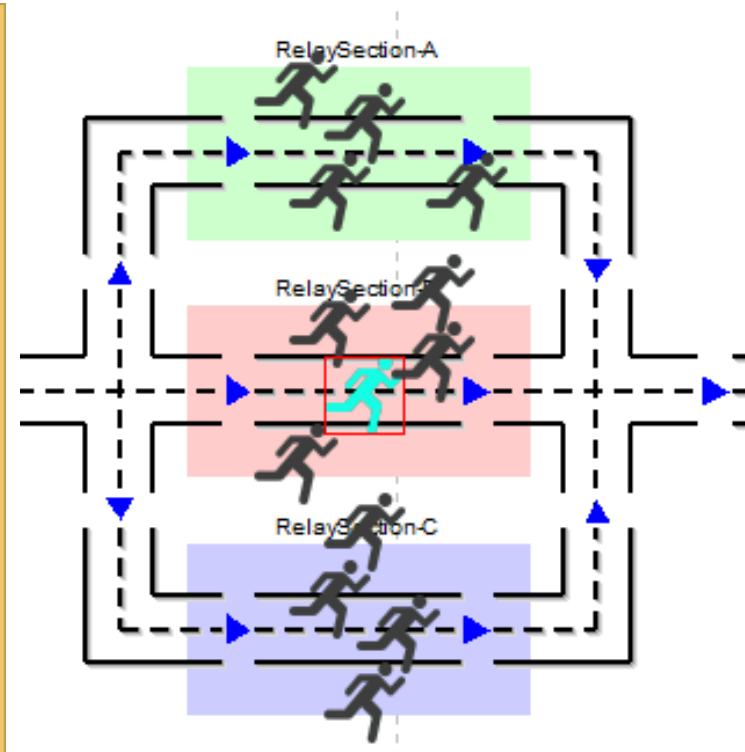
```
((("RelaySection-A":"RelaySection")<- "Is inside") OR ((("RelaySection-A":"RelaySection")-> "Is inside")) AND (<"Runner">["AssignedRelaySection" != "A"])) OR (((("RelaySection-B":"RelaySection")<- "Is inside") OR ((("RelaySection-B":"RelaySection")-> "Is inside")) AND (<"Runner">["AssignedRelaySection"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

Preview

Query results - RelayForOrganizers

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

Apply Cancel Help



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

Thank you.



Dominik Bork



Nikolaos Tantouris



Niksa Visic



Michael Walch



Simon Doppler



Franz Staffel