Iceland Liechtenstein Norway grants

UNIVERSITATEA LUCIAN BLAGA DIN SIBIU

University of South-Eastern Norway

MAPIoT

Assembly lines for picking fruits/vegetables

Daniel MORARIU

Ion MIRONESCU

July 25 – August 07, 2022 - Melsom High School, Sandefjord, Norway

Contents:

✓ **Modeling and simulation on the ADOxx platform**:

- ➢ Short introduction in meta-modeling languages and ADOxx .
- ➢ Bee-Up tool for modeling.
- ➢ Modeling CPS components
- ➢ Controlling CPS components

✓ **Interfacing with cyber-physical systems (CPS)**

- ➢ Architecture of the combined systems (Bee-Up + CPS)
- ➢ ADOxx/Bee-Up interfaces with cyber-physical systems
- ➢ AdoScript commands and feedback
- ➢ Developing a command-and-control model in Bee-up

✓ **Developing practical application**

- ➢ Developing an application for the robotic arm (DoDot).
- ➢ Developing an application for the mobile robot (mBot).

Why modeling?

https://automatorobotics.com/

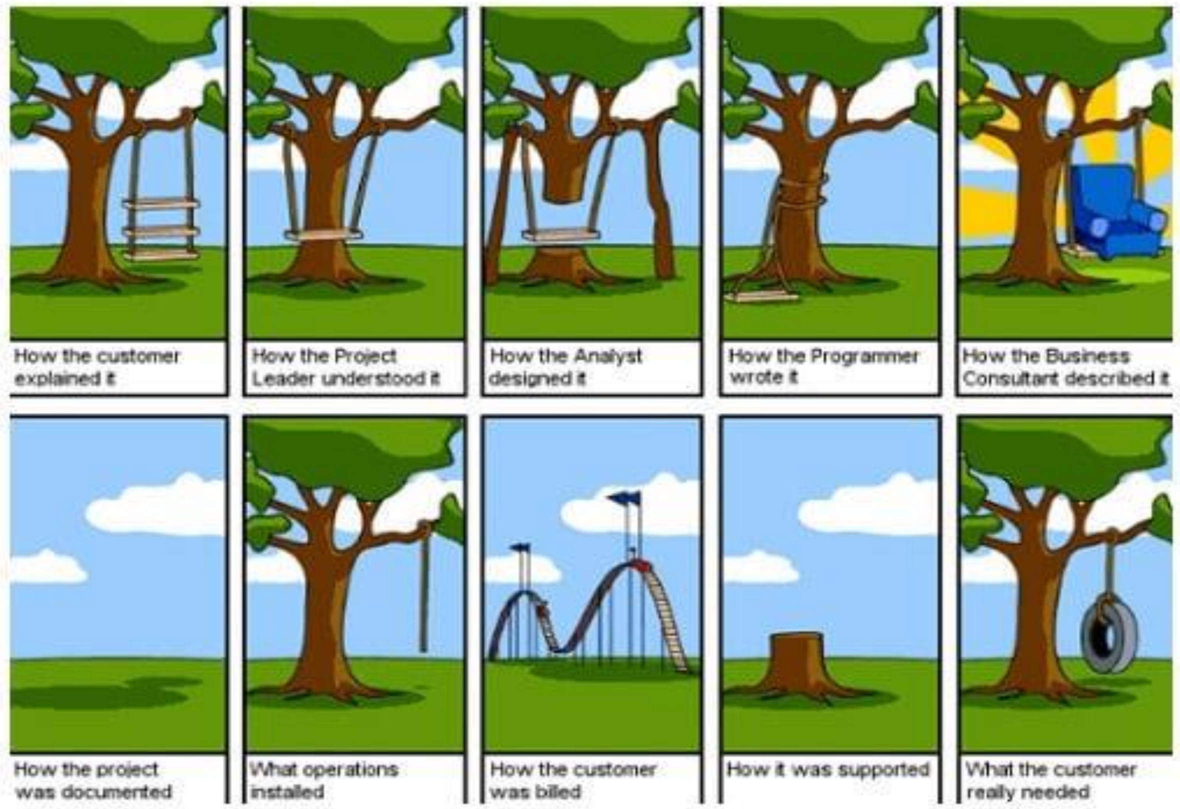
ADOxx as conceptual modeling language

- ✓ ADOxx is the meta-modelling development and configuration platform for implementing modelling methods.
  - ✓ Why modeling?
    - ✓ If the object you want to create or change is simple, then you can do it directly.
    - ✓ For complex systems that are likely to change over time, you need a model.
    - ✓ “Without explicit modelling there is a high risk that the implementation is not what is intended.” (John Zachmann, 2012)
  - ✓ Conceptual modelling
    - ✓ the modelling language that describes the syntax, semantics and notation.
    - ✓ the modelling procedure that describe how to create valid models.
    - ✓ algorithms and mechanisms that provide “functionality to use and evaluate” models described by a modelling language.

ADOxx as conceptual modeling language

✓ „Conceptual modelling is the activity of formally describing some aspects of the physical and social world around us for the purposes of understanding and communication.“

Mylopoulos (1992) Conceptual modeling and Telos1


How the customer explained it
How the Project Leader understood it
How the Analyst designed it
How the Programmer wrote it
How the Business Consultant described it
How the project was documented
What operations installed
How the customer was billed
How it was supported
What the customer really needed

Conceptual Modelling Purpose - Dimensions

I. **Analysis: Says what is.**

The model provides a

- description of the *phenomena of interest*,
- analysis of *relationships* among those constructs,
- the *degree of generalizability* in constructs and relationships
- the *boundaries* within which relationships, and observations hold.

II. **Explanation: Says what is, how, why, when, and where.**

The model provides *an explanation of how, why, and when things happened*, relying on varying views of causality methods for argumentation. This explanation usually promotes greater understanding or insights by others into the phenomena of interest.

III. **Prediction: Says what is and what will be**.

The model states *what will happen in the future* if certain preconditions hold. The degree of certainty in the prediction is expected to be only approximate or probabilistic.

IV. **Explanation and prediction: Says what is, how, why, when, where, and what will be**.

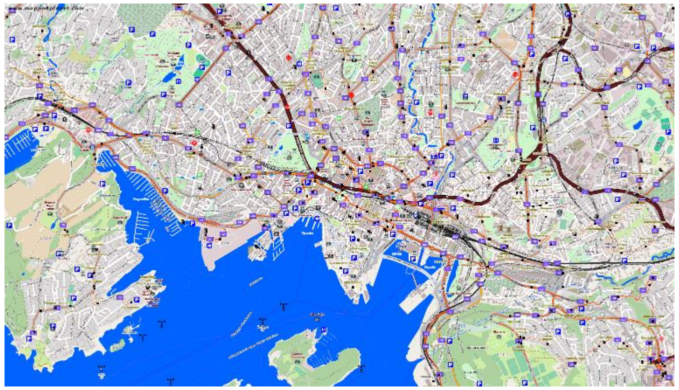
The model provides predictions and has both testable propositions and causal explanations.

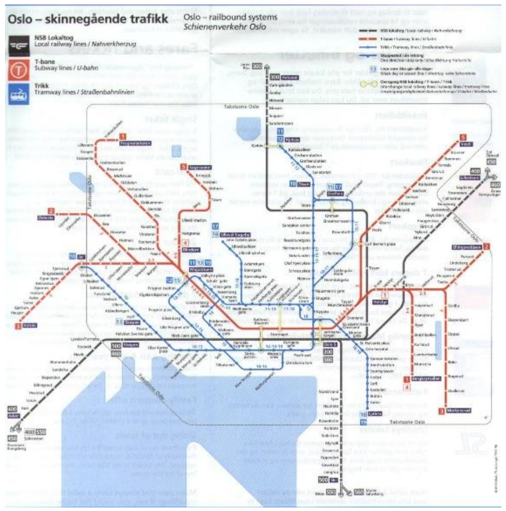
V. **Design and action: Says how to do something.**

The model gives explicit prescriptions (e.g., methods, techniques, principles of form and function) for constructing an artifact.

Abstraction is the key to Modelling

- ✓ Simplification
  - ✓ Withdrawing or removing something
  - ✓ Leaving out of consideration one or more
- ✓ Generalization
  - ✓ Formulating general concepts by abstracting common properties of instances
  - ✓ A general concept formed by extracting




Oslo – skinnegående trafikk
Oslo – railbound systems
Schienenverkehr Oslo
NSB Lokaltog
Local railway lines / Nahverkehrszug
T-bane
Subway lines / U-bahn
Trikk
Tramway lines / Straßenbahnlinien

What is the Bee-Up tool for modeling

Hybridizes several commonly used modelling languages in one prototypical implementation.

BPMN – Business Process Model and Notation

EPC – Event-driven Process Chains

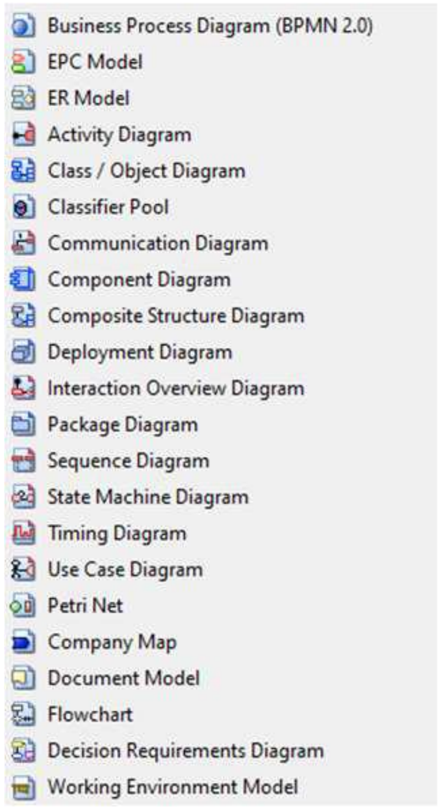
ER – Entity Relationship

-

UML – Unified Modeling Language

Petri Nets

Actual Model Types:

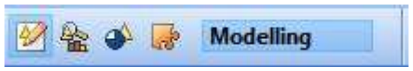


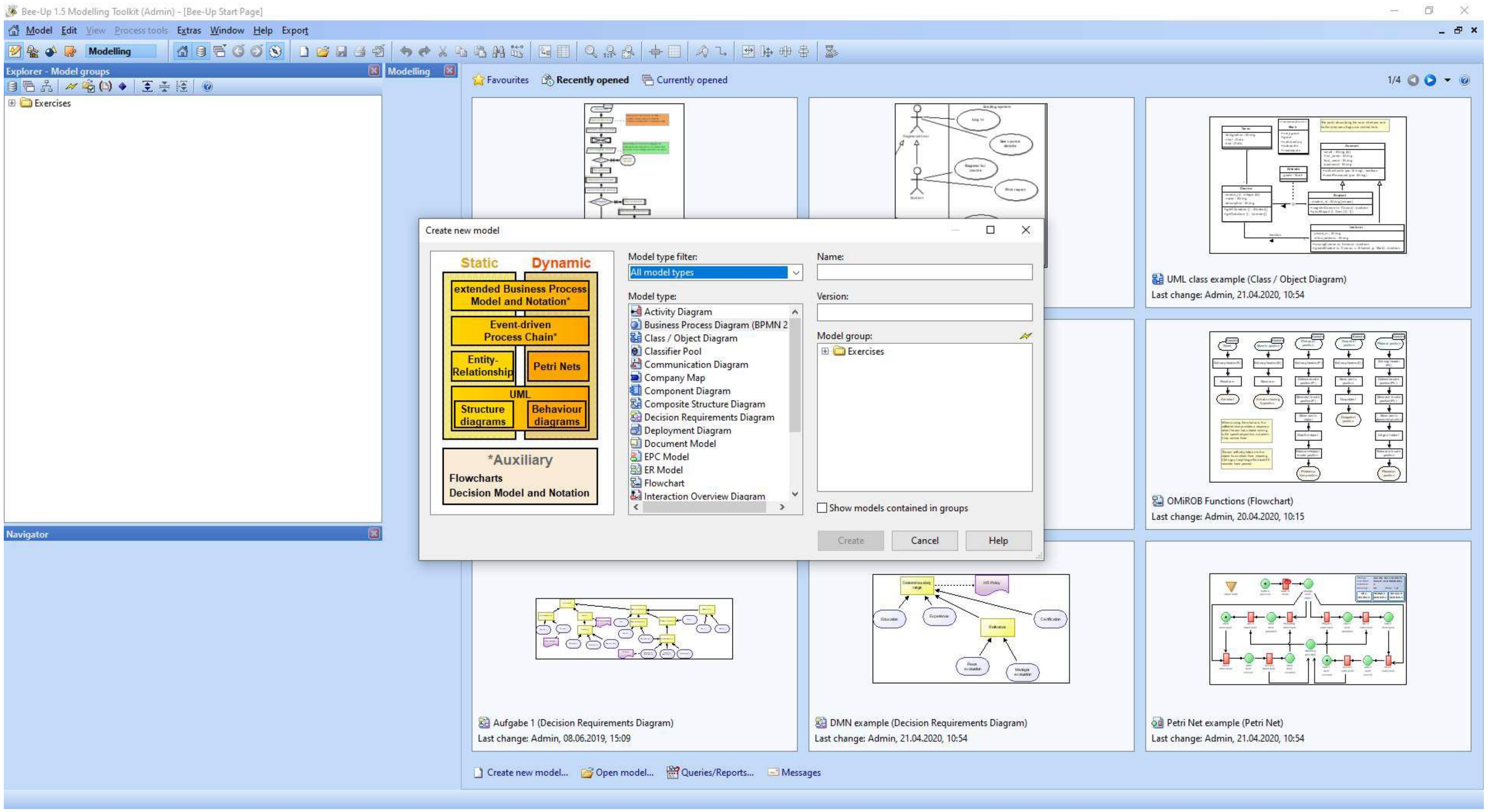
- Business Process Diagram (BPMN 2.0)
- EPC Model
- ER Model
- Activity Diagram
- Class / Object Diagram
- Classifier Pool
- Communication Diagram
- Component Diagram
- Composite Structure Diagram
- Deployment Diagram
- Interaction Overview Diagram
- Package Diagram
- Sequence Diagram
- State Machine Diagram
- Timing Diagram
- Use Case Diagram
- Petri Net
- Company Map
- Document Model
- Flowchart
- Decision Requirements Diagram
- Working Environment Model

Flowchart in Bee-Up

- ✓ Represents a workflow or a process.
- ✓ Can be defined as a diagrammatic representation of an algorithm, a step-by-step approach to solving a task.
- ✓ UML activity diagrams and Drakon-charts can be extensions of the flowchart.
- ✓ Flowchart Types
  - ✓ Document flowcharts, showing controls over a document-flow through a system.
  - ✓ Data flowcharts, showing controls over a data-flow in a system.
  - ✓ System flowcharts, showing controls at a physical or resource level.
  - ✓ Program flowchart, showing the controls in a program within a system.

Bee-Up Overview





Bee-Up 1.5 Modelling Toolkit (Admin) - [Bee-Up Start Page]

Model Edit View Process tools Extras Window Help Export

Explorer - Model groups

Exercises

Navigator

Modelling

Favourites Recently opened Currently opened

1/4

Create new model

Model type filter: All model types

Model type:
- Activity Diagram
- Business Process Diagram (BPMN 2
- Class / Object Diagram
- Classifier Pool
- Communication Diagram
- Company Map
- Component Diagram
- Composite Structure Diagram
- Decision Requirements Diagram
- Deployment Diagram
- Document Model
- EPC Model
- ER Model
- Flowchart
- Interaction Overview Diagram

Name:

Version:

Model group:

Exercises

Show models contained in groups

Create Cancel Help

UML class example (Class / Object Diagram)
Last change: Admin, 21.04.2020, 10:54

OMiROB Functions (Flowchart)
Last change: Admin, 20.04.2020, 10:15

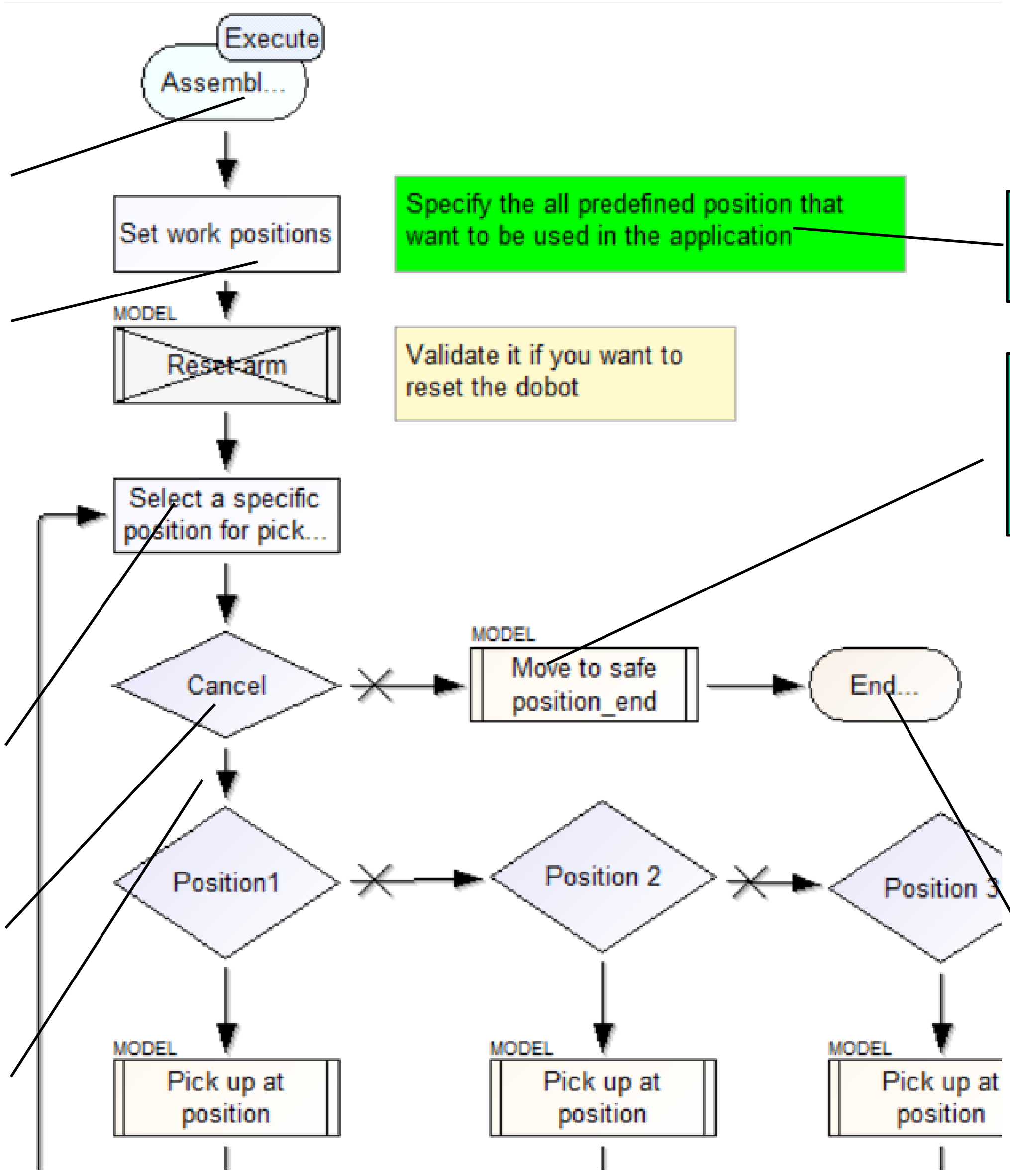
Aufgabe 1 (Decision Requirements Diagram)
Last change: Admin, 08.06.2019, 15:09

DMN example (Decision Requirements Diagram)
Last change: Admin, 21.04.2020, 10:54

Petri Net example (Petri Net)
Last change: Admin, 21.04.2020, 10:54

Create new model... Open model... Queries/Reports... Messages

Flowchart


Execute
Assembl...
Set work positions
Specify the all predefined position that want to be used in the application
MODEL
Reset arm
Validate it if you want to reset the dobot
Select a specific position for pick...
Cancel
MODEL
Move to safe position_end
End...
Position1
Position 2
Position 3
MODEL
Pick up at position
MODEL
Pick up at position
MODEL
Pick up at position

Start Terminal

Activity

Input/ Output

Decision

Flowline

Comment

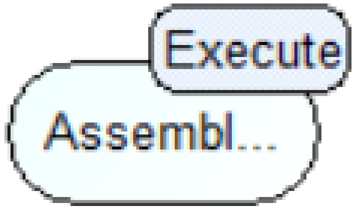
Predefined Function

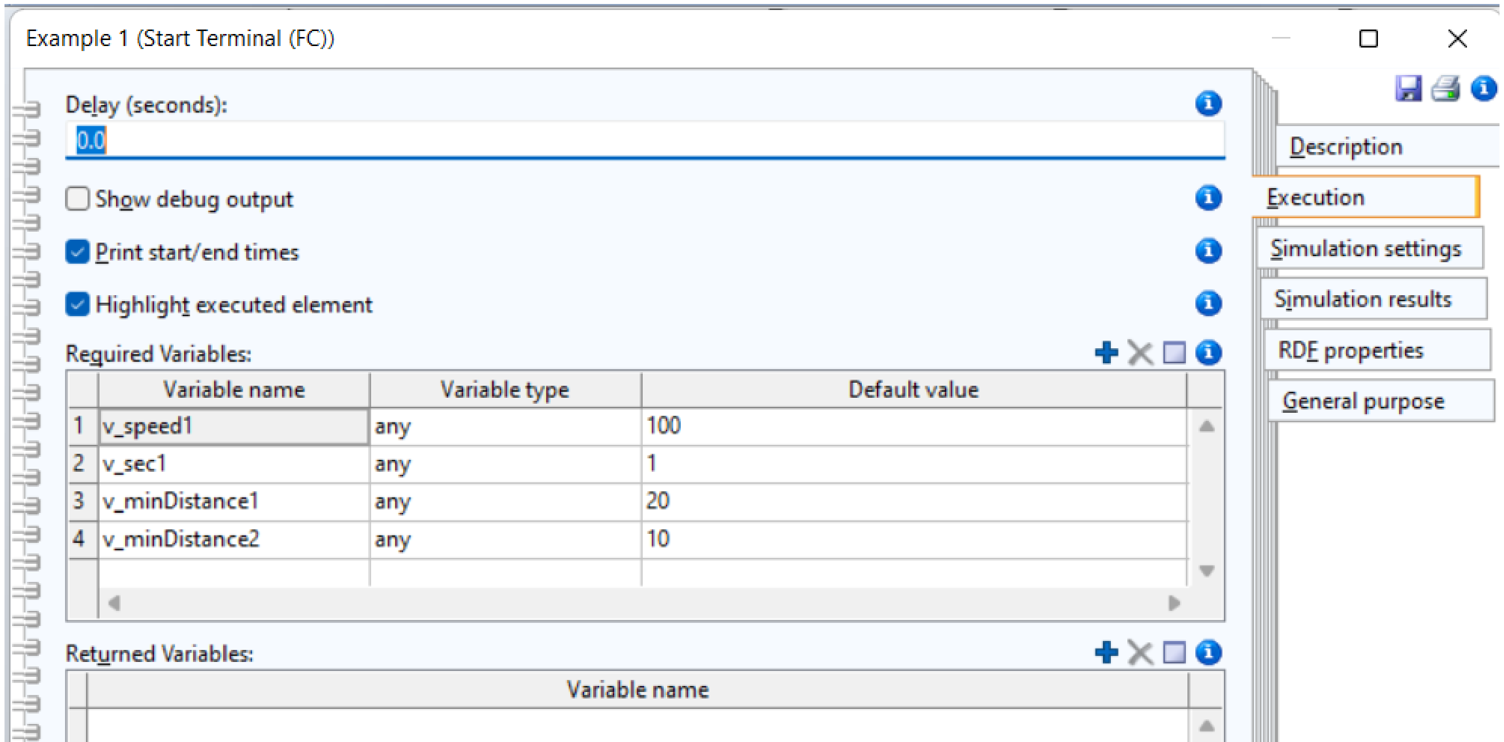
End Terminal

Start terminal block

Allows us to specify a start of a program or a function

- Description tab
  - Name
- Execution tab
  - Required variables
  - Returned variables





Delay (seconds): 0.0

[ ] Show debug output

[✓] Print start/end times

[✓] Highlight executed element

Required Variables:

	Variable name	Variable type	Default value
1	v_speed1	any	100
2	v_sec1	any	1
3	v_minDistance1	any	20
4	v_minDistance2	any	10

Returned Variables:

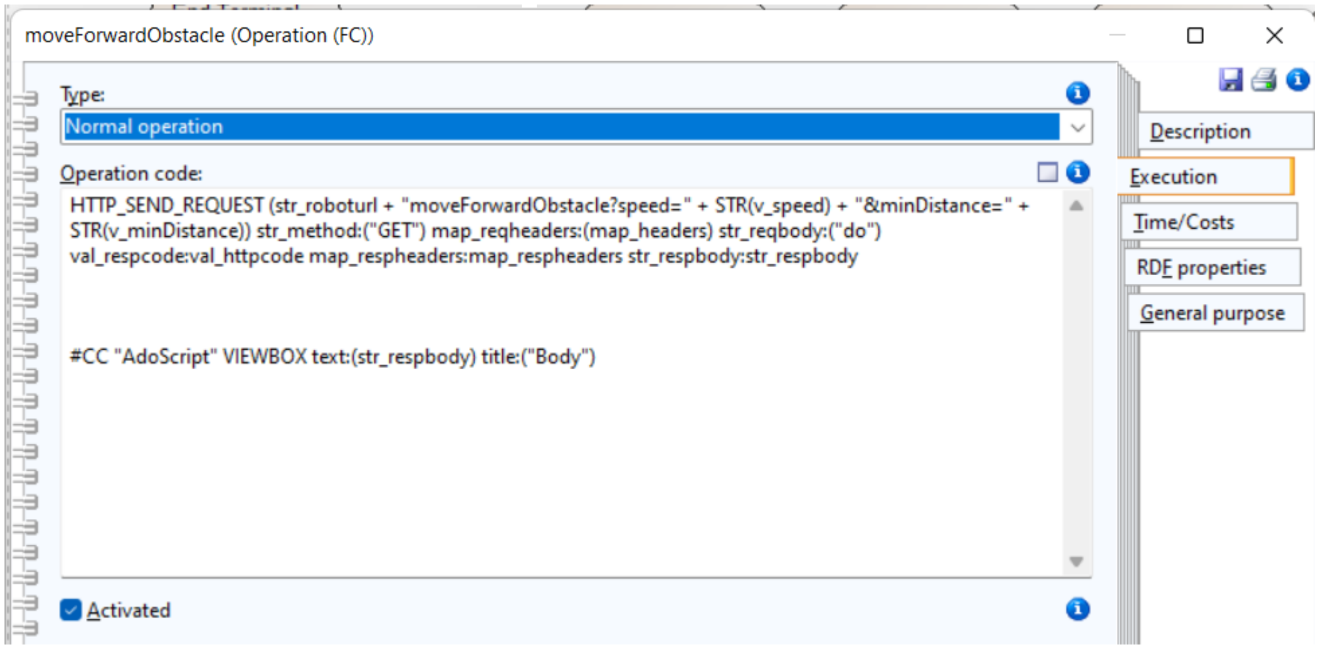
Variable name

Activity / Operation block

Select a specific position for pick...

Allows us to specify a command for the robot

- Description
  - Name
- Execution
  - Activated
  - Operation code



Type: Normal operation

Operation code:

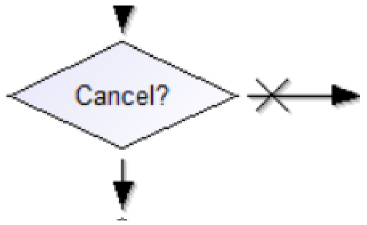
```
HTTP_SEND_REQUEST (str_roboturl + "moveForwardObstacle?speed=" + STR(v_speed) + "&minDistance=" +
STR(v_minDistance)) str_method:("GET") map_reqheaders:(map_headers) str_reqbody:("do")
val_respcode:val_httpcode map_respheaders:map_respheaders str_respbody:str_respbody



#CC "AdoScript" VIEWBOX text:(str_respbody) title:("Body")
```

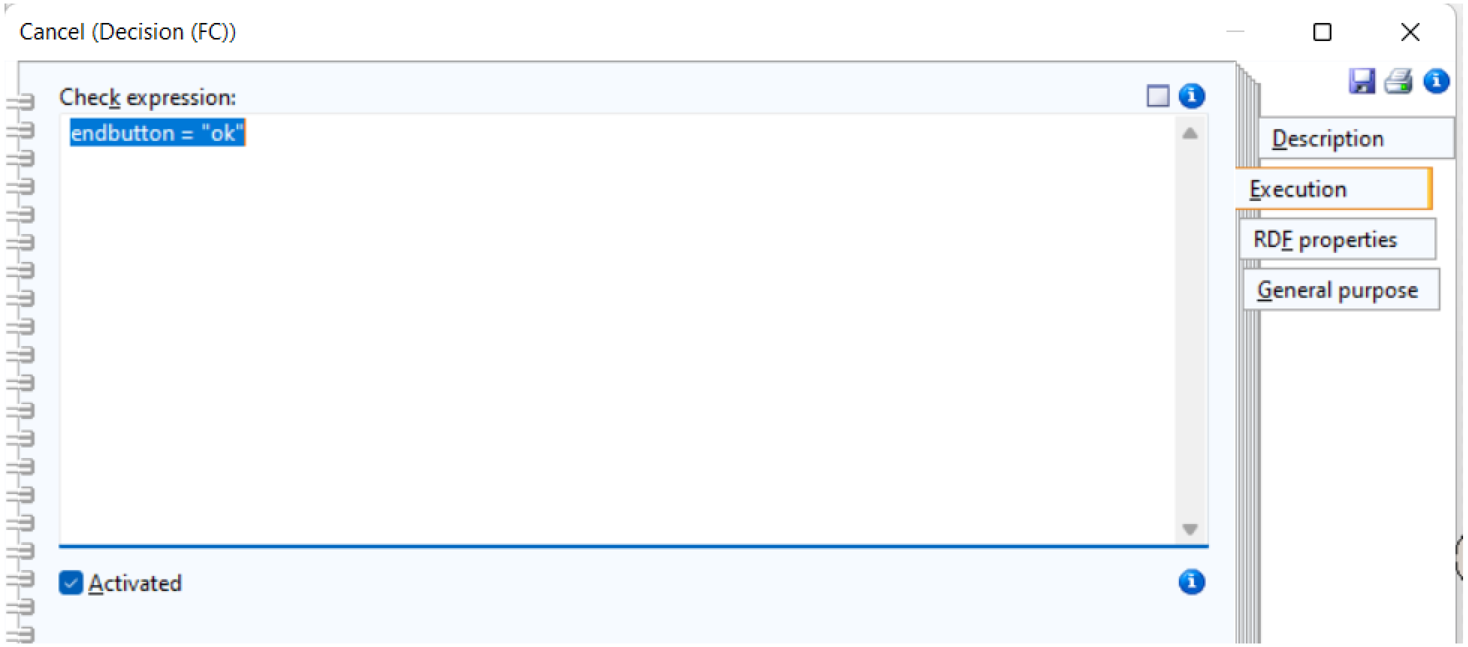
[✓] Activated

Decision block

Allows us to test a condition and follow two different directions

- Description
  - Name
- Execution
  - Check expression




Cancel (Decision (FC))

Check expression:

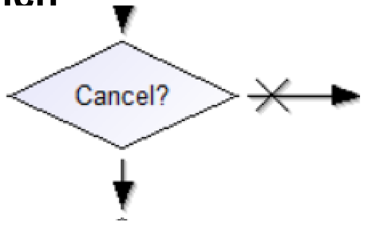
```
endbutton = "ok"
```

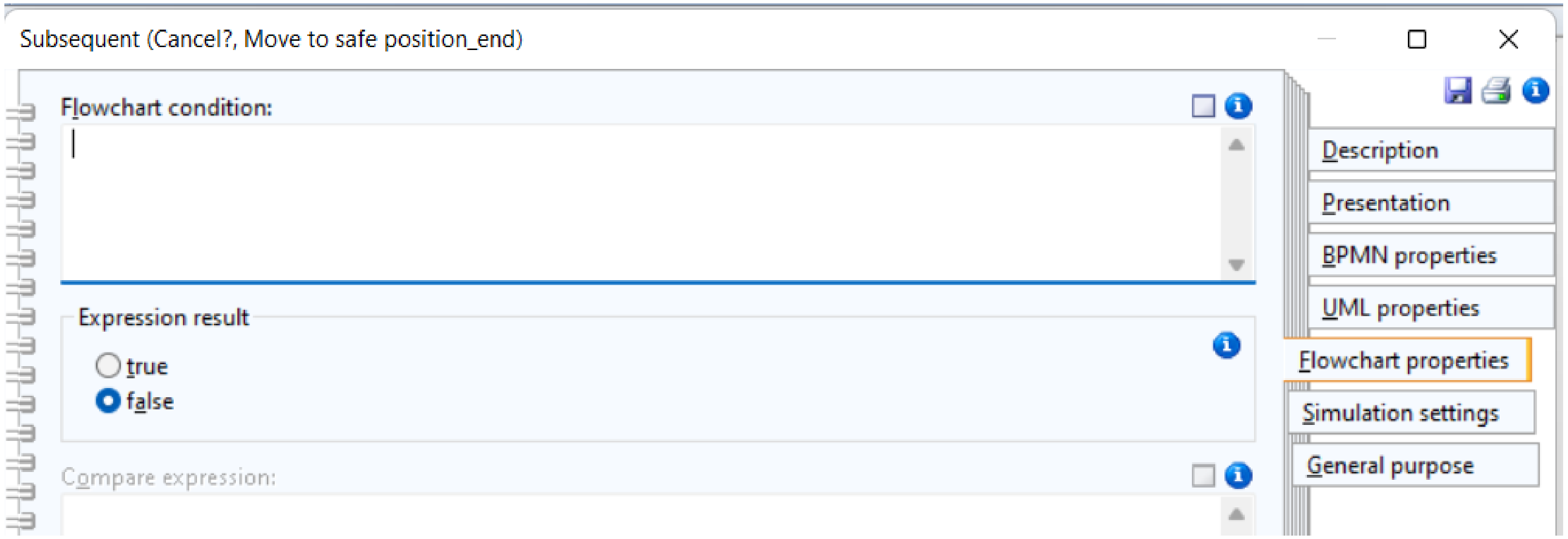
✓ Activated

Follow line / subsequent block

Allows us to specify the value of the condition to follow a specific branch

- Flowchart properties
  - Expression result





Subsequent (Cancel?, Move to safe position_end)

Flowchart condition:

Expression result

( ) true
[✓ false]

Compare expression:

Iceland Liechtenstein Norway grants

UNIVERSITATEA LUCIAN BLAGA DIN SIBIU

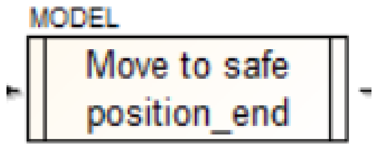
USN University of South-Eastern Norway

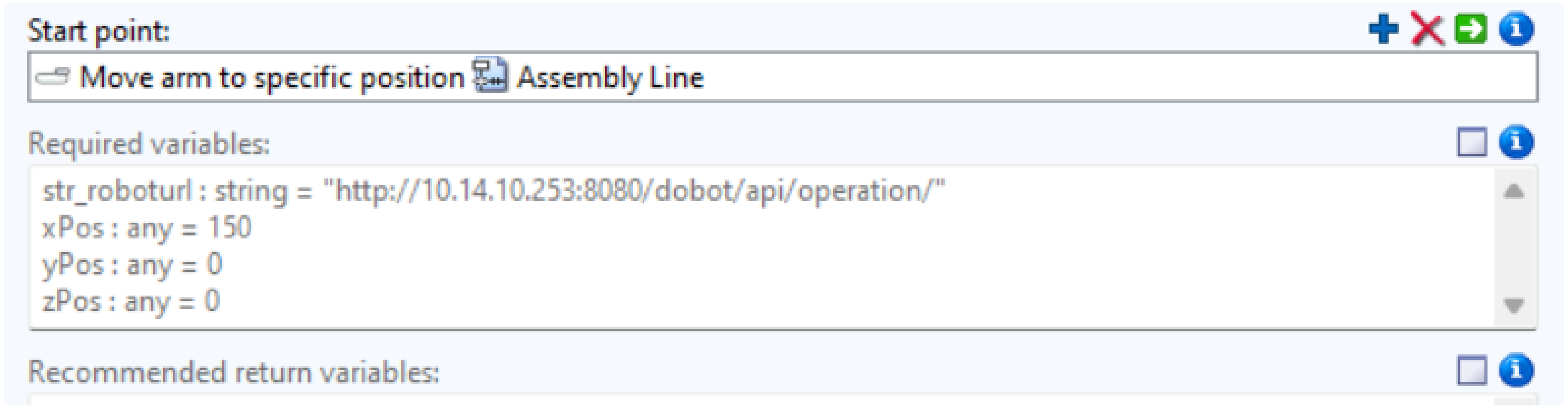
MAPIoT

Predefined functions / external operations block

Allows us to call a specific implemented function

- Description
  - Name
  - Activated
  - External type -> Model
- Model
  - Start point – select the called model
  - Passed variables




Start point:

Move arm to specific position Assembly Line

Required variables:

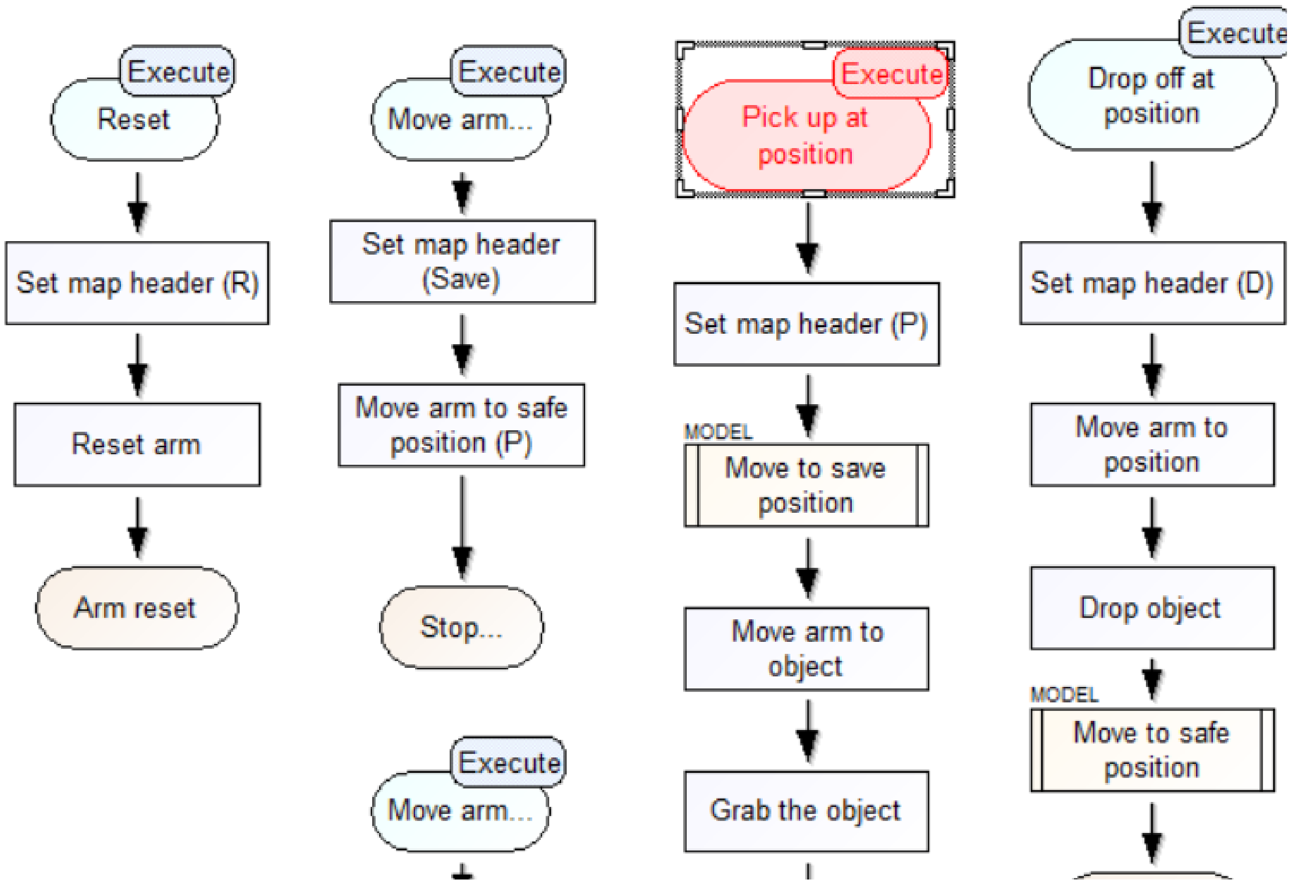
```
str_roboturl : string = "http://10.14.10.253:8080/dobot/api/operation/"
xPos : any = 150
yPos : any = 0
zPos : any = 0
```

Recommended return variables:

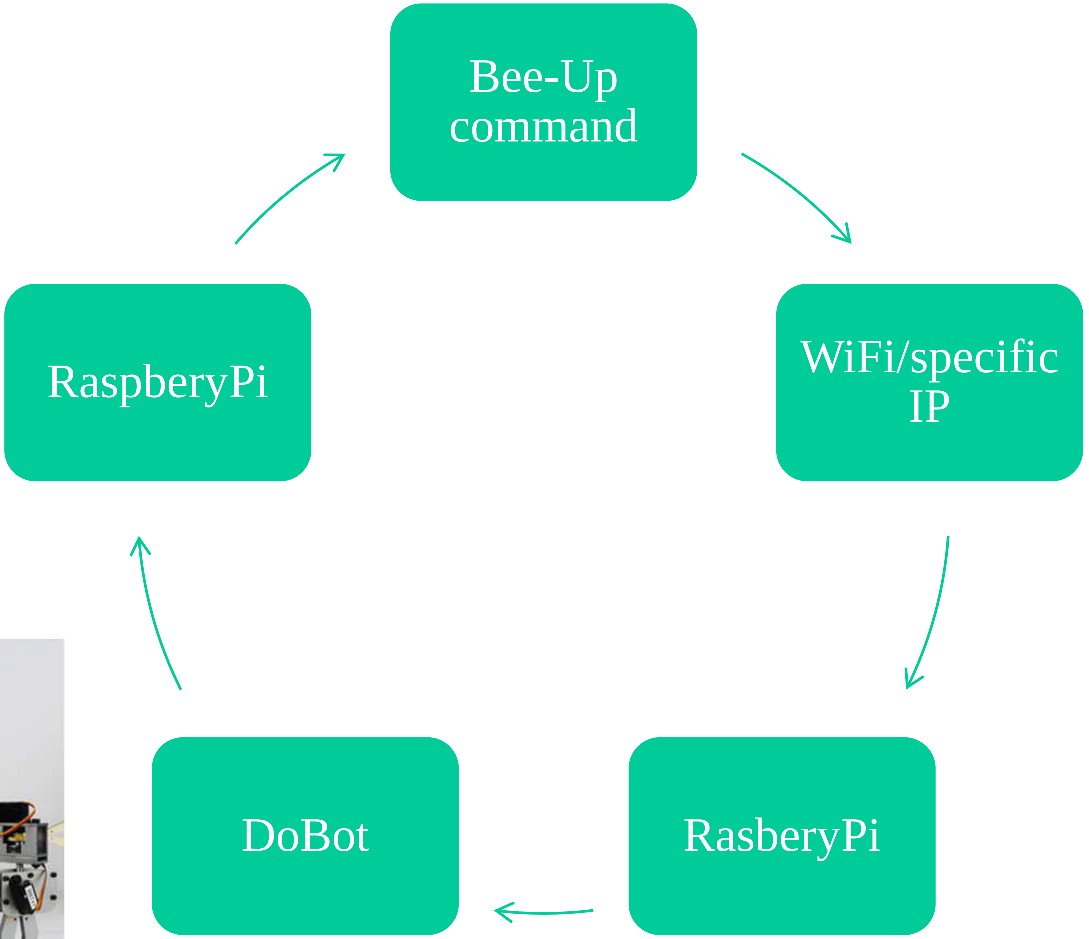
Iceland Liechtenstein Norway grants — Universitatea Lucian Blaga din Sibiu — University of South-Eastern Norway — MAPIoT

Models for Runtime - Used Models

- Available capabilities
  - Reset
  - Move to a specific position
  - Grab on (Pick up at position)
  - Grab off (Drop off at position)



Modeling/Controlling CPS components:



ADOxx/Bee-Up interfaces with cyber-physical systems

✓ `HTTP_SEND_REQUEST (str_url)`

```
   str_method:string
   map_reqheaders:map
   str_reqbody:string
   val_respcode:reference
   map_respheaders:reference
   str_respbody:reference
```

- ✓ `str_url` – the URL that should be contacted provided as a string
- ✓ `str_method` – the HTTP method that should be sent with the request. (Usually POST)
- ✓ `ap_reqheaders` – the headers that should be sent with the request as a map.
- ✓ `val_respcode` – a reference variable that will contain an integer with the response code.
- ✓ `map_respheaders` – a reference variable that will contain a map with the headers of the response.
- ✓ `str_respbody` / `arr_respbody` – a reference variable that will hold the body of the response

AdoScript Commands and feedback

- ✓ VIEWBOX opens a view box to display longer text messages.

```
CC "AdoScript" VIEWBOX  text:strValue [ title:strValue ] [ fontname:strValue ]
                                      [ fontheight:intValue ]
```

- ✓ EDITBOX opens a box where the user can edit text.

```
CC "AdoSript" EDITBOX text:strValue [ title:strValue ] [ oktext:strValue ]
                      [ fontname:strValue ] [ fontheight:intValue ] [ fileeditor ].
```

- ✓ LISTBOX opens a box where the user can select values of a list of values.

```
CC "AdoScript" LISTBOX entries:strValue [ toksep:strValue ]
          [ selection:strValue ] [ title:strValue ]
          [ boxtext:strValue ] [ oktext:strValue ]
                 [ w:intValue h:intValue ] [ extra:{ Extra } ].
```

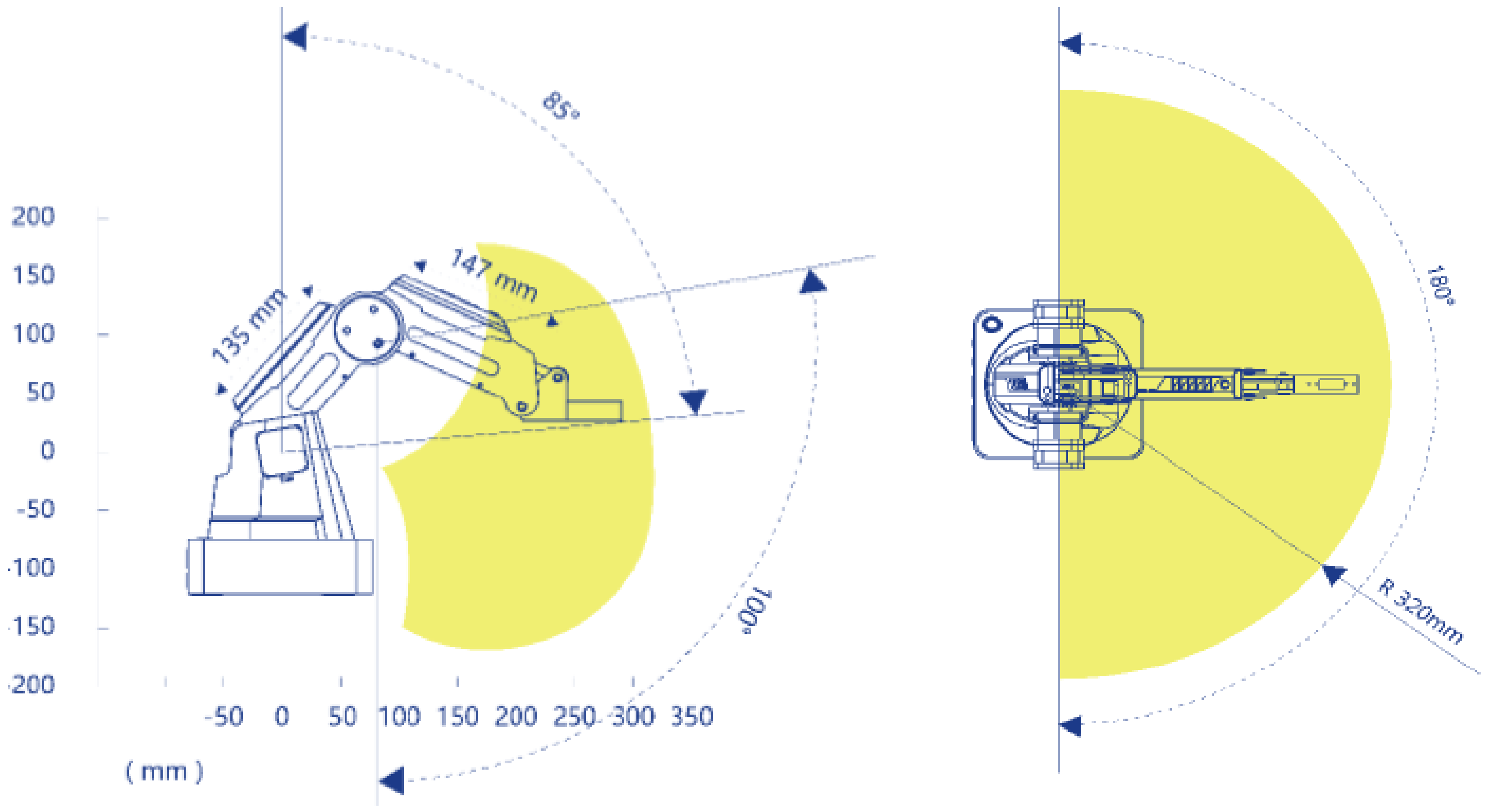
More commands: https://www.adoxx.org/AdoScriptDoc/

Developing an application for the DoBot robotic arm

- ✓ DoBot – list with all commands and a short description
  - ✓ web address http://10.14.10.253:8080/dobot/ui/#/
  - ✓ moveToPosition x,y,z
  - ✓ moveToHomePosition
  - ✓ turnOnSuctionCup
  - ✓ TurnOffSuctionCup
  - ✓ getPosition
- ✓ Working DoBot address: http://10.14.10.253:8080/dobot/api/operation
- ✓ Example move to x=200, y=0, z=0:

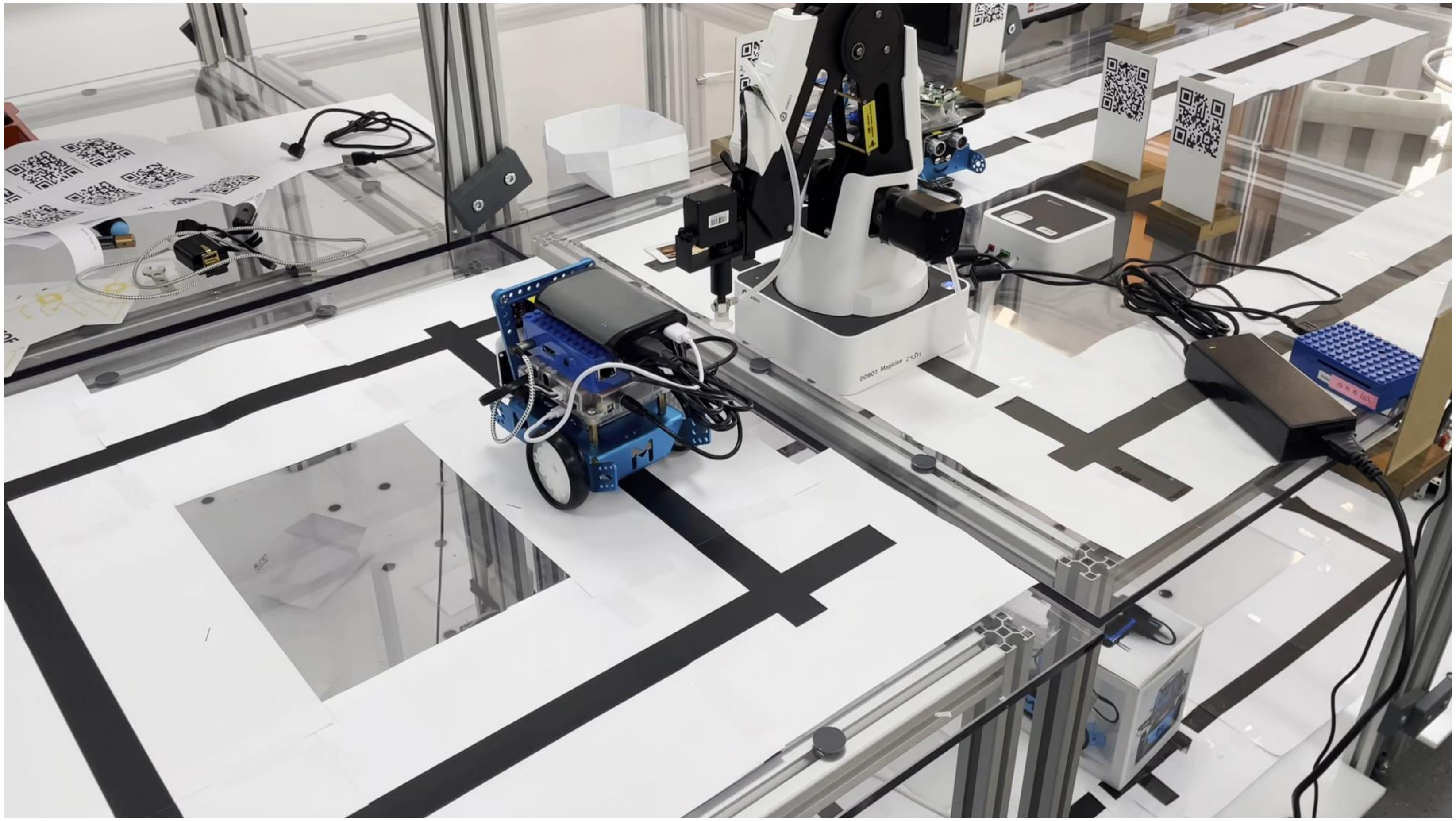
```
HTTP_SEND_REQUEST
("http://10.14.10.253:8080/dobot/api/operation/moveToPosition ?x=200&y=0&z=0")
str_method:("POST")                      map_reqheaders:(map_headers)
str_reqbody:("")                          val_respcode:val_httpcode
map_respheaders:map_respheaders          str_respbody:str_respbody
```

DoBot coordinates



https://forum.dobot.cc/t/coordinates-and-gripper-range-for-ptpcmd-in-dobot-magician/203/4

DOBOT in action



Developing an application for the mobile robot.

- ✓ mBot – list with all commands and a short descriptions
  - ✓ Using the web interface http://10.14.10.252:8080/mBot/ui/
  - ✓ mBot API for Line Following – need to specify the speed (between 50-200)
    - ✓ moveStraight – moves until both sensors are out of line
    - ✓ turnRight – moves until right sensor is on the line
    - ✓ turnLeft – moves until left sensor is on the line
    - ✓ jumpGap – moves until both sensors are on the line
  - ✓ mBot Movement Operation – need to specify the speed and duration in seconds
    - ✓ moveForward – moves forward for a period
    - ✓ moveBackward – moves backward for a period
    - ✓ turnRight – turn right a specified period
    - ✓ turnLeft – turn left a specified period

mBot commands

- ✓ mBot – list with all commands and a short description
  - ✓ Using the web interface http://10.14.10.252:8080/mBot/ui/
  - ✓ mBot API for Obstacle Avoidance – moves until it encounters an obstacle at the minimum distance specified (the sensor specification 5-80 cm)
    - ✓ moveForwardObstacle – move forward until the obstacle at the specified distance is met.
    - ✓ moveBackwardObstacle – move backward in order to increase the distance from the met obstacle.
    - ✓ turnRightObstacle –move in right direction until the obstacle disappears
    - ✓ turnLeftObstacle – move in the left direction until the obstacle disappears

- ✓ Working mBot address: http://10.14.10.252:8080/mBot/api/

mBot in action