SoS MDE Tutorial

AMADEOS Blockly4SoS

Motivations and HowTo

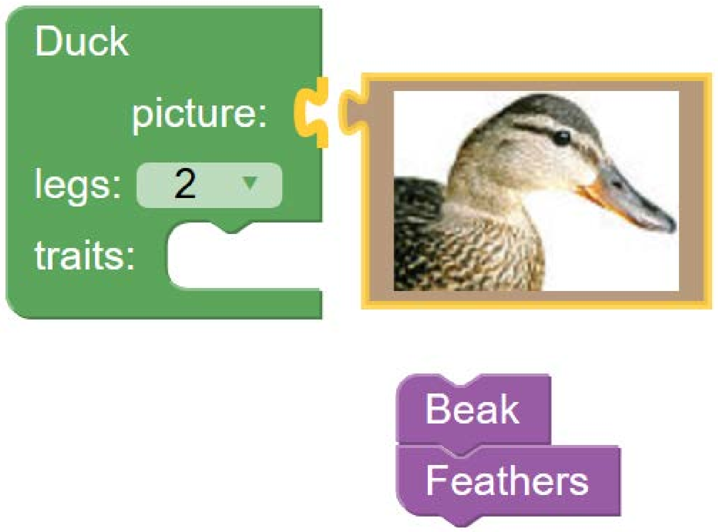
https://blockly4sos.resiltech.com

The tool can be downloaded here!!!

Requirements

- Any web browser (firefox recommended) for the tool at https://blockly4sos.resiltech.com
- Python 2.7
- PlantUML viewer/Atom → https://atom.io/
  - Install package plantuml-viewer
    (open Atom; Packages→ Settings View → Open → Install; write «plantuml-viewer»)
    (for visualizing sequence diagrams: load the code; CTRL-ALT-P)

Google Blockly



1. Is a visual programming editor, used to program using blocks
2. Only compatible blocks can be connected together
3. Can be made "correct by design"
4. Supports code and XML generation
5. Only a modern web browser is required (any device/OS)

Example of applications using Blockly

Most basic example:

- https://developers.google.com/blockly/
  - → let's have a look
  - → really didactic!

More resources:

- Blockly games examples:
- https://blockly-games.appspot.com/
- More serious:
  - Fashion – https://www.madewithcode.com/projects/fashion
  - Stock market – https://bot.binary.com/bot.html
  - Android – appinventor.mit.edu/explore/designer-blocks.html
- Electronics:
  - Codebug – https://www.codebug.org.uk/create/codebug/new/
  - Ardublockly – http://ardublockly.embeddedlog.com/

Blockly4SoS supporting facility tool

A tool to:

model

and

simulate

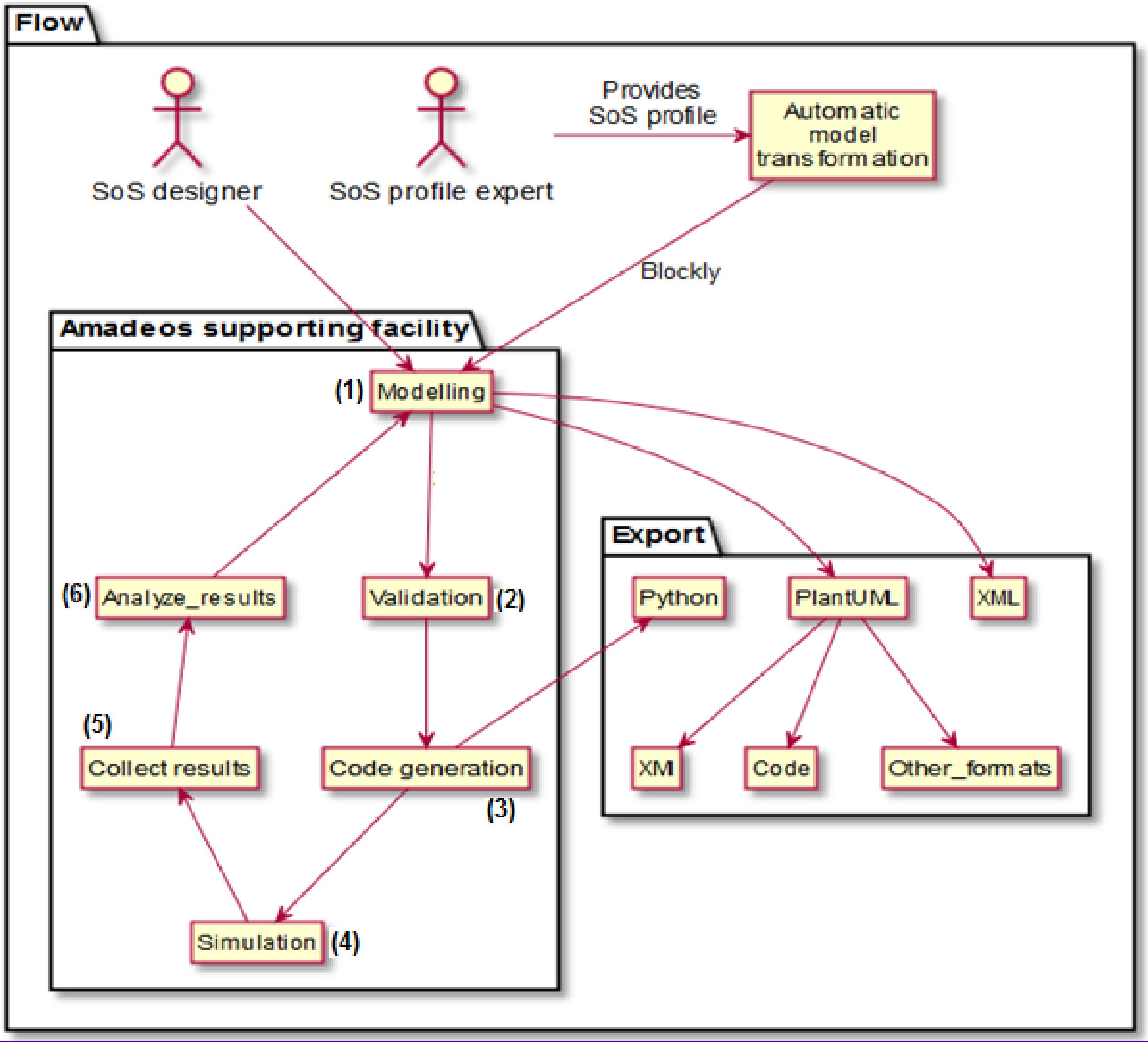
Systems-of-Systems

- Link to the homepage of tool

http://blockly4sos.resiltech.com

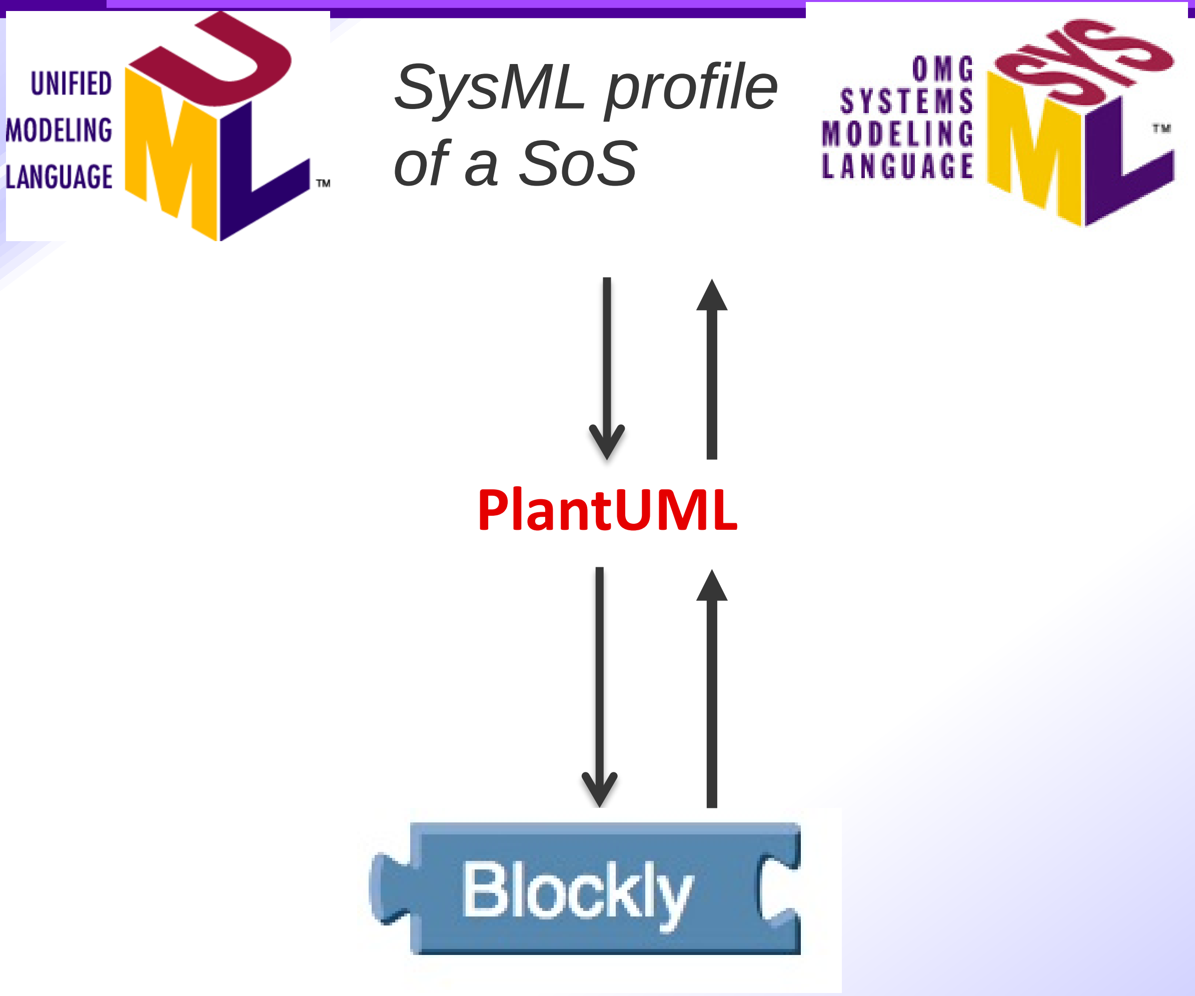
Though any modern browser is OK, Firefox is the recommended browser (I will use Chrome)

The overall MDE workflow



(1) SoS designer starts modelling SoS using Blockly
(2) The model is validated based on the constraints defined
(3) Executable code is generated in Python
(4) Various scenarios are **simulated**
(5) Results are collected through logs
(6) Logs are analyzed for design/run-time errors/mistakes

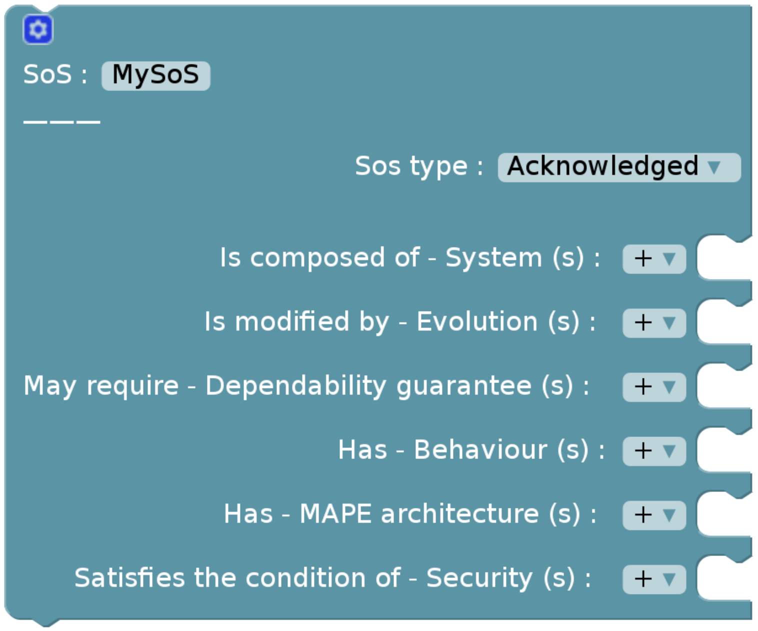
Model transformation



The use of PlantUML as intermediate language makes debugging of model transformation easier

Let's start with a simple block

By default, an SoS block is created on the workspace



SoS : MySoS

———

Sos type : Acknowledged

Is composed of - System (s) : +

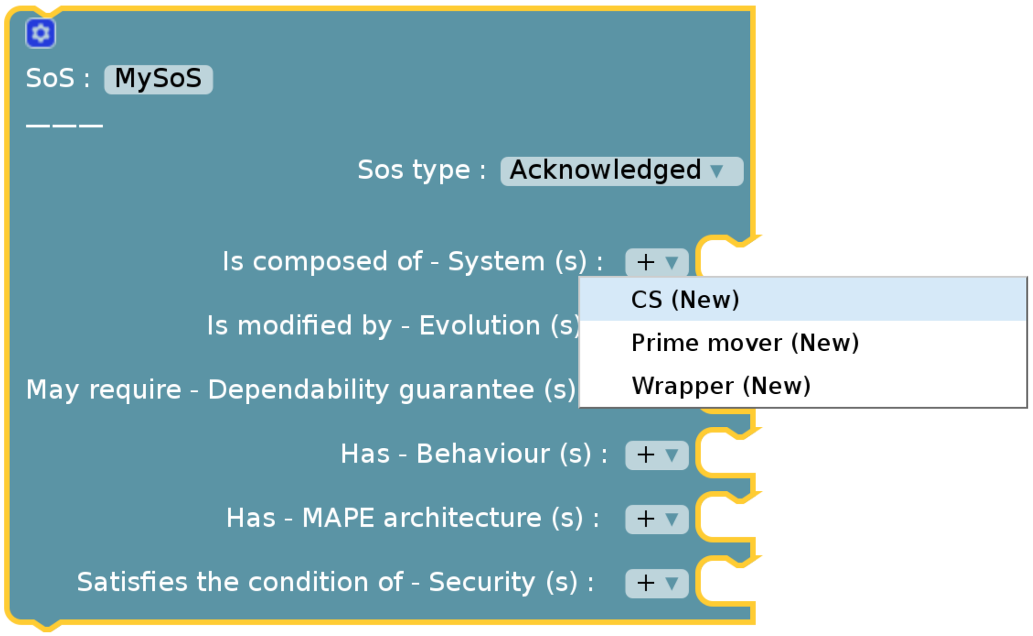
Is modified by - Evolution (s) : +

May require - Dependability guarantee (s) : +

Has - Behaviour (s) : +

Has - MAPE architecture (s) : +

Satisfies the condition of - Security (s) : +



SoS : MySoS

———

Sos type : Acknowledged

Is composed of - System (s) : +

Is modified by - Evolution (s)

May require - Dependability guarantee (s)

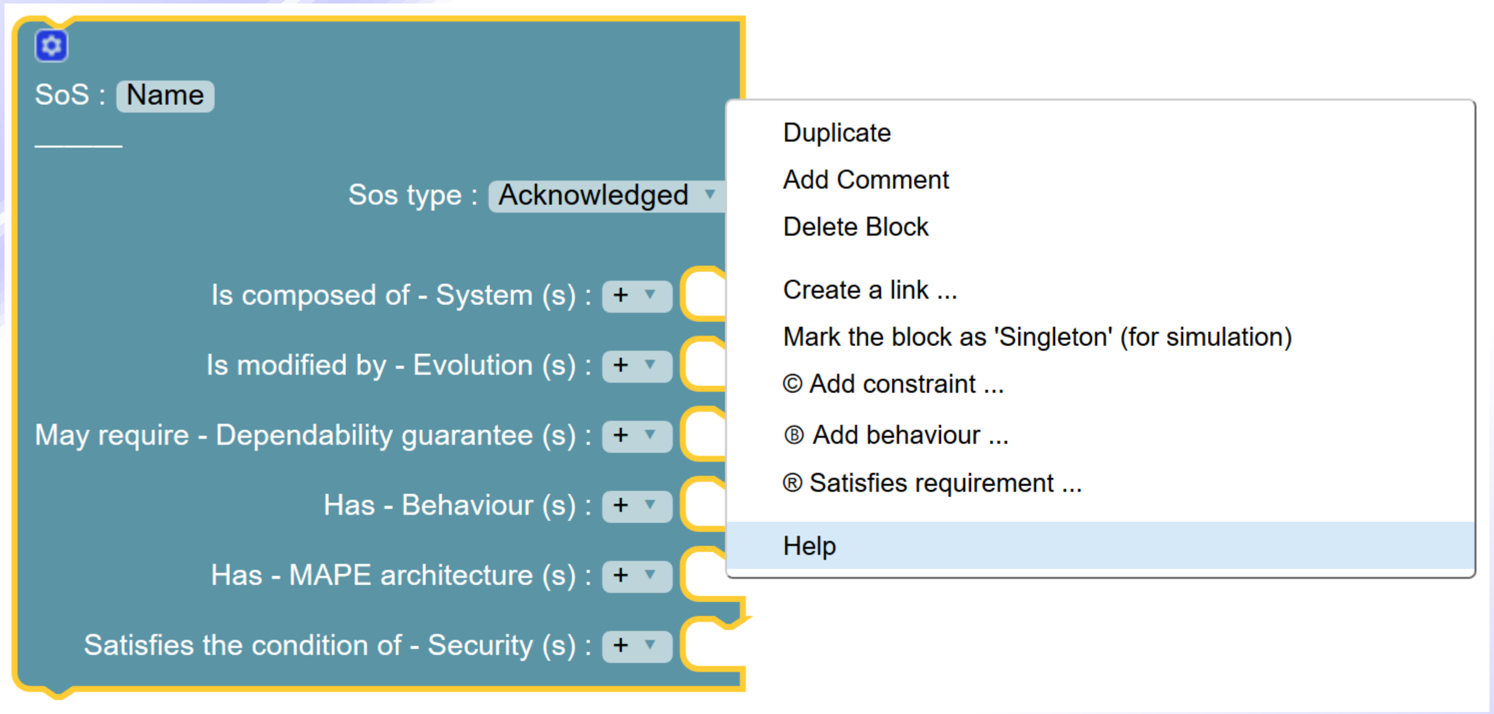
Has - Behaviour (s) : +

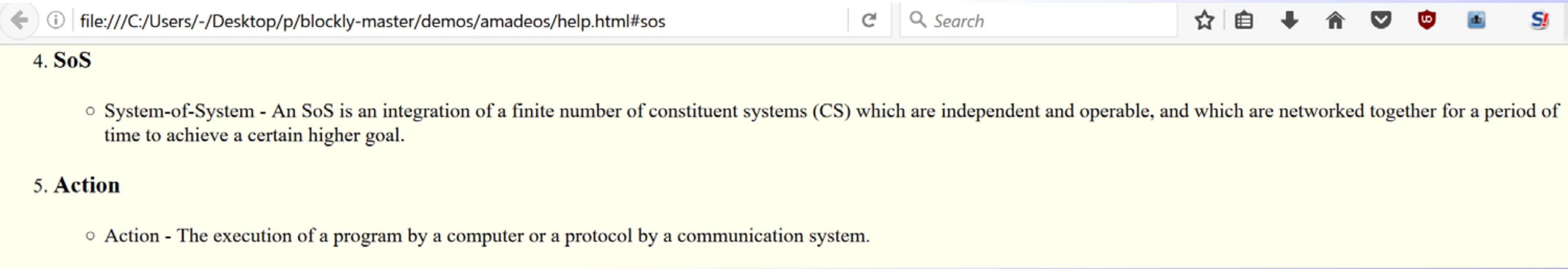
Has - MAPE architecture (s) : +

Satisfies the condition of - Security (s) : +

Figure: Add new blocks by clicking on the (+) drop-down/from left-hand side toolbox

Help/Glossary


SoS : Name
Sos type : Acknowledged
Is composed of - System (s) : +
Is modified by - Evolution (s) : +
May require - Dependability guarantee (s) : +
Has - Behaviour (s) : +
Has - MAPE architecture (s) : +
Satisfies the condition of - Security (s) : +
Duplicate
Add Comment
Delete Block
Create a link ...
Mark the block as 'Singleton' (for simulation)
© Add constraint ...
Ⓑ Add behaviour ...
® Satisfies requirement ...
Help


file:///C:/Users/-/Desktop/p/blockly-master/demos/amadeos/help.html#sos
Search

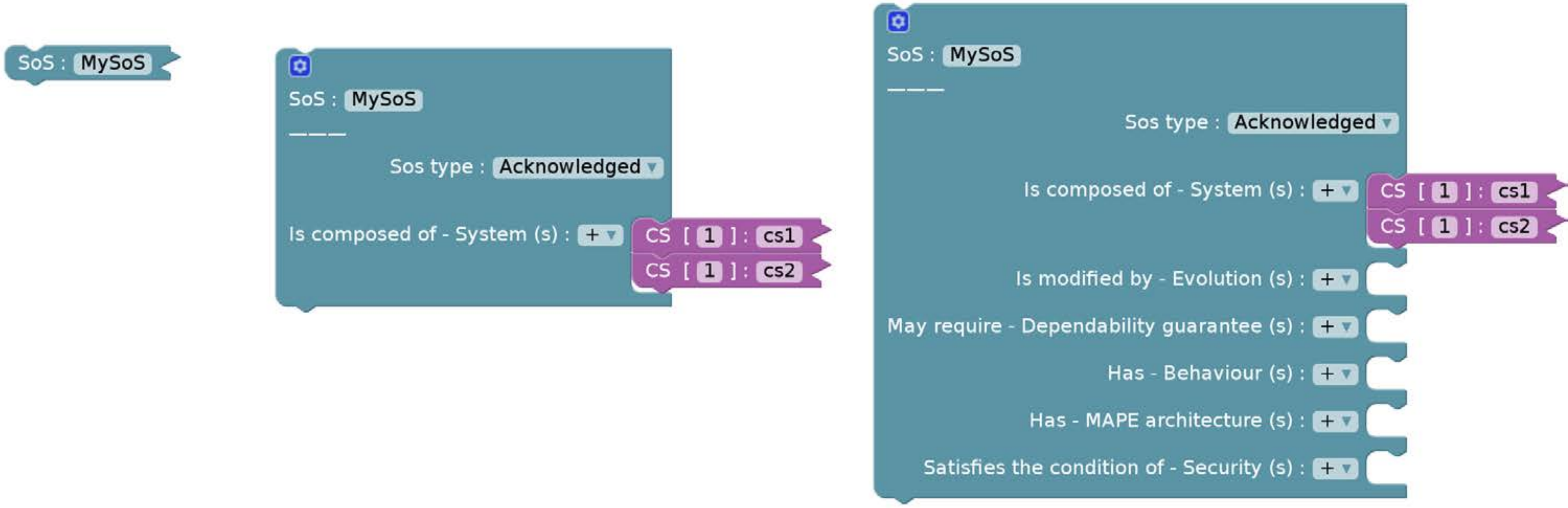
4. **SoS**

   - System-of-System - An SoS is an integration of a finite number of constituent systems (CS) which are independent and operable, and which are networked together for a period of time to achieve a certain higher goal.

5. **Action**

   - Action - The execution of a program by a computer or a protocol by a communication system.

Three ways of viewing a block


SoS : MySoS

SoS : MySoS

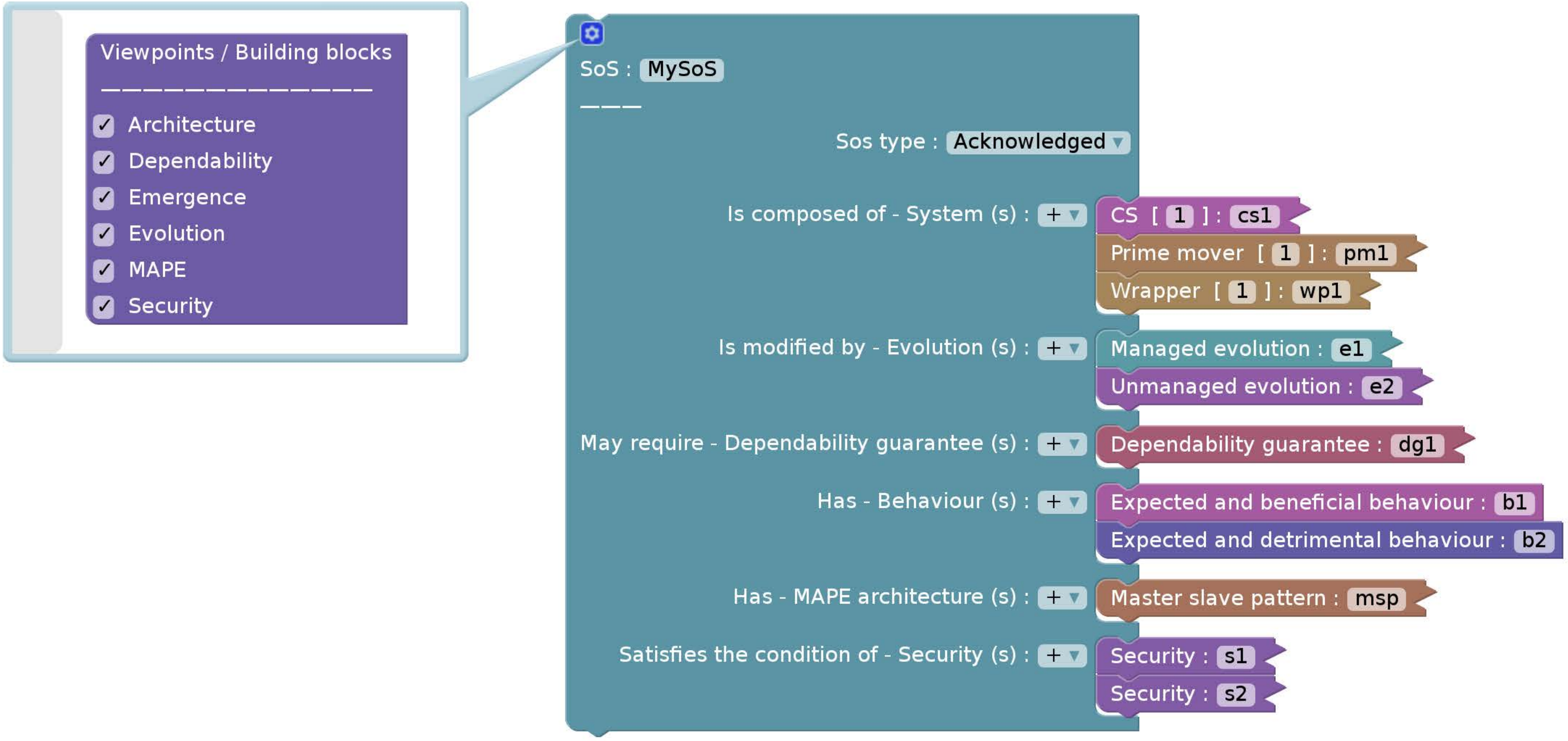
Sos type : Acknowledged
Is composed of - System (s) : +
CS [1] : cs1
CS [1] : cs2

SoS : MySoS

Sos type : Acknowledged
Is composed of - System (s) : +
CS [1] : cs1
CS [1] : cs2
Is modified by - Evolution (s) : +
May require - Dependability guarantee (s) : +
Has - Behaviour (s) : +
Has - MAPE architecture (s) : +
Satisfies the condition of - Security (s) : +

Figure: 3 ways of viewing a block - cycle between views by **double clicking** the block

All viewpoints and building blocks of a block



Viewpoints / Building blocks
————————————
- ✓ Architecture
- ✓ Dependability
- ✓ Emergence
- ✓ Evolution
- ✓ MAPE
- ✓ Security

SoS : MySoS
———
Sos type : Acknowledged

Is composed of - System (s) : + ▾
- CS [1] : cs1
- Prime mover [1] : pm1
- Wrapper [1] : wp1

Is modified by - Evolution (s) : + ▾
- Managed evolution : e1
- Unmanaged evolution : e2

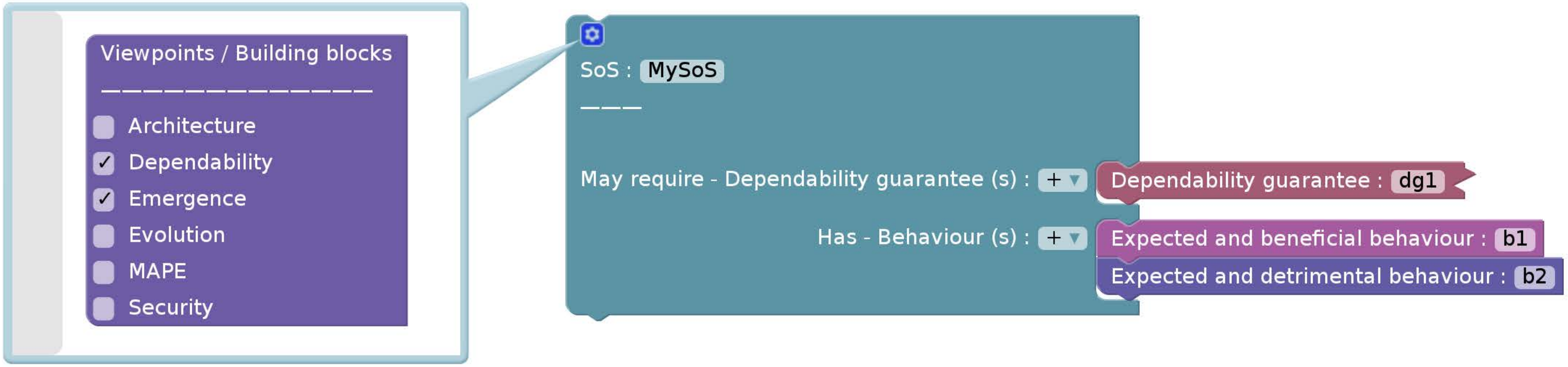
May require - Dependability guarantee (s) : + ▾
- Dependability guarantee : dg1

Has - Behaviour (s) : + ▾
- Expected and beneficial behaviour : b1
- Expected and detrimental behaviour : b2

Has - MAPE architecture (s) : + ▾
- Master slave pattern : msp

Satisfies the condition of - Security (s) : + ▾
- Security : s1
- Security : s2

Filter some of the viewpoints/ building blocks



Viewpoints / Building blocks

——————————————

- ☐ Architecture
- ✓ Dependability
- ✓ Emergence
- ☐ Evolution
- ☐ MAPE
- ☐ Security

SoS : MySoS

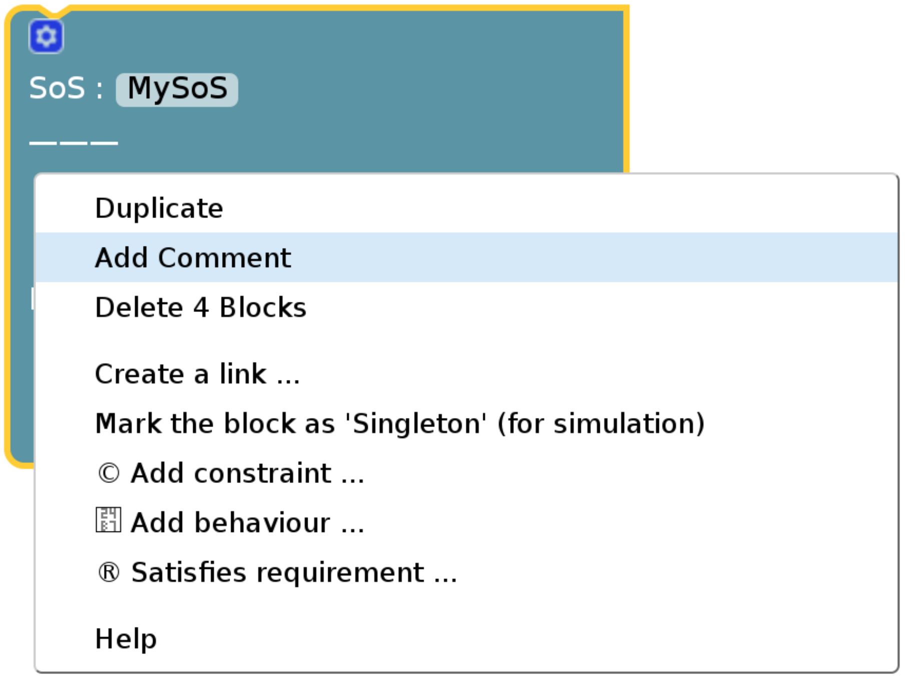
———

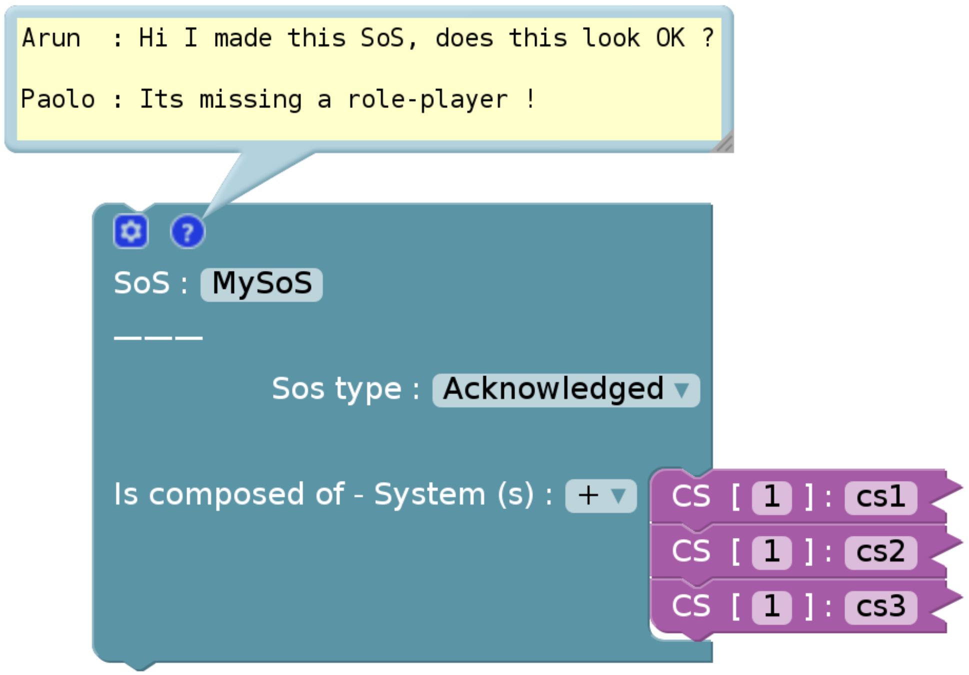
May require - Dependability guarantee (s) : + ▾ Dependability guarantee : dg1

Has - Behaviour (s) : + ▾ Expected and beneficial behaviour : b1

Expected and detrimental behaviour : b2

Comment your design

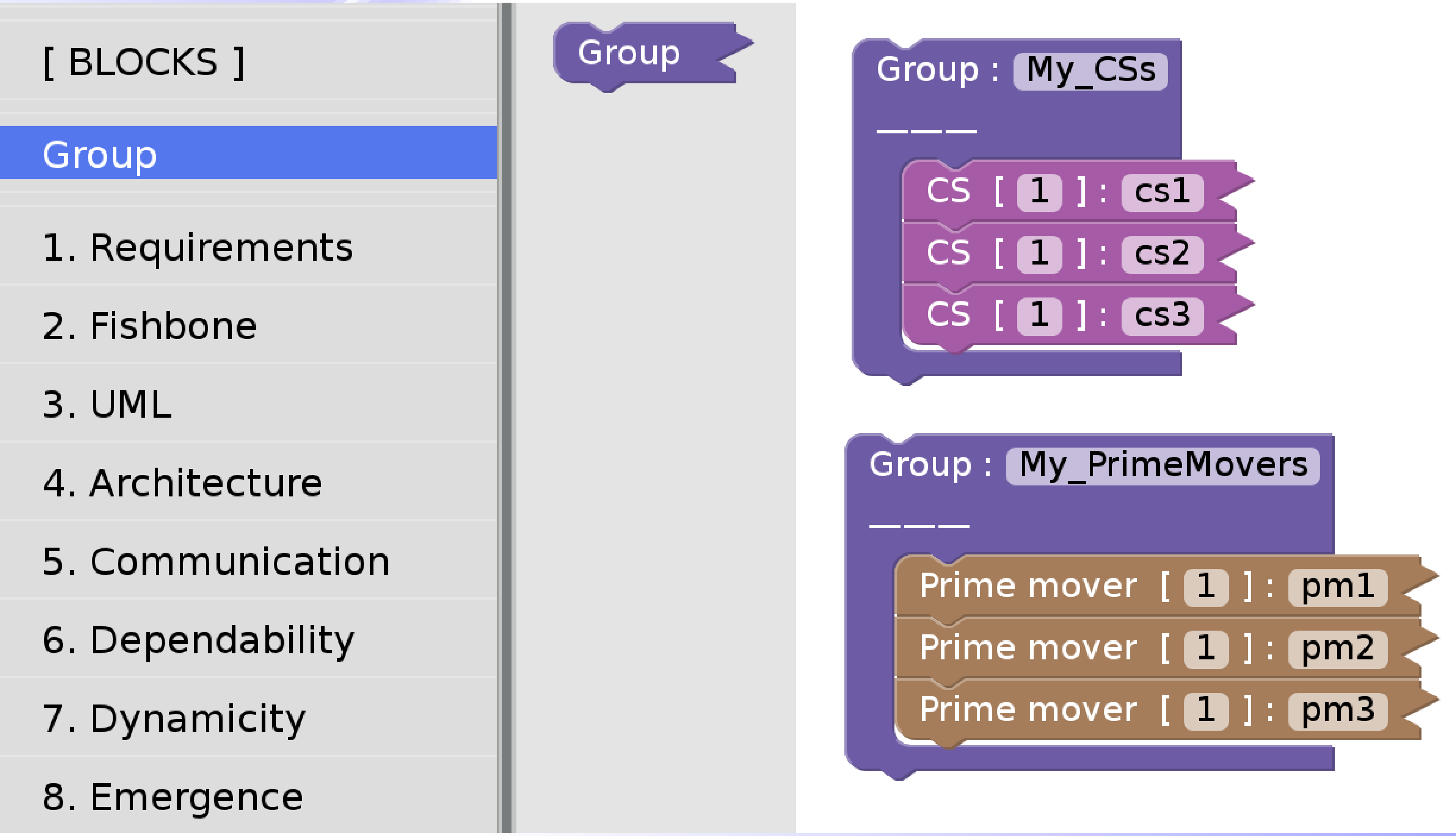




```
Arun  : Hi I made this SoS, does this look OK ?

Paolo : Its missing a role-player !
```

Modularize the design by grouping

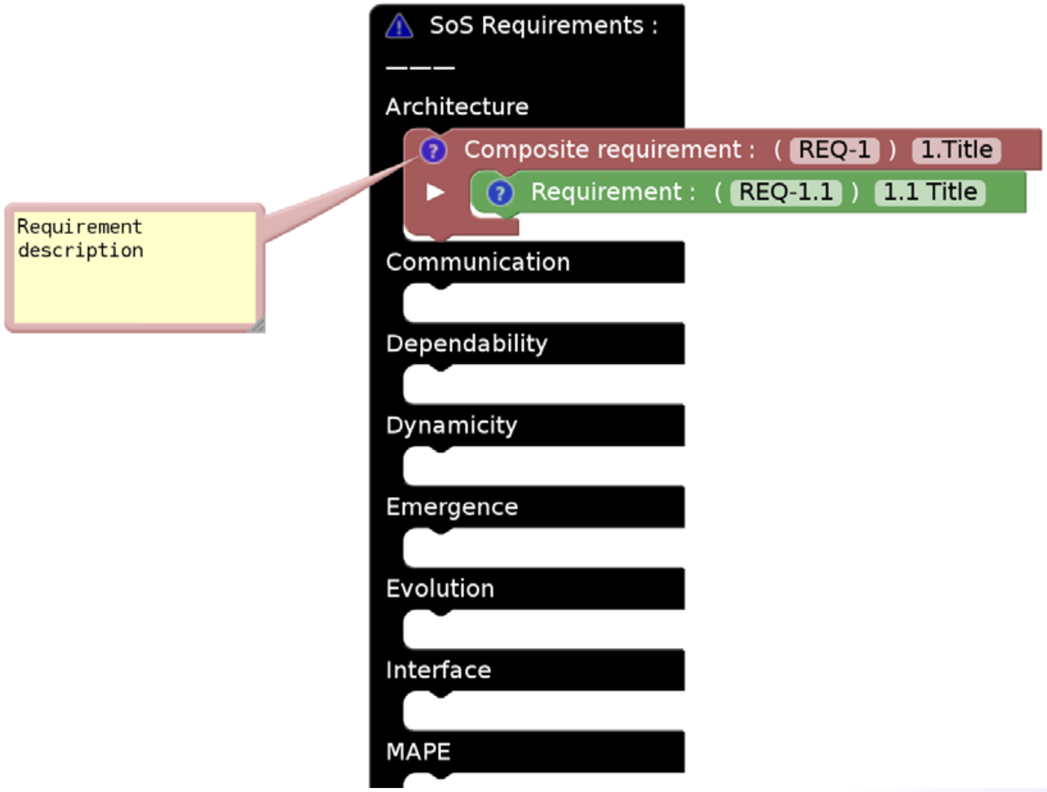


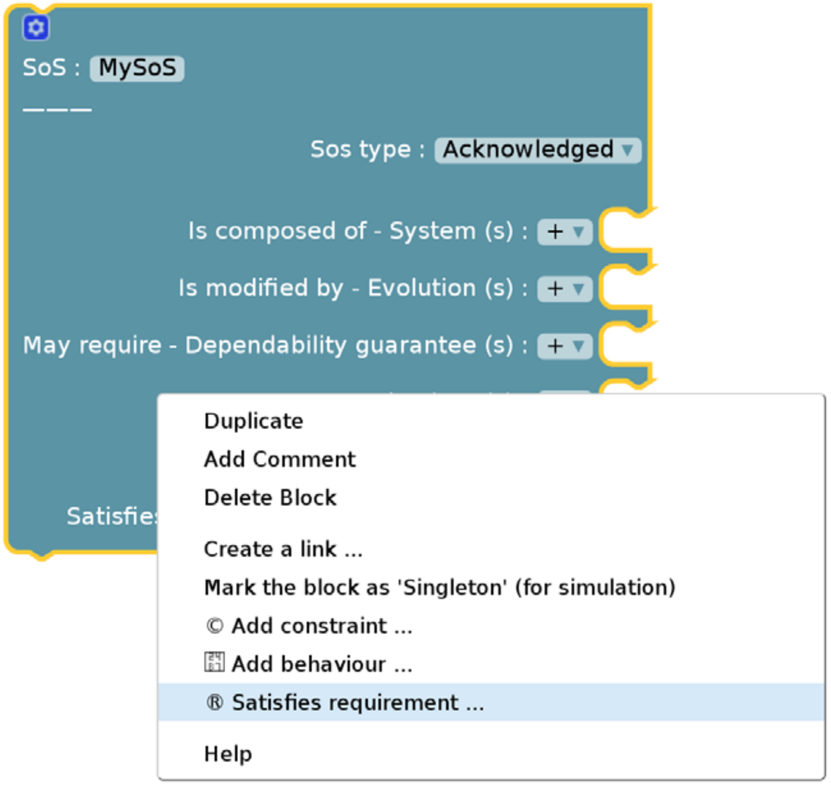
[BLOCKS]

Group

1. Requirements
2. Fishbone
3. UML
4. Architecture
5. Communication
6. Dependability
7. Dynamicity
8. Emergence

Manage requirements for each viewpoint




SoS : MySoS
———
Sos type : Acknowledged
Is composed of - System (s) : +
Is modified by - Evolution (s) : +
May require - Dependability guarantee (s) : +
Satisfies
Duplicate
Add Comment
Delete Block
Create a link ...
Mark the block as 'Singleton' (for simulation)
© Add constraint ...
Add behaviour ...
® Satisfies requirement ...
Help

Model validation

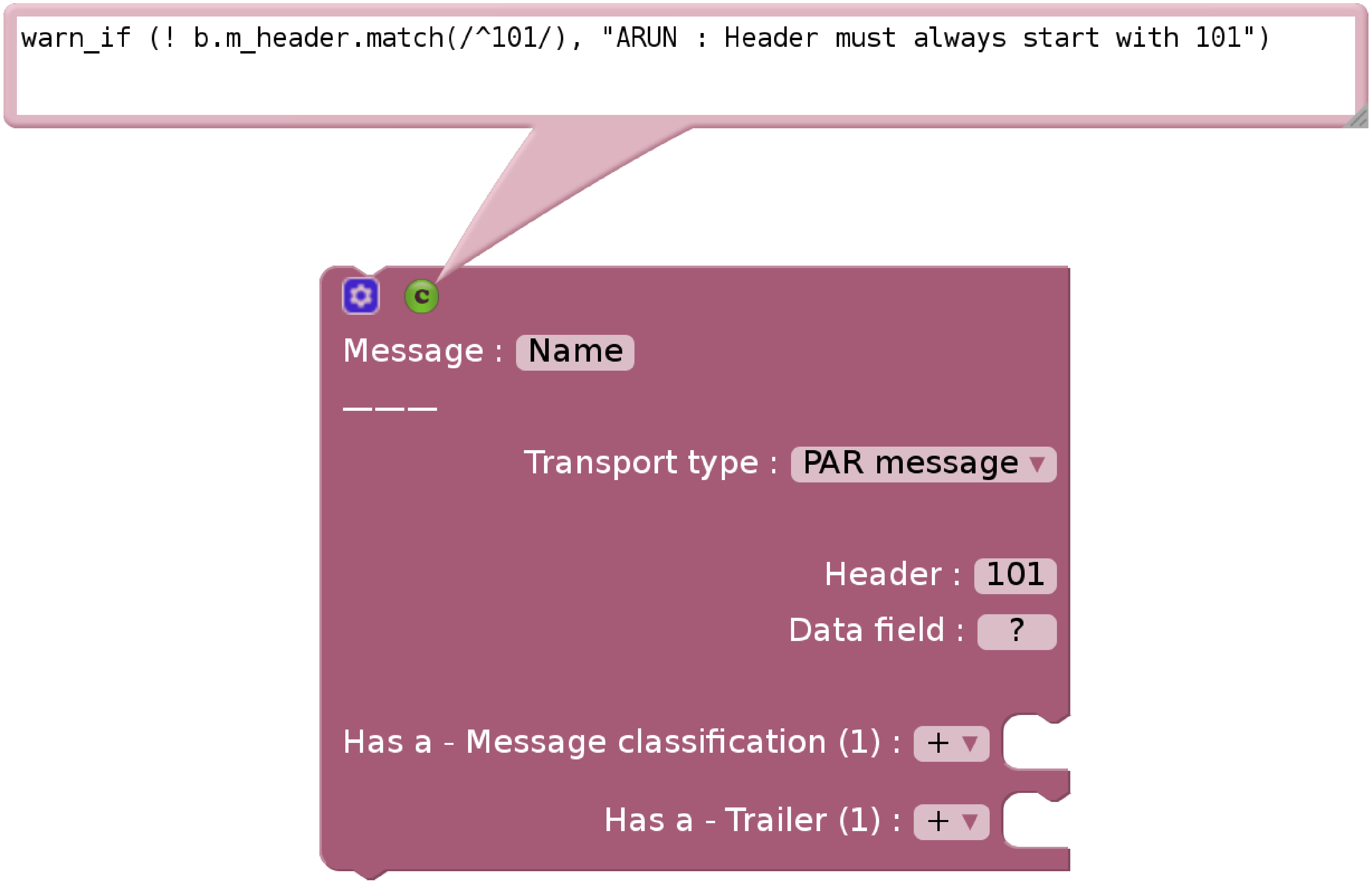
By default, Blockly models validation by letting only compatible blocks to be connected with each other.

User can add custom validation in JavaScript by using the below constraint functions:

```
1. warn_if ( on_condition , " WarningMessage ");
2. detach_if ( on_condition , block );
```

Two helper functions for model validation

Model validation example – looks ok



```
warn_if (! b.m_header.match(/^101/), "ARUN : Header must always start with 101")
```

Message : Name

———

Transport type : PAR message

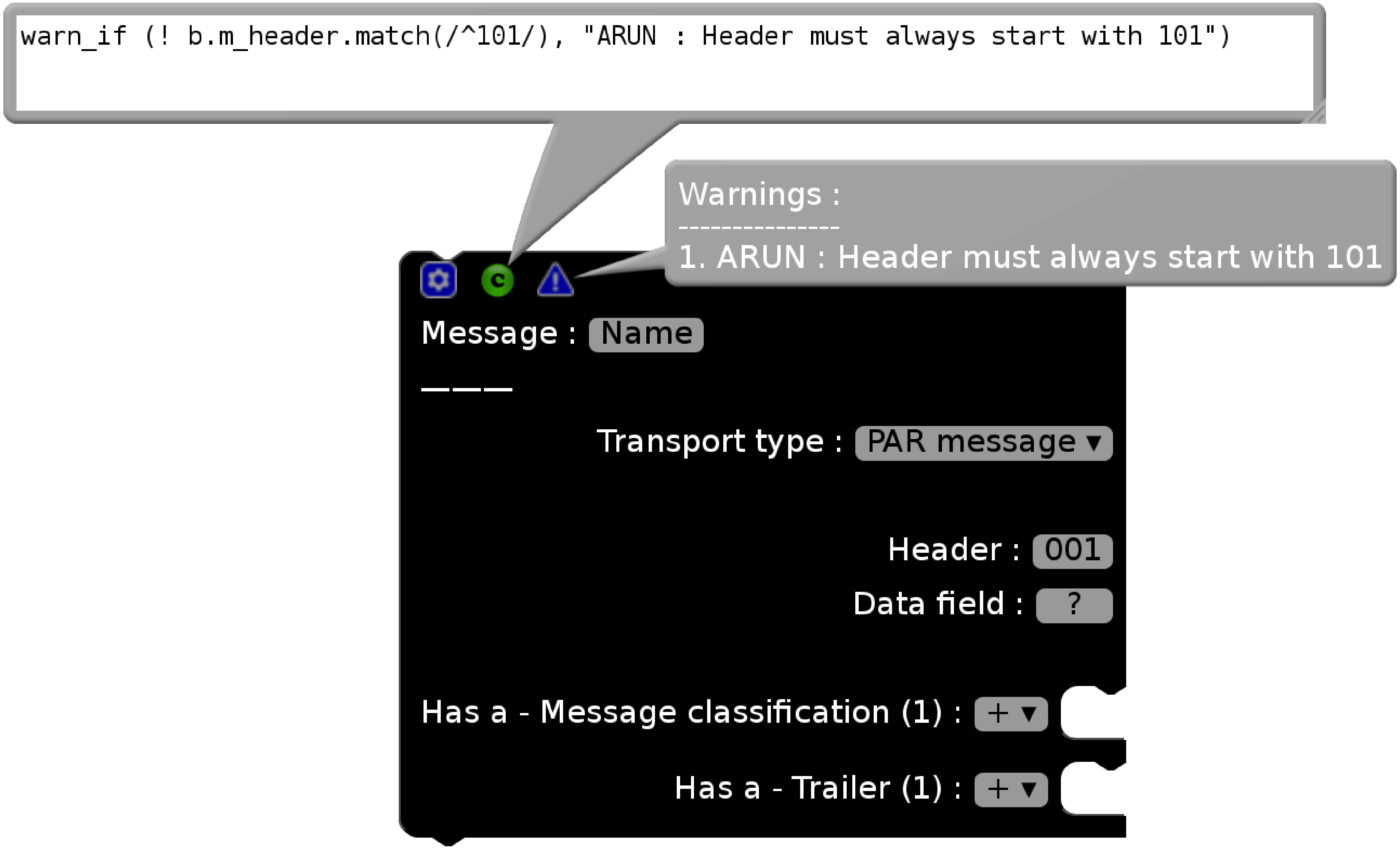
Header : 101

Data field : ?

Has a - Message classification (1) : +

Has a - Trailer (1) : +

Model validation example - a warning



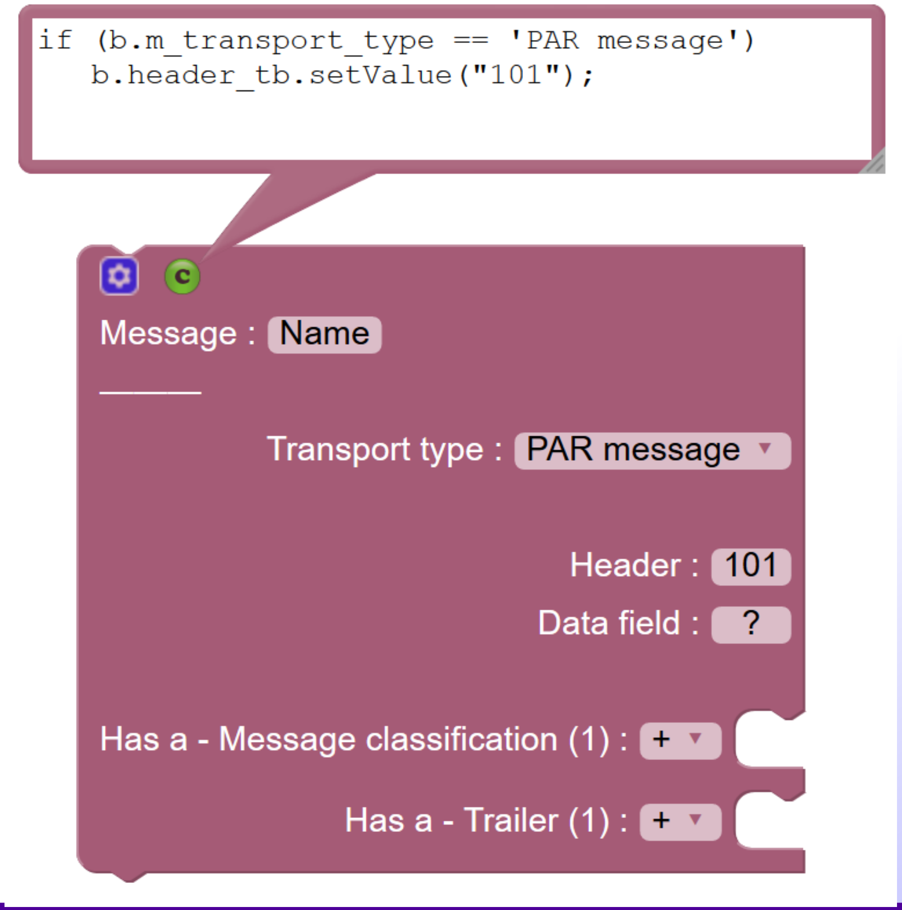
```
warn_if (! b.m_header.match(/^101/), "ARUN : Header must always start with 101")
```

Warnings :
----
1. ARUN : Header must always start with 101

Message : Name
Transport type : PAR message ▾
Header : 001
Data field : ?
Has a - Message classification (1) : + ▾
Has a - Trailer (1) : + ▾

Forcing values !

Some times its useful to forcefully set values instead of showing warnings !



```
if (b.m_transport_type == 'PAR message')
   b.header_tb.setValue("101");
```

Message : Name

Transport type : PAR message

Header : 101

Data field : ?

Has a - Message classification (1) : +

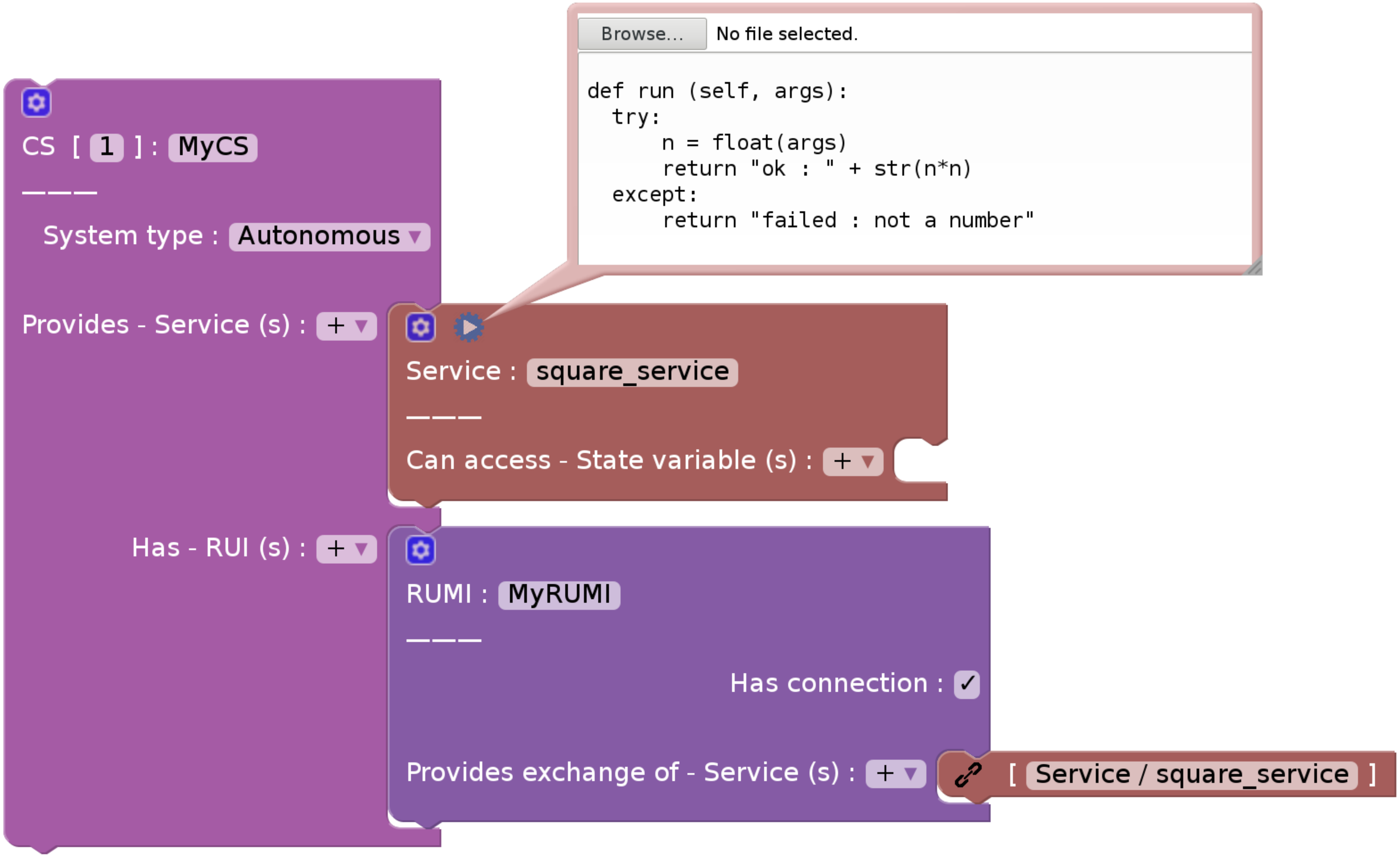
Has a - Trailer (1) : +

Dynamic behavior modeling

- **Why ?**
  - A static model is like a car without an engine !

- **Prerequisite for running simulations:**
  - Python 2.7 (preferably at c:npython27 directory)
  - PlantUML viewer (atom editor) for viewing results
  - You may also install other software/system .... to interact with the simulation software !

Add behavior – Dynamic part of model



Browse... No file selected.

```
def run (self, args):
  try:
      n = float(args)
      return "ok : " + str(n*n)
  except:
      return "failed : not a number"
```

CS [1] : MyCS

System type : Autonomous

Provides - Service (s) :

Service : square_service

Can access - State variable (s) :

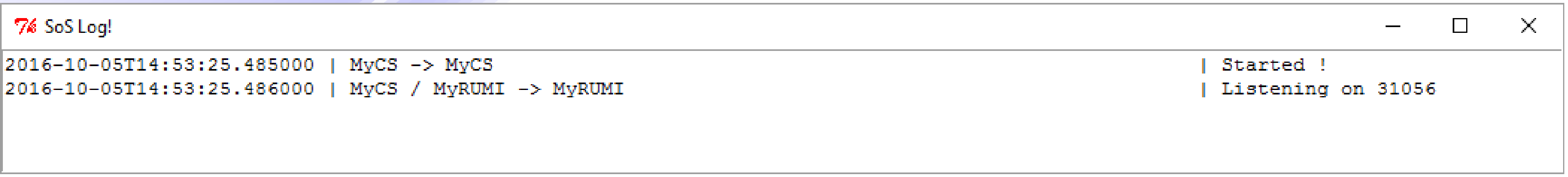
Has - RUI (s) :

RUMI : MyRUMI

Has connection : ✓

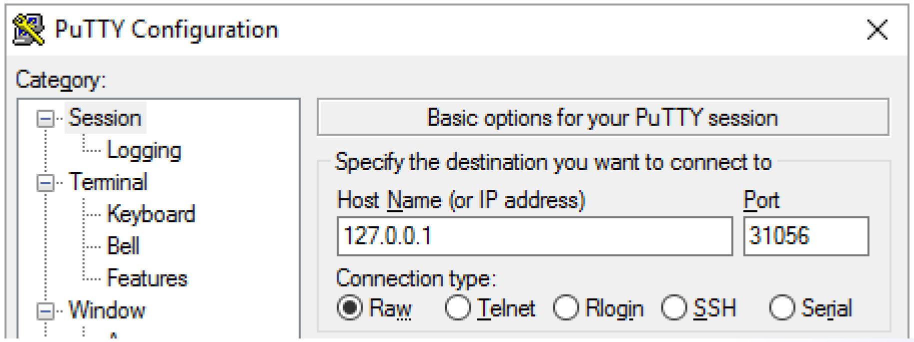
Provides exchange of - Service (s) : [Service / square_service]

Simulation run



SoS Log!

```
2016-10-05T14:53:25.485000 | MyCS -> MyCS                                          | Started !
2016-10-05T14:53:25.486000 | MyCS / MyRUMI -> MyRUMI                               | Listening on 31056
```



PuTTY Configuration

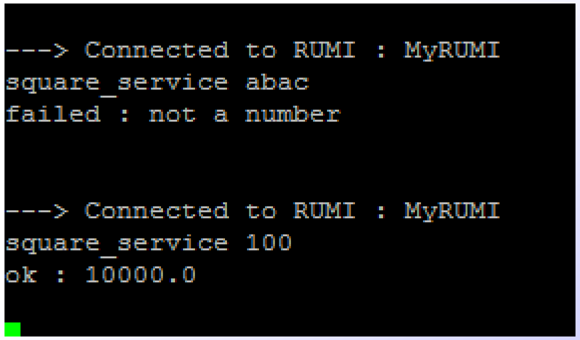
Basic options for your PuTTY session

Specify the destination you want to connect to

Host Name (or IP address): 127.0.0.1

Port: 31056

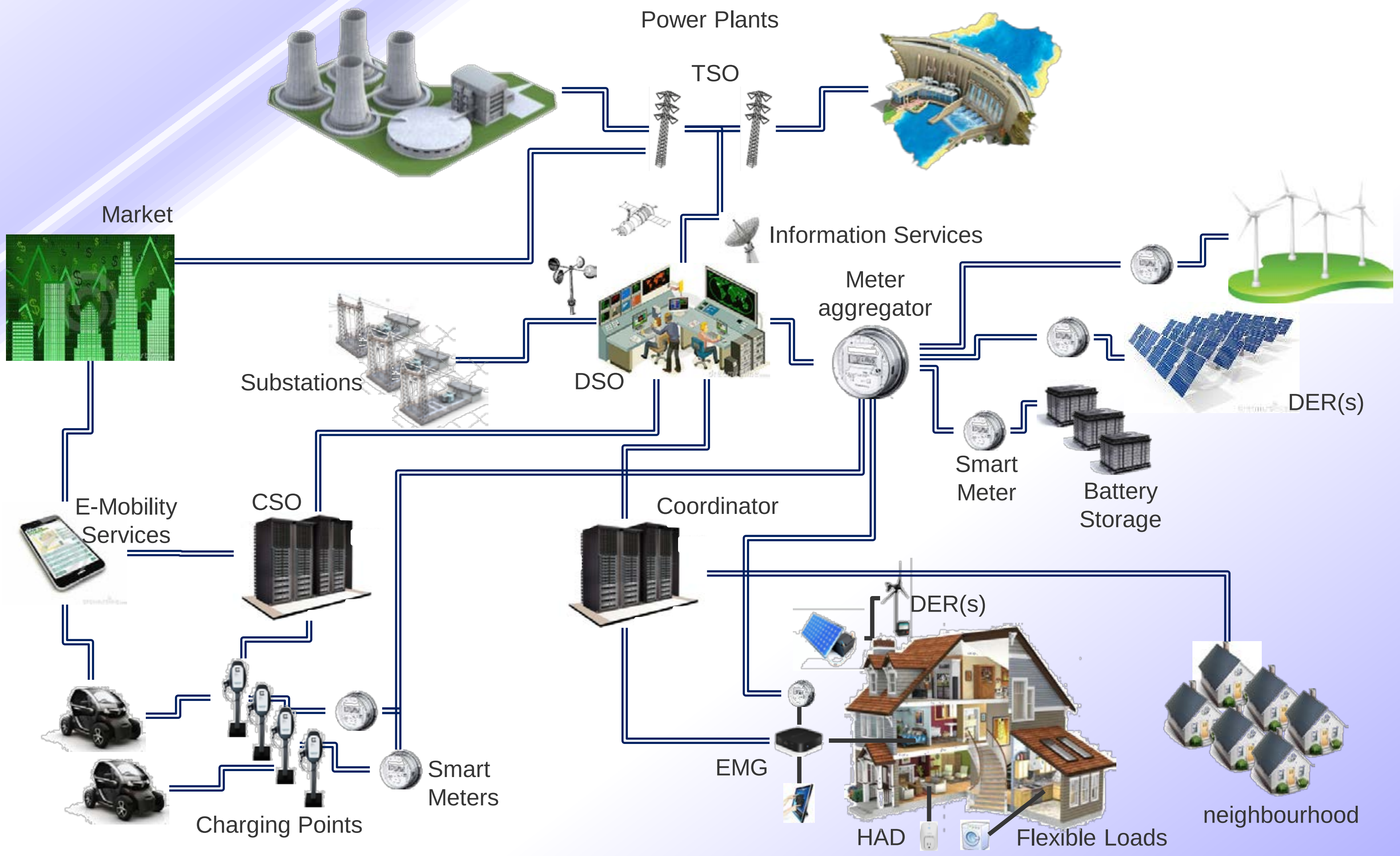
Connection type: [✓ Raw] Telnet Rlogin SSH Serial



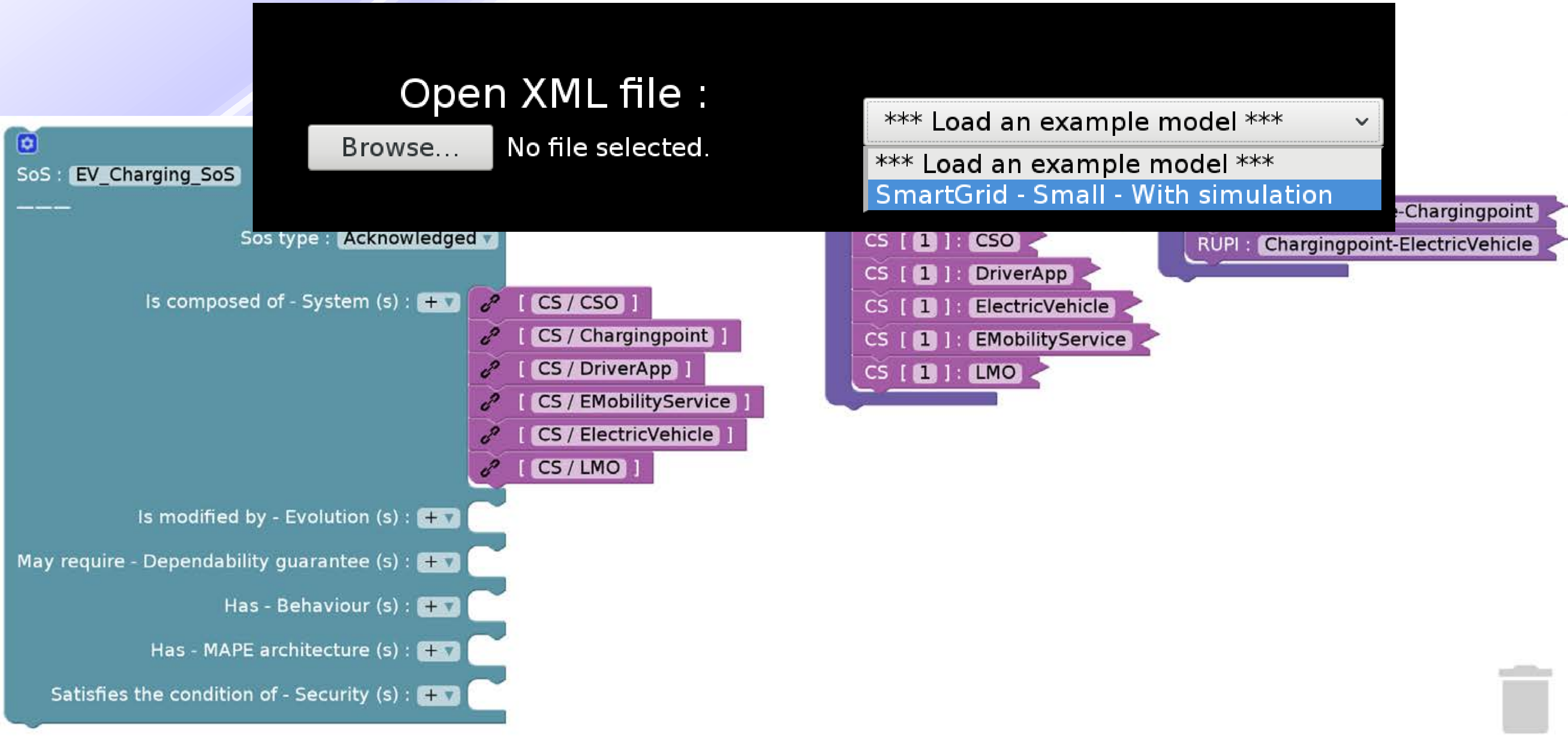
```
---> Connected to RUMI : MyRUMI
square_service abac
failed : not a number


---> Connected to RUMI : MyRUMI
square_service 100
ok : 10000.0
```

Example model: smart grids



Load the example model


Open XML file :

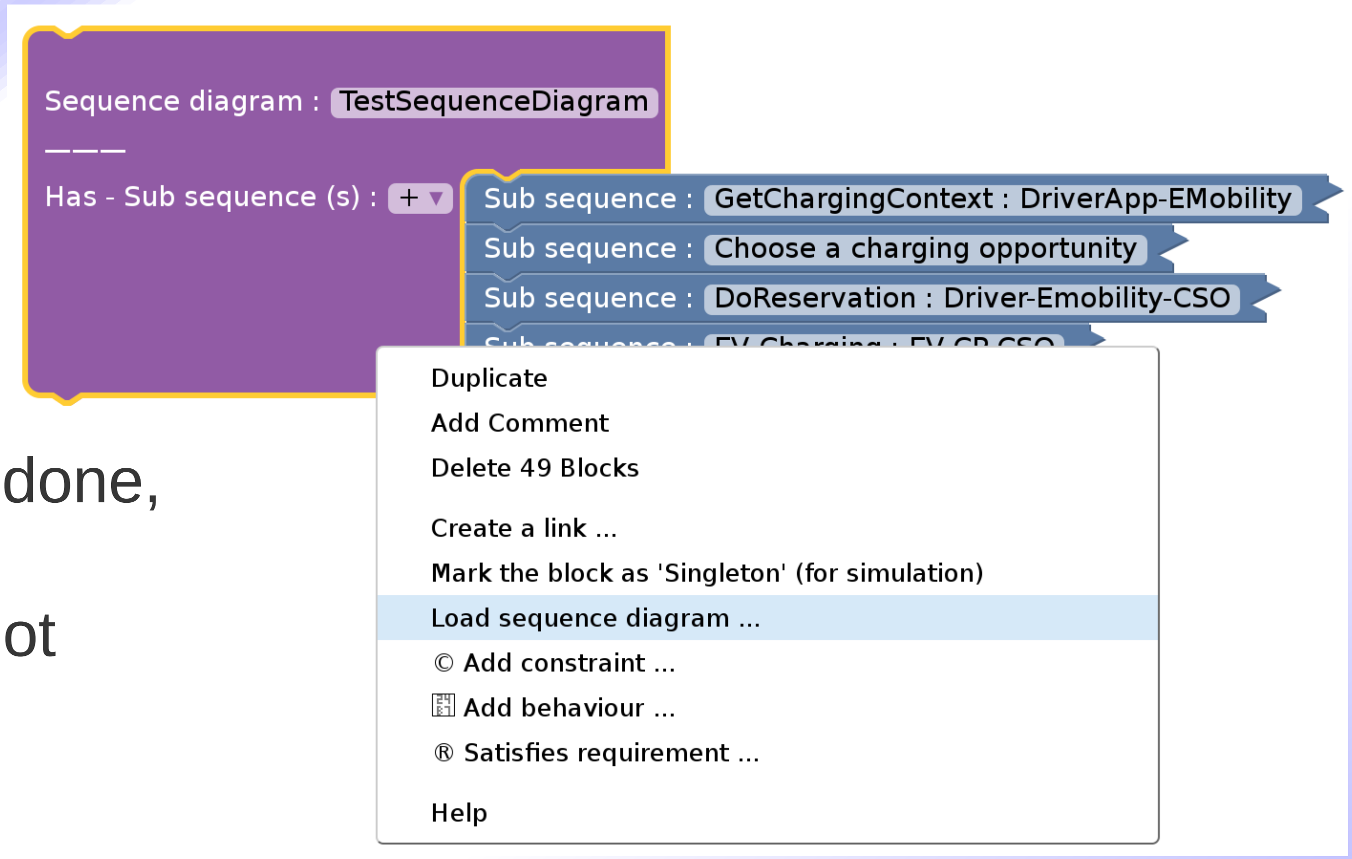
Browse... No file selected.

*** Load an example model ***

*** Load an example model ***
SmartGrid - Small - With simulation

SoS : EV_Charging_SoS
Sos type : Acknowledged
Is composed of - System (s) :
[CS / CSO]
[CS / Chargingpoint]
[CS / DriverApp]
[CS / EMobilityService]
[CS / ElectricVehicle]
[CS / LMO]
Is modified by - Evolution (s) :
May require - Dependability guarantee (s) :
Has - Behaviour (s) :
Has - MAPE architecture (s) :
Satisfies the condition of - Security (s) :
CS [1] : CSO
CS [1] : DriverApp
CS [1] : ElectricVehicle
CS [1] : EMobilityService
CS [1] : LMO
-Chargingpoint
RUPI : Chargingpoint-ElectricVehicle

Load sequence diagram



Sequence diagram : TestSequenceDiagram

Has - Sub sequence (s) :

Sub sequence : GetChargingContext : DriverApp-EMobility

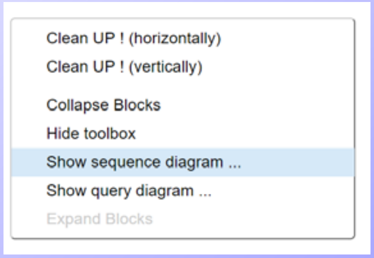
Sub sequence : Choose a charging opportunity

Sub sequence : DoReservation : Driver-Emobility-CSO

Duplicate
Add Comment
Delete 49 Blocks
Create a link ...
Mark the block as 'Singleton' (for simulation)
Load sequence diagram ...
© Add constraint ...
Add behaviour ...
® Satisfies requirement ...
Help

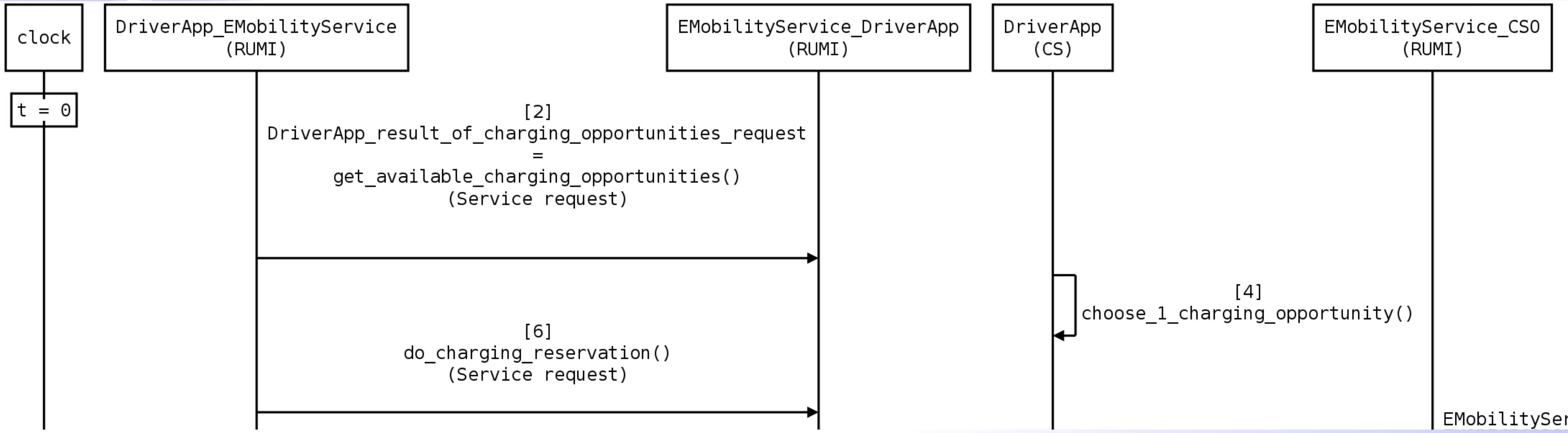
If this step is not done, the seq. diagram simulation may not work.

Right click on workspace to view the sequence diagram menu



Clean UP ! (horizontally)
Clean UP ! (vertically)
Collapse Blocks
Hide toolbox
Show sequence diagram ...
Show query diagram ...
Expand Blocks

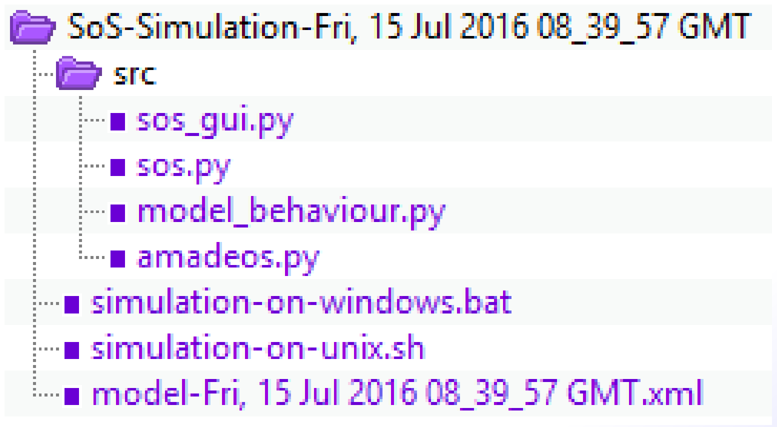
Load sequence diagram


clock
DriverApp_EMobilityService
(RUMI)
EMobilityService_DriverApp
(RUMI)
DriverApp
(CS)
EMobilityService_CSO
(RUMI)
t = 0
[2]
DriverApp_result_of_charging_opportunities_request
=
get_available_charging_opportunities()
(Service request)
[4]
choose_1_charging_opportunity()
[6]
do_charging_reservation()
(Service request)
EMobilitySer

Auto generated sequence diagram

Code generation

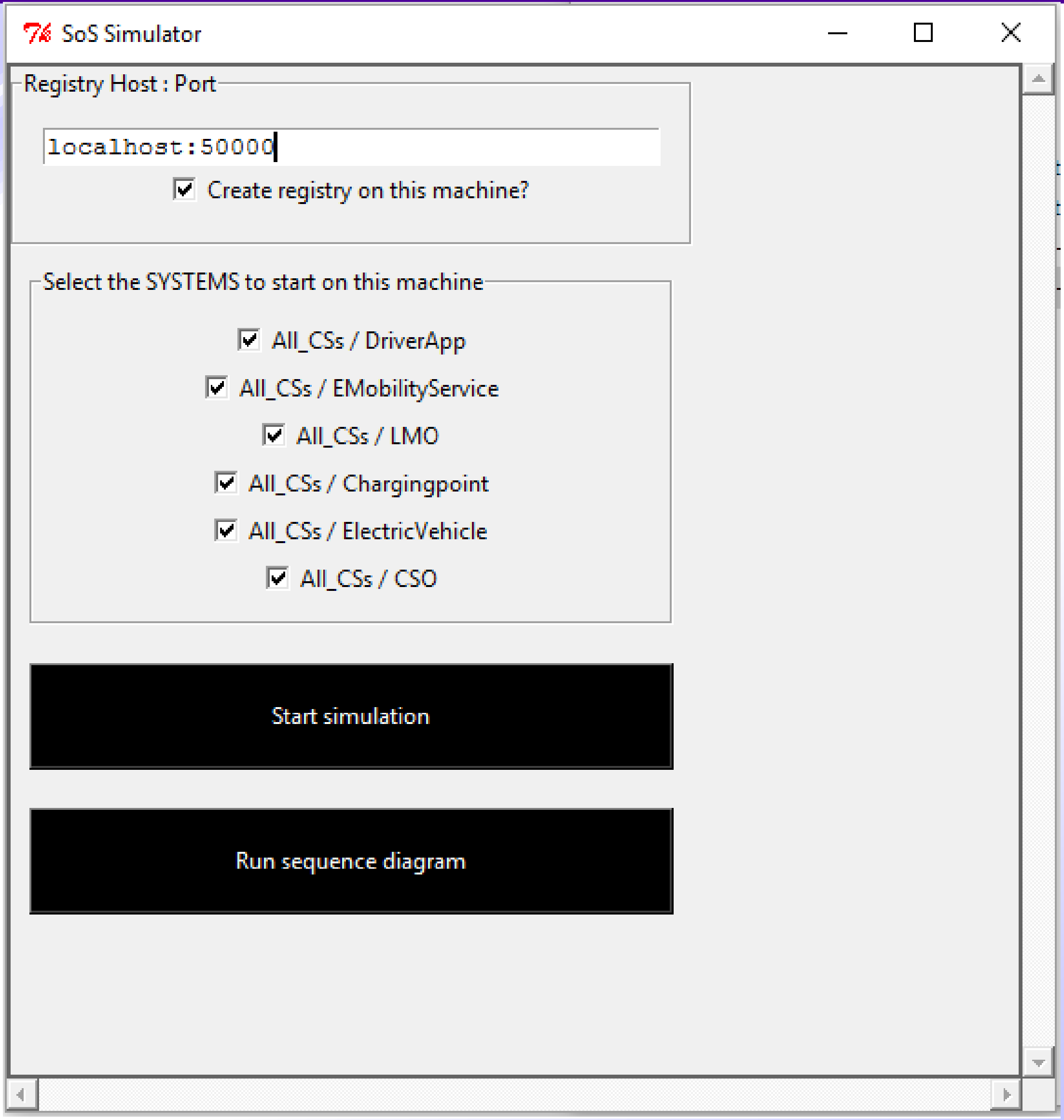
The simulation code is generated in the following format:



```
SoS-Simulation-Fri, 15 Jul 2016 08_39_57 GMT
├── src
│   ├── sos_gui.py
│   ├── sos.py
│   ├── model_behaviour.py
│   └── amadeos.py
├── simulation-on-windows.bat
├── simulation-on-unix.sh
└── model-Fri, 15 Jul 2016 08_39_57 GMT.xml
```

The simulation can be started by clicking on simulation-on-windows.bat or simulation-on-unix.sh depending on the platform of execution

Run simulation



Registry Host : Port

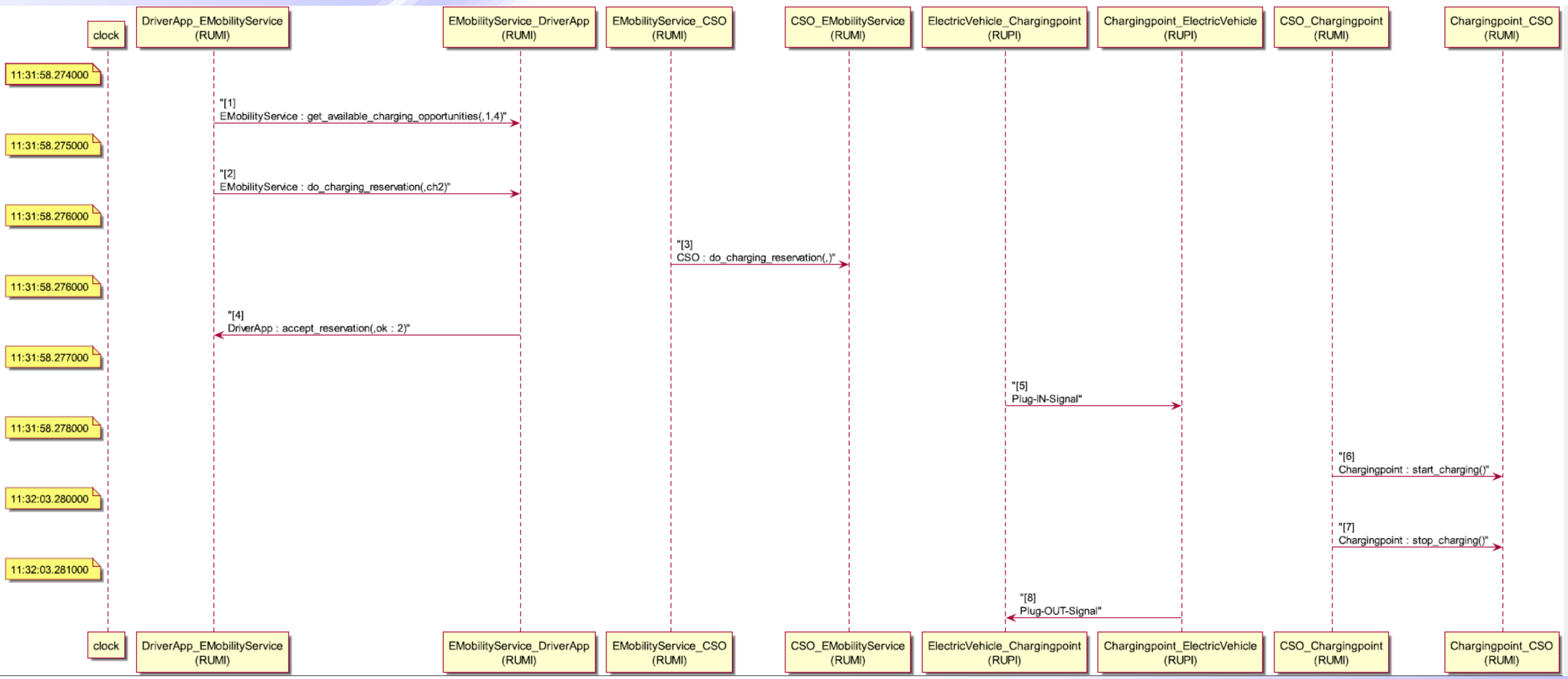
localhost:50000

☑ Create registry on this machine?

Select the SYSTEMS to start on this machine

☑ All_CSs / DriverApp
☑ All_CSs / EMobilityService
☑ All_CSs / LMO
☑ All_CSs / Chargingpoint
☑ All_CSs / ElectricVehicle
☑ All_CSs / CSO

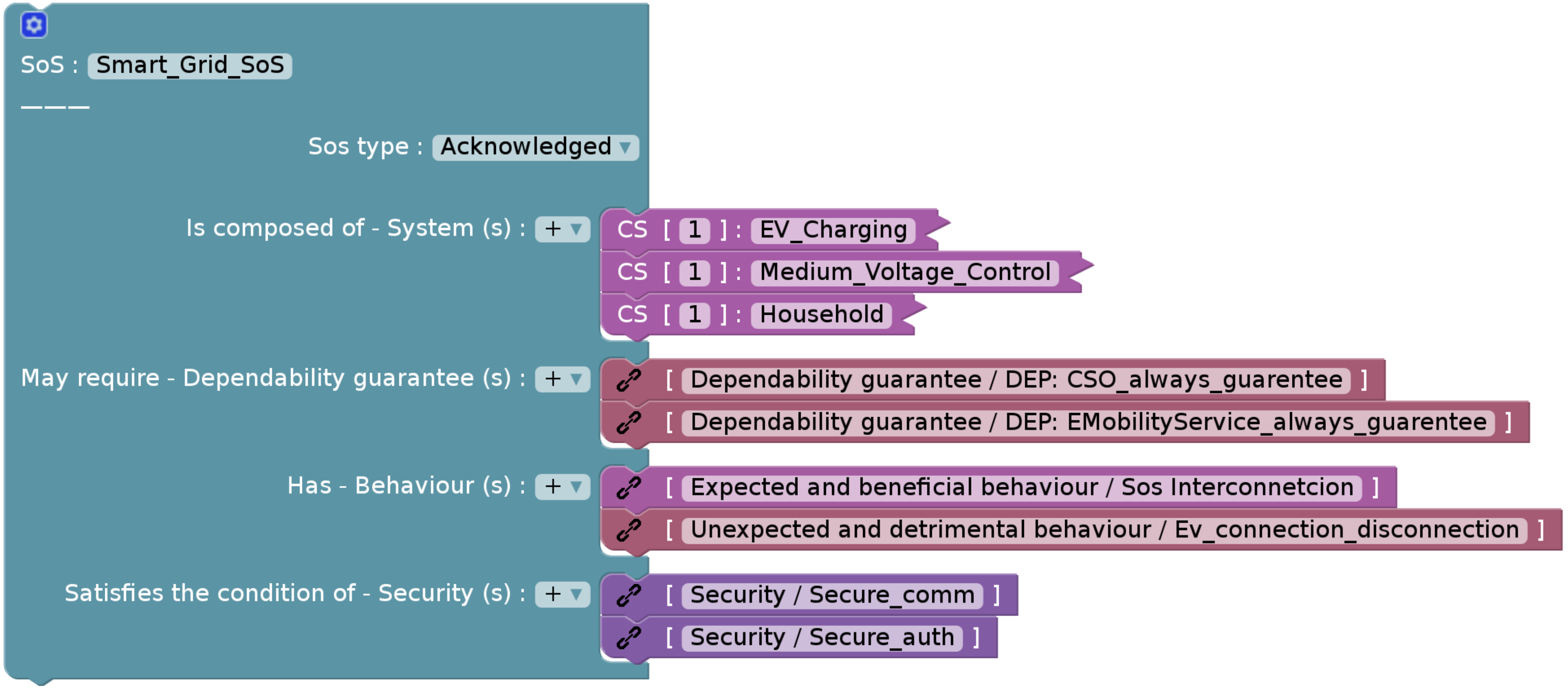
Example simulation result with timestamp


clock
DriverApp_EMobilityService (RUMI)
EMobilityService_DriverApp (RUMI)
EMobilityService_CSO (RUMI)
CSO_EMobilityService (RUMI)
ElectricVehicle_Chargingpoint (RUPI)
Chargingpoint_ElectricVehicle (RUPI)
CSO_Chargingpoint (RUMI)
Chargingpoint_CSO (RUMI)
11:31:58.274000
"[1] EMobilityService : get_available_charging_opportunities(,1,4)"
11:31:58.275000
"[2] EMobilityService : do_charging_reservation(,ch2)"
11:31:58.276000
"[3] CSO : do_charging_reservation(,)"
11:31:58.276000
"[4] DriverApp : accept_reservation(,ok : 2)"
11:31:58.277000
"[5] Plug-IN-Signal"
11:31:58.278000
"[6] Chargingpoint : start_charging()"
11:32:03.280000
"[7] Chargingpoint : stop_charging()"
11:32:03.281000
"[8] Plug-OUT-Signal"

The result of simulation is found in the "result.seq" file, which is the run-time sequence diagram with timestamp.
It can be visualized with the Atom PlantUML viewer.
This result should be compliant with the <u>sequence diagram that was designed.</u>

Model querying

Search inside a large model !



SoS : Smart_Grid_SoS

Sos type : Acknowledged

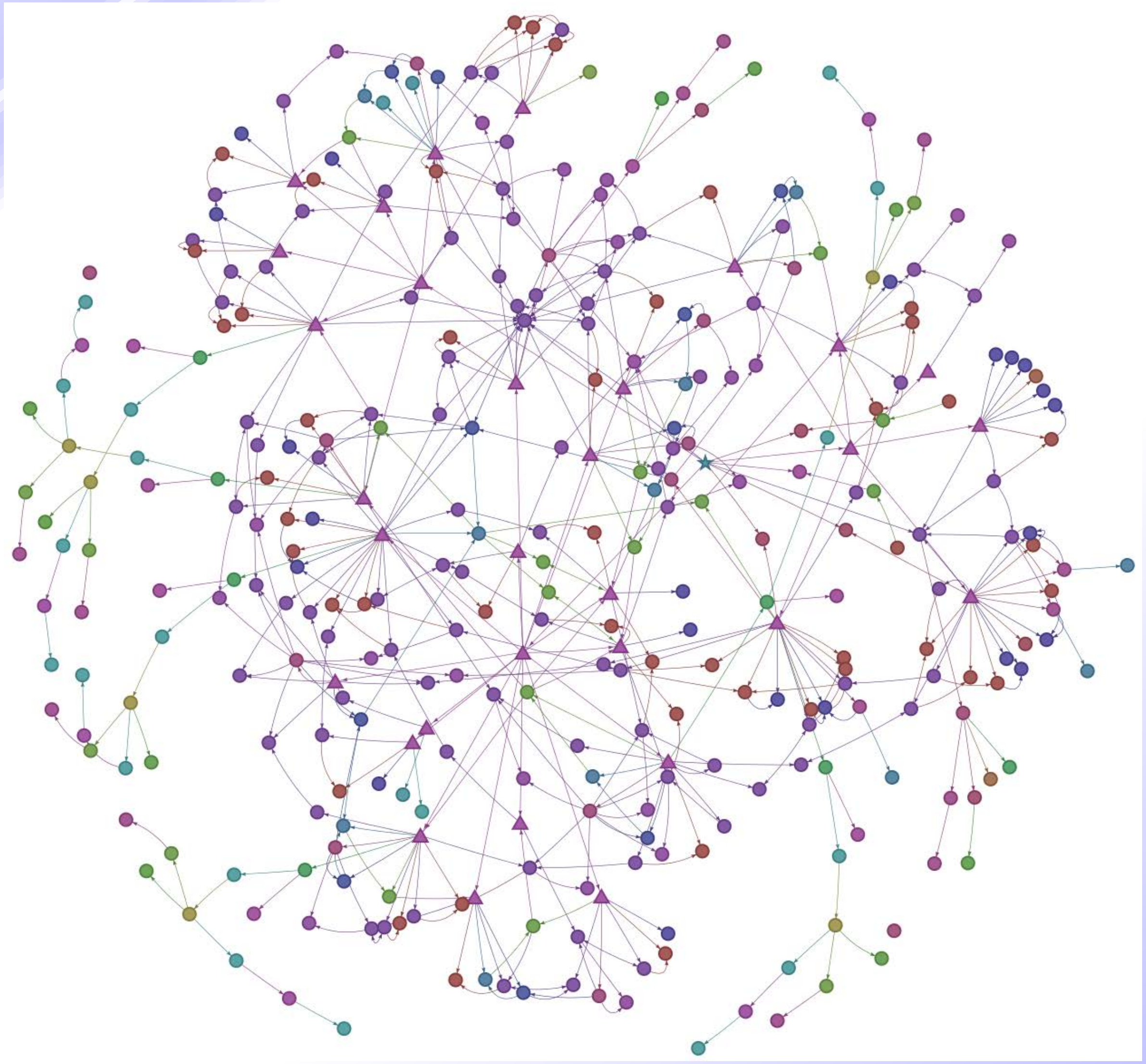
Is composed of - System (s) :
- CS [1] : EV_Charging
- CS [1] : Medium_Voltage_Control
- CS [1] : Household

May require - Dependability guarantee (s) :
- [Dependability guarantee / DEP: CSO_always_guarentee]
- [Dependability guarantee / DEP: EMobilityService_always_guarentee]

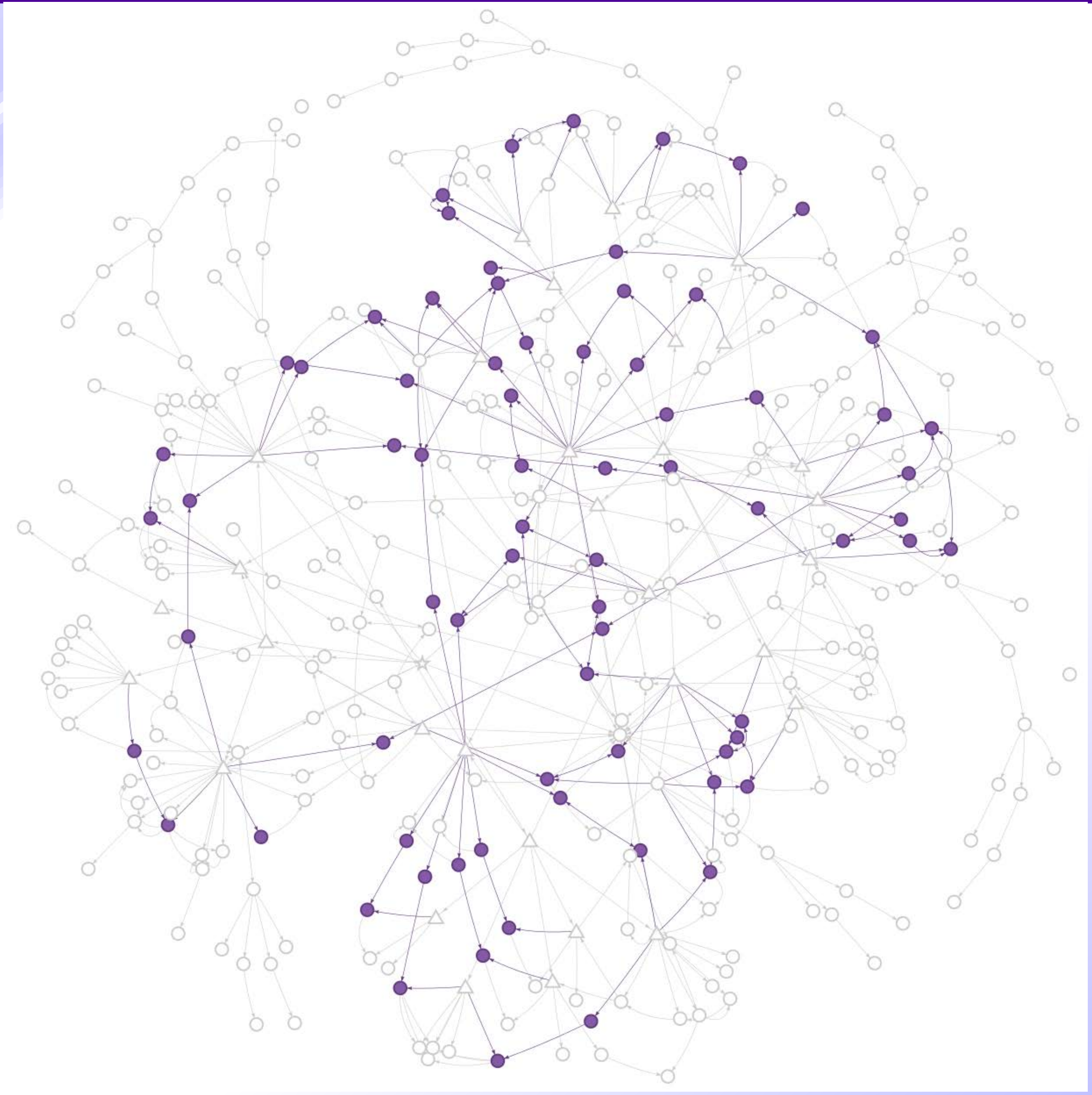
Has - Behaviour (s) :
- [Expected and beneficial behaviour / Sos Interconnetcion]
- [Unexpected and detrimental behaviour / Ev_connection_disconnection]

Satisfies the condition of - Security (s) :
- [Security / Secure_comm]
- [Security / Secure_auth]

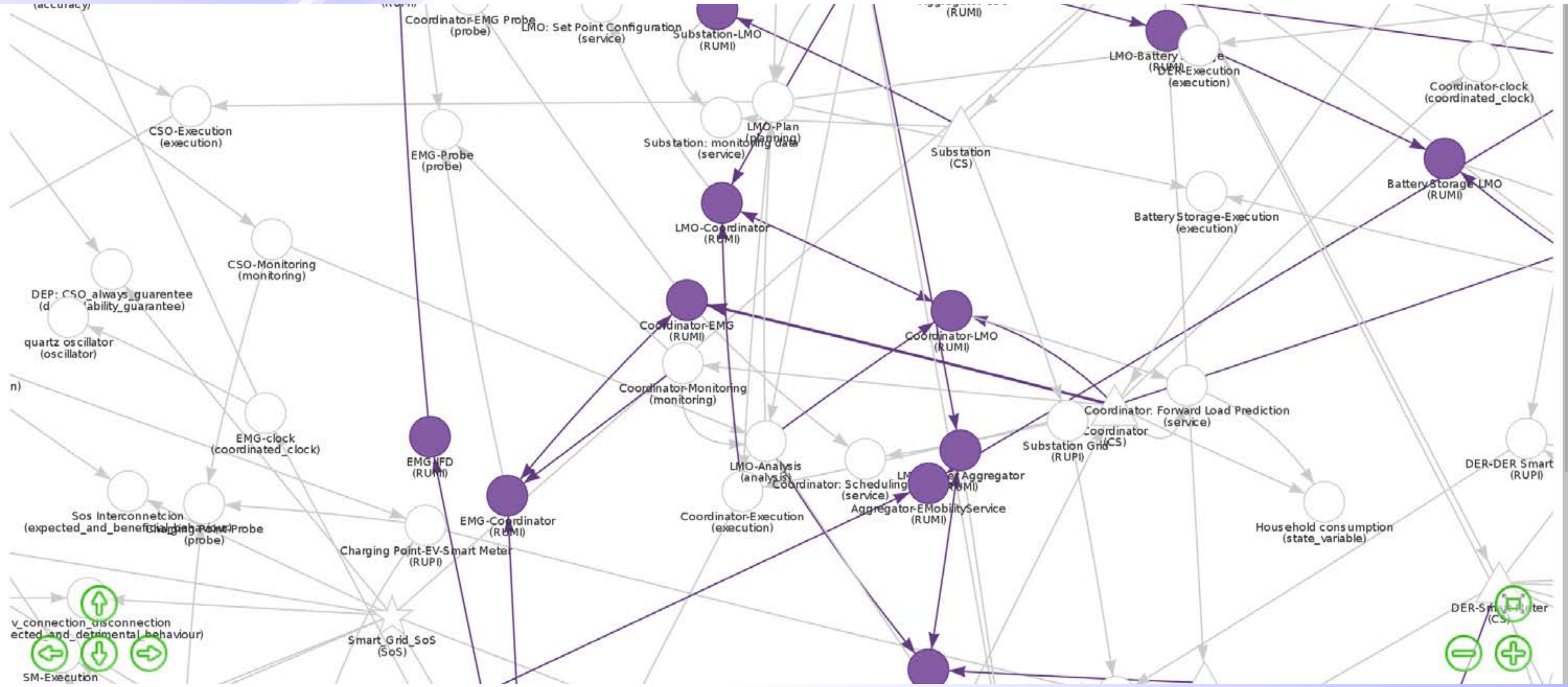
Model query - <u>return true;</u> (i.e. get all blocks)



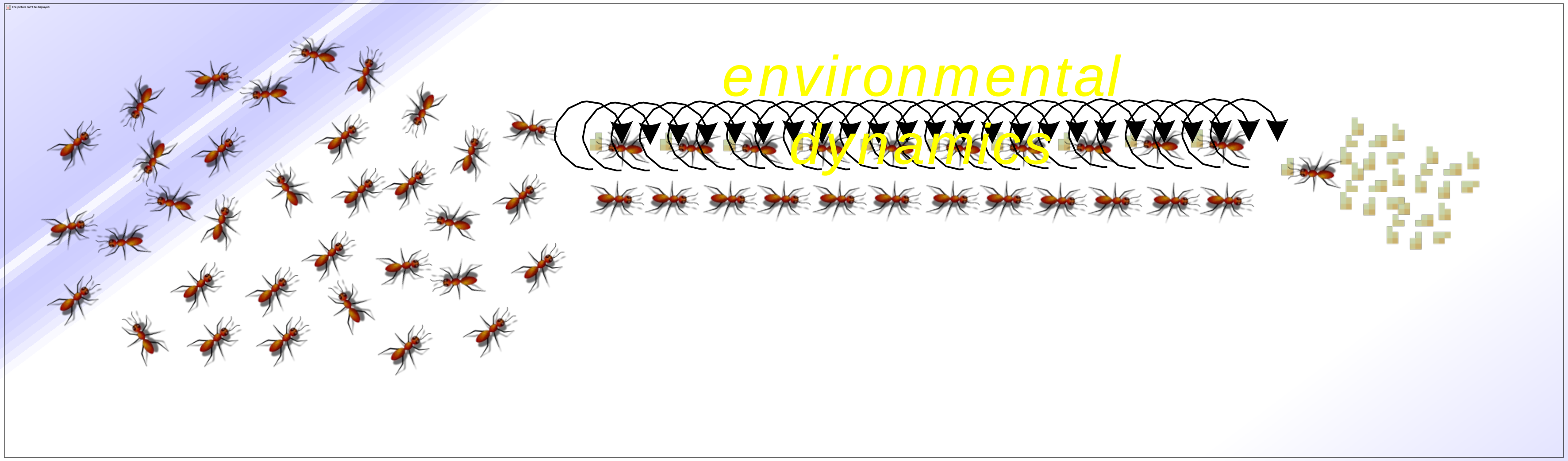
Model query - <u>return block.of type == 'RUMI';</u>



Model query - zoomed results

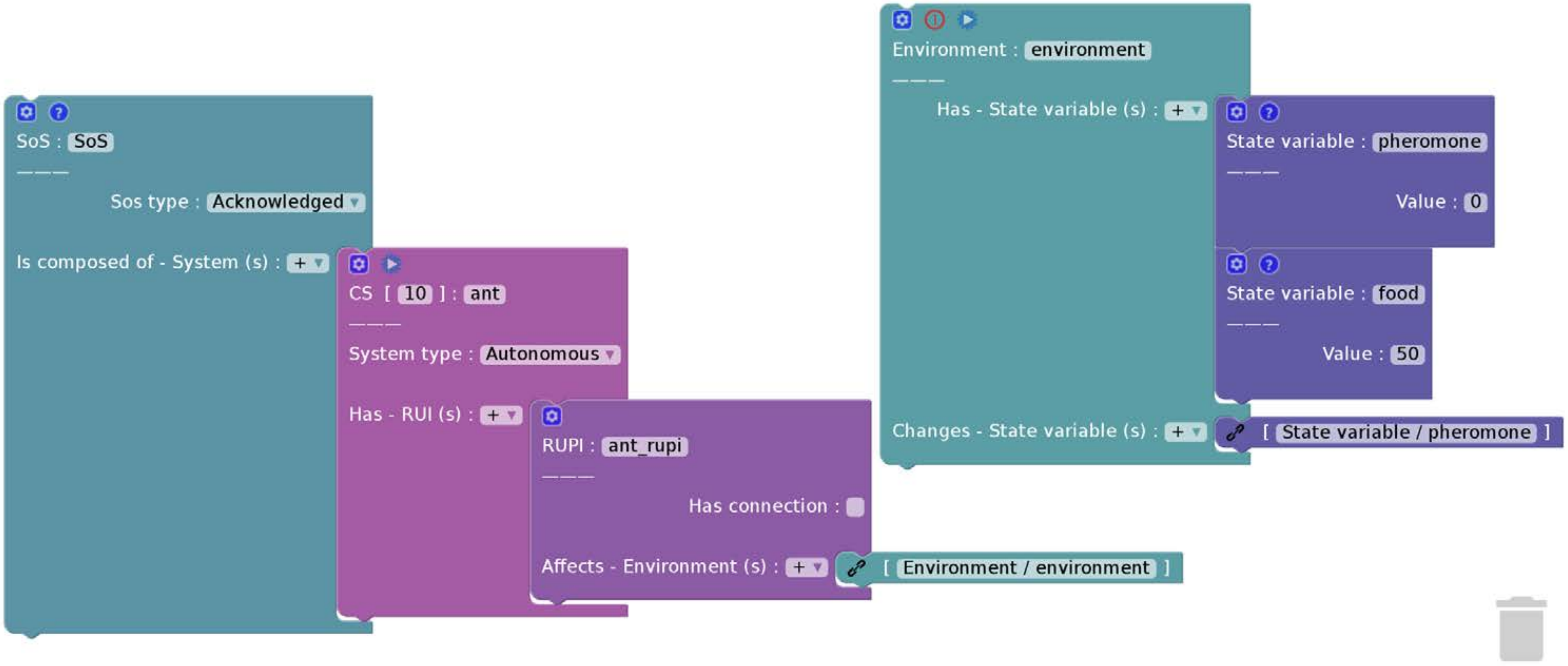

Coordinator-EMG Probe (probe)
LMO: Set Point Configuration (service)
Substation-LMO (RUMI)
LMO-Battery (RUMI)
DER-Execution (execution)
Coordinator-clock (coordinated_clock)
CSO-Execution (execution)
EMG-Probe (probe)
LMO-Plan (planning)
Substation: monitoring data (service)
Substation (CS)
Battery Storage LMO (RUMI)
LMO-Coordinator (RUMI)
Battery Storage-Execution (execution)
CSO-Monitoring (monitoring)
DEP: CSO_always_guarentee
quartz oscillator (oscillator)
Coordinator-EMG (RUMI)
Coordinator-LMO (RUMI)
Coordinator-Monitoring (monitoring)
Coordinator: Forward Load Prediction (service)
Coordinator (CS)
EMG-clock (coordinated_clock)
EMG FD (RUMI)
LMO-Analysis (analysis)
Coordinator: Scheduling (service)
Aggregator (RUMI)
Aggregator-EMobilityService (RUMI)
Substation Grid (RUPI)
DER-DER Smart (RUPI)
Sos Interconnetcion (expected_and_beneficial_behaviour)
Charging Point-Probe (probe)
EMG-Coordinator (RUMI)
Coordinator-Execution (execution)
Household consumption (state_variable)
Charging Point-EV-Smart Meter (RUPI)
v_connection_disconnection
ected_and_detrimental_behaviour)
Smart_Grid_SoS (SoS)
DER-Smart Meter (CS)
SM-Execution

Stigmergic Channels



- Ants find food and build/enforce trail by leaving traces (*pheromone*) in environment on way back.
- In case food source depleted,
  - ants stop leaving traces,
  - The environment evaporates traces autonomously
    ⇒ *environmental dynamics*.
  - the trail disappears.

Ants model



SoS : SoS
Sos type : Acknowledged
Is composed of - System (s) :

CS [10] : ant
System type : Autonomous
Has - RUI (s) :

RUPI : ant_rupi
Has connection :
Affects - Environment (s) : [Environment / environment]

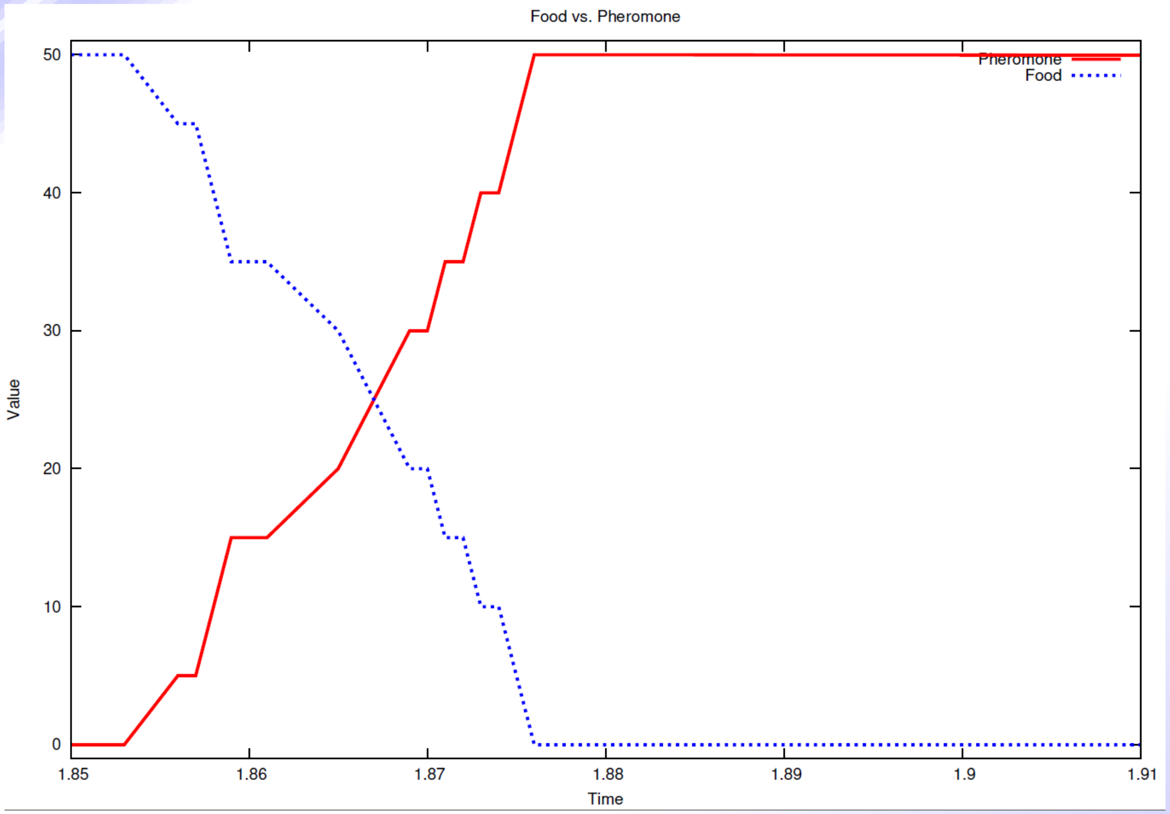
Environment : environment
Has - State variable (s) :
Changes - State variable (s) : [State variable / pheromone]

State variable : pheromone
Value : 0

State variable : food
Value : 50

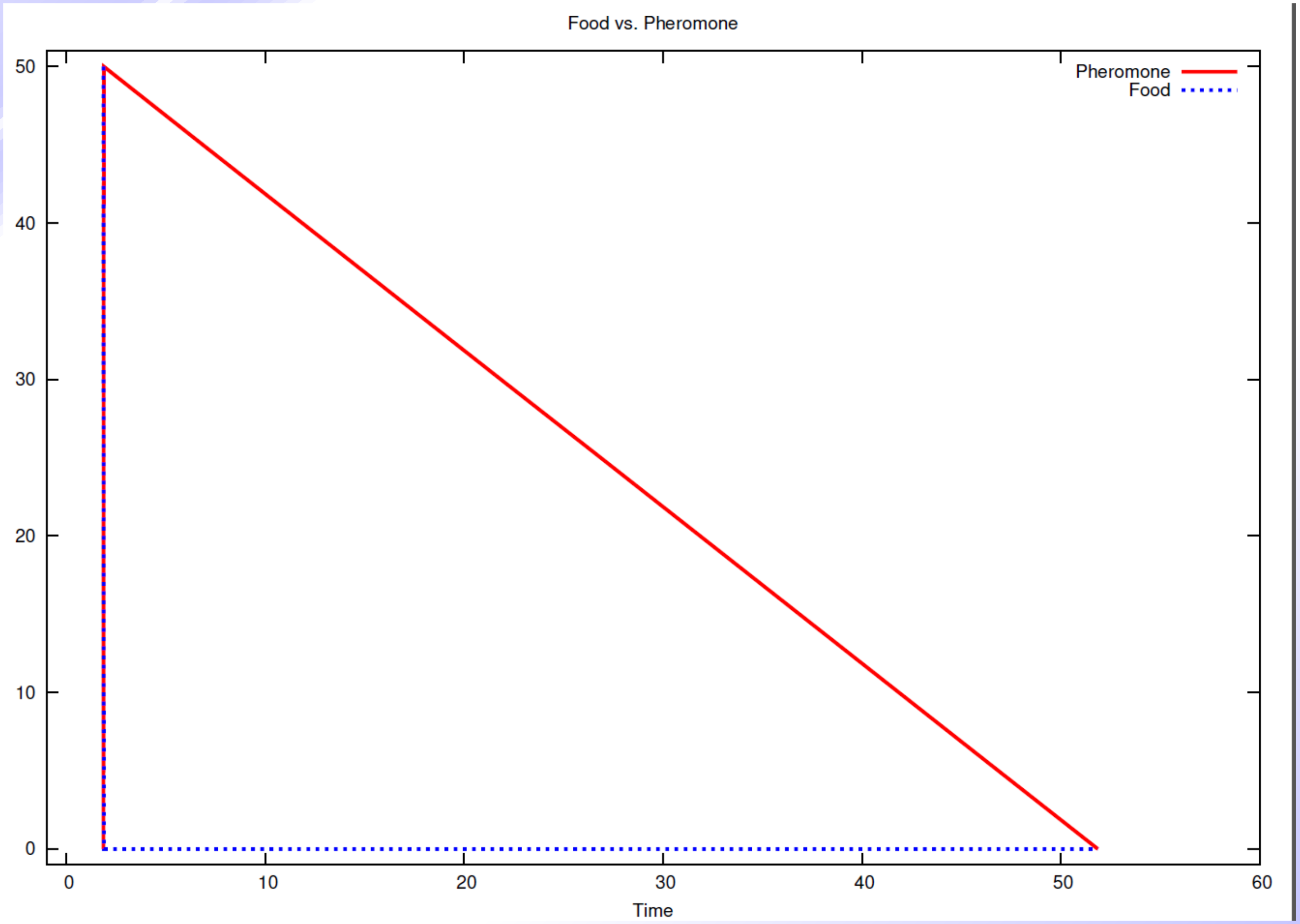
Please note the cardinality (of ants) and singleton (of environment) in the model !

Pheromone vs. Food simulation - 1



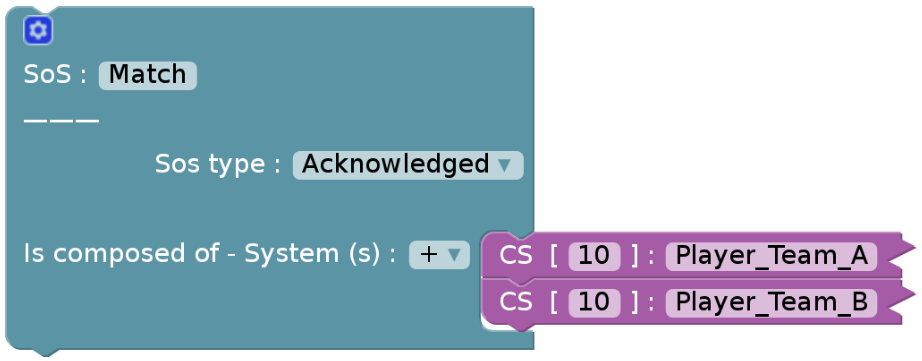
Change in pheromone and food as ants find food

Pheromone vs. Food simulation - 2



Change in pheromone after food becomes zero and pheromone is depleted by the environment

Football model (with no strategy…)



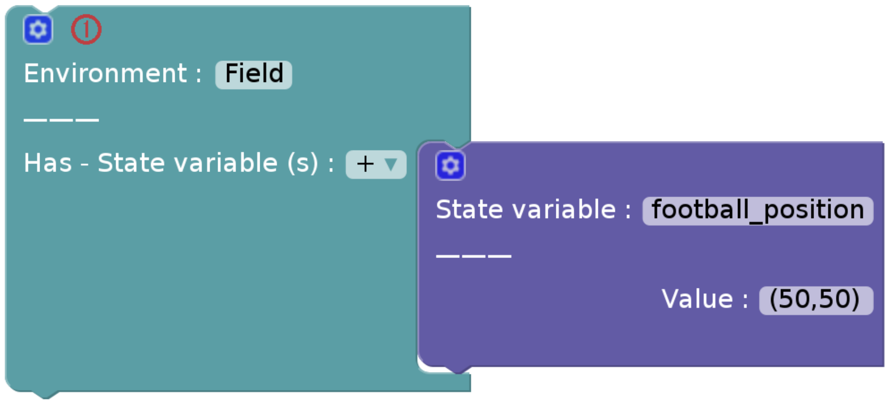
SoS : Match

Sos type : Acknowledged

Is composed of - System (s) : +

CS [10] : Player_Team_A

CS [10] : Player_Team_B



Environment : Field

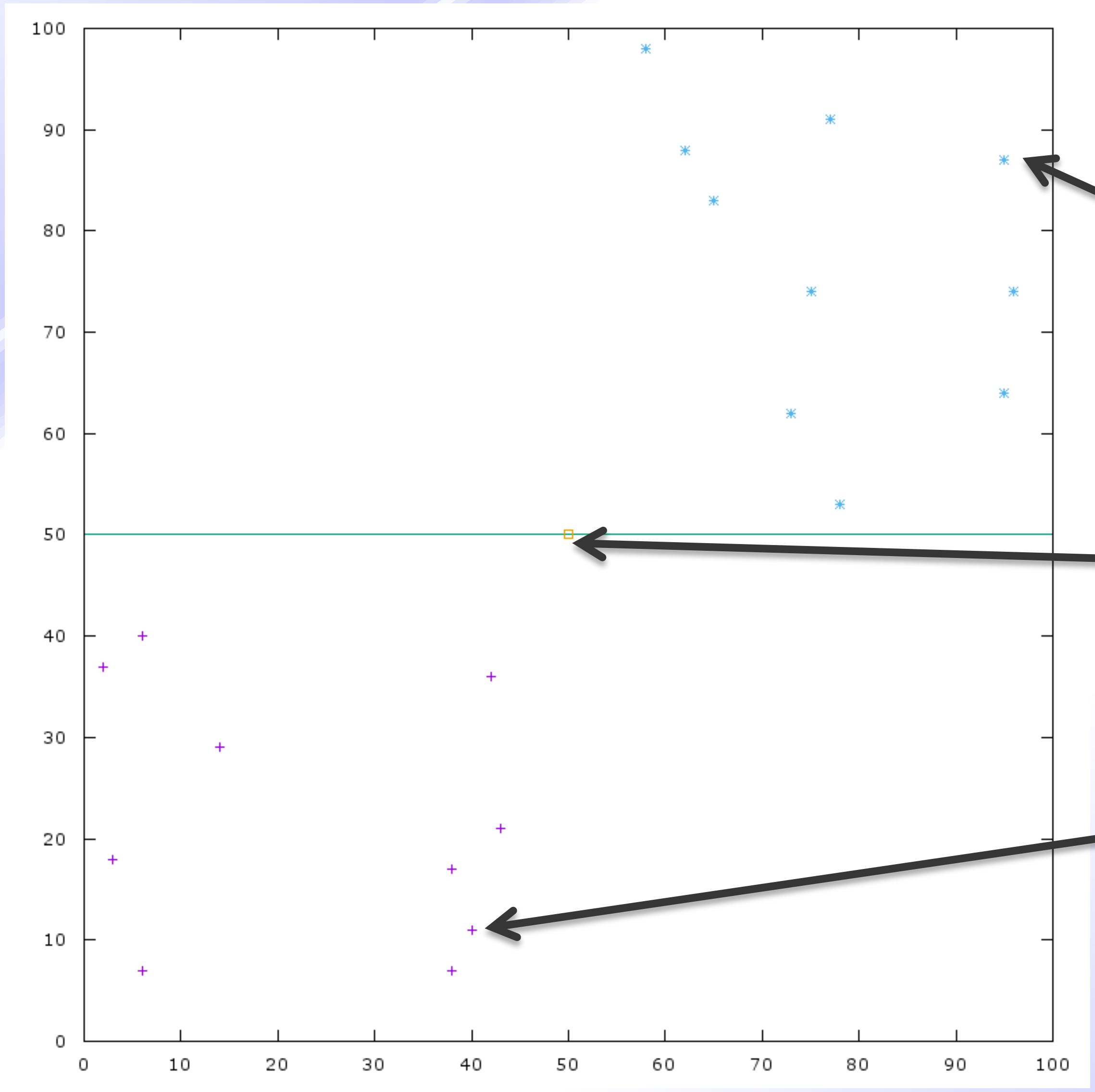
Has - State variable (s) : +

State variable : football_position

Value : (50,50)

In this model, the position of players is random

Food ball simulation results - At 0th second

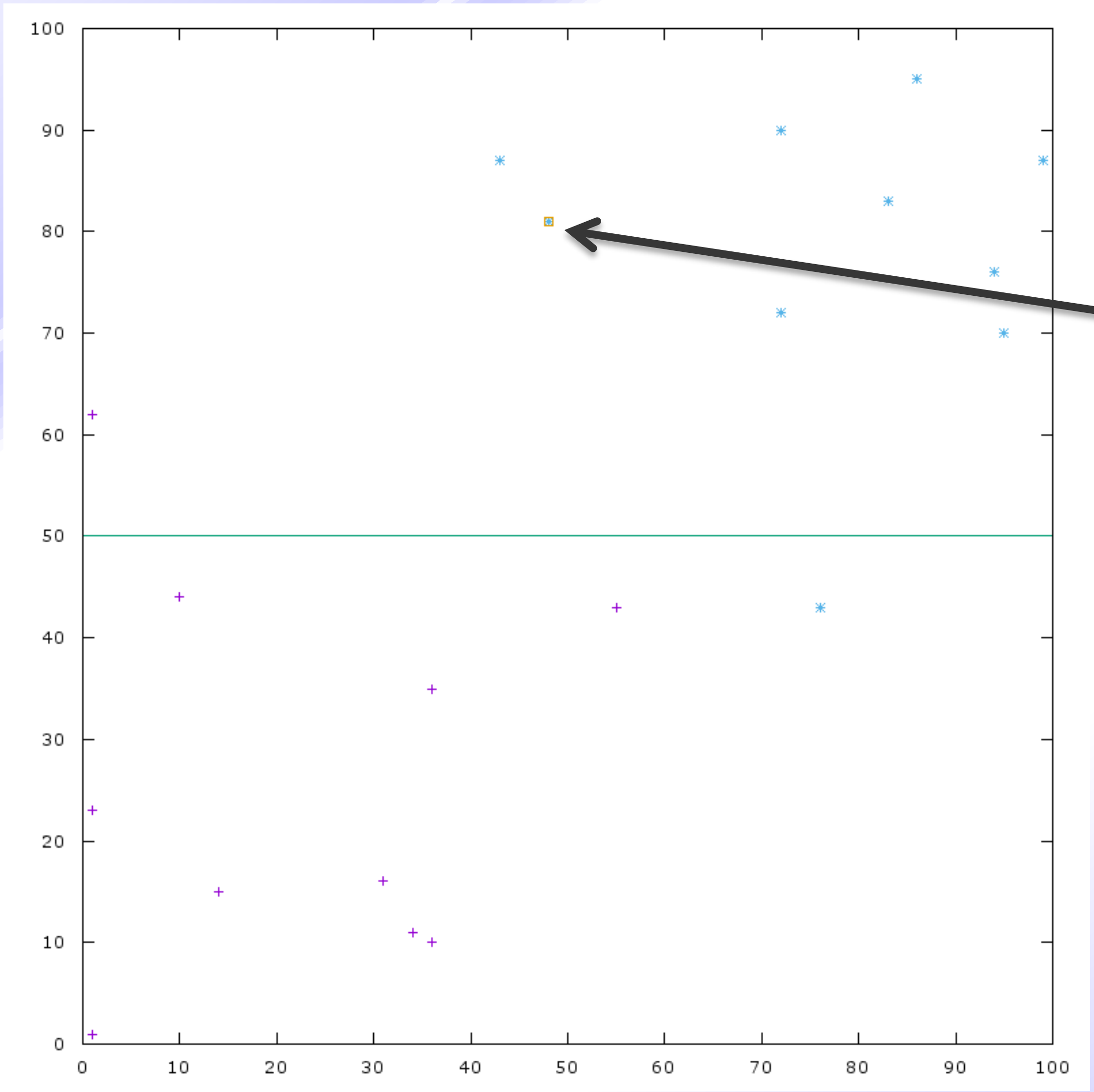


Team A

Football at position (50,50)

Team B

Food ball simulation results - At 25th second



Foot ball is with a team A player

References

- "Cyber-Physical Systems of Systems - Foundations, a conceptual model and some derivations: the AMADEOS legacy", edited by A. Bondavalli, S. Bouchenak, H. Kopetz, to appear in LNCS State-of-the-Art Surveys – Springer.
- AMADEOS SoS Profile:
  - https://github.com/arun-babu/amadeos-project
- Blockly for SoS:
  - http://blockly4sos.resiltech.com
- Blockly for SoS - User Guide
  - http://blockly4sos.resiltech.com/user-guide.pdf

Known bugs

- When requirements are matched to a component:
  - if the component is deleted, requirements are not updated
  - component cannot be modified.
- Not clear how to add constraints on a CS name and number of instances: documentation is not adequate.
- When constraints are deleted on a black block (constraint not satisfied) : the block remains black.
- Some visualization issues in the text viewer, if text is too big