The AMADEOS Supporting facility tool

Paolo Lollini
lollini@unifi.it,
University of Florence, Italy

Arun Babu
arun.babu@resiltech.com,
ResilTech srl, Italy

Outline

- Goals
- Google Blockly
- Blockly4SoS Customization
- Examples of Simulations

Goal

- To build a tool …
  - To create object diagrams based on a UML/SysML profile
  - Which is simple, intuitive, and straight forward
  - Which supports code generation
  - And is fast !

Supporting facility tool for AMADEOS

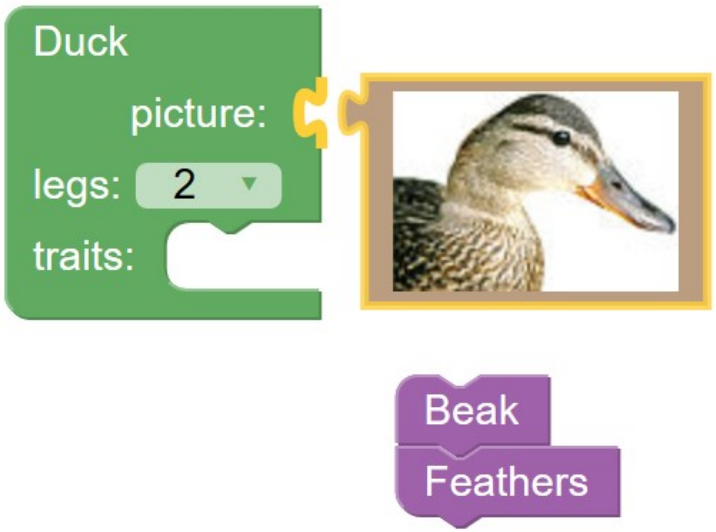
Things we considered

1. A generic SoS designer
2. Should have simple and intuitive user interface
3. In real life, SoS designers are not CS / OO / SysML experts
4. Reduce cost – By requiring less training/support for designers
5. Reduce lines in design (to avoid spaghetti diagrams)
6. Warn designers when they make mistakes
7. Rapid modeling & simulation

Where should the designer spend the effort ?

On modeling. Not on anything else !

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

1. Most basic example:
   - https://developers.google.com/blockly/

2. Blockly games examples:
   - https://blockly-games.appspot.com/

3. Real world examples using Blockly based ideas:
   - Fashion – https://www.madewithcode.com/projects/fashion
   - Stock market – https://bot.binary.com/bot.html
   - Andorid – appinventor.mit.edu/explore/designer-blocks.html
   - Electronics:
     - Codebug – https://www.codebug.org.uk/create/codebug/new/
     - Ardublockly – http://ardublockly.embeddedlog.com/

Blockly customization

Blockly has been customized to make it more UML/SysML like. E.g.:

1. Support constraints
2. Support behaviors
3. Support links
4. Support viewpoints
5. Support intuitive maximize, collapse, and semi-collapse
6. Support requirements management
7. Guide user to select compatible blocks
8. Blockly to PlantUML conversion
9. Blockly to Python conversion (different from the default one)
10. Blockly to Graph conversion and Graph querying
11. Support sequence diagrams
12. Custom minification of JavaScript for faster loading ...

AMADEOS supporting facility tool

- **A tool to:**
  - **Model**
  - **Validate**
  - **Query and**
  - **Simulate**

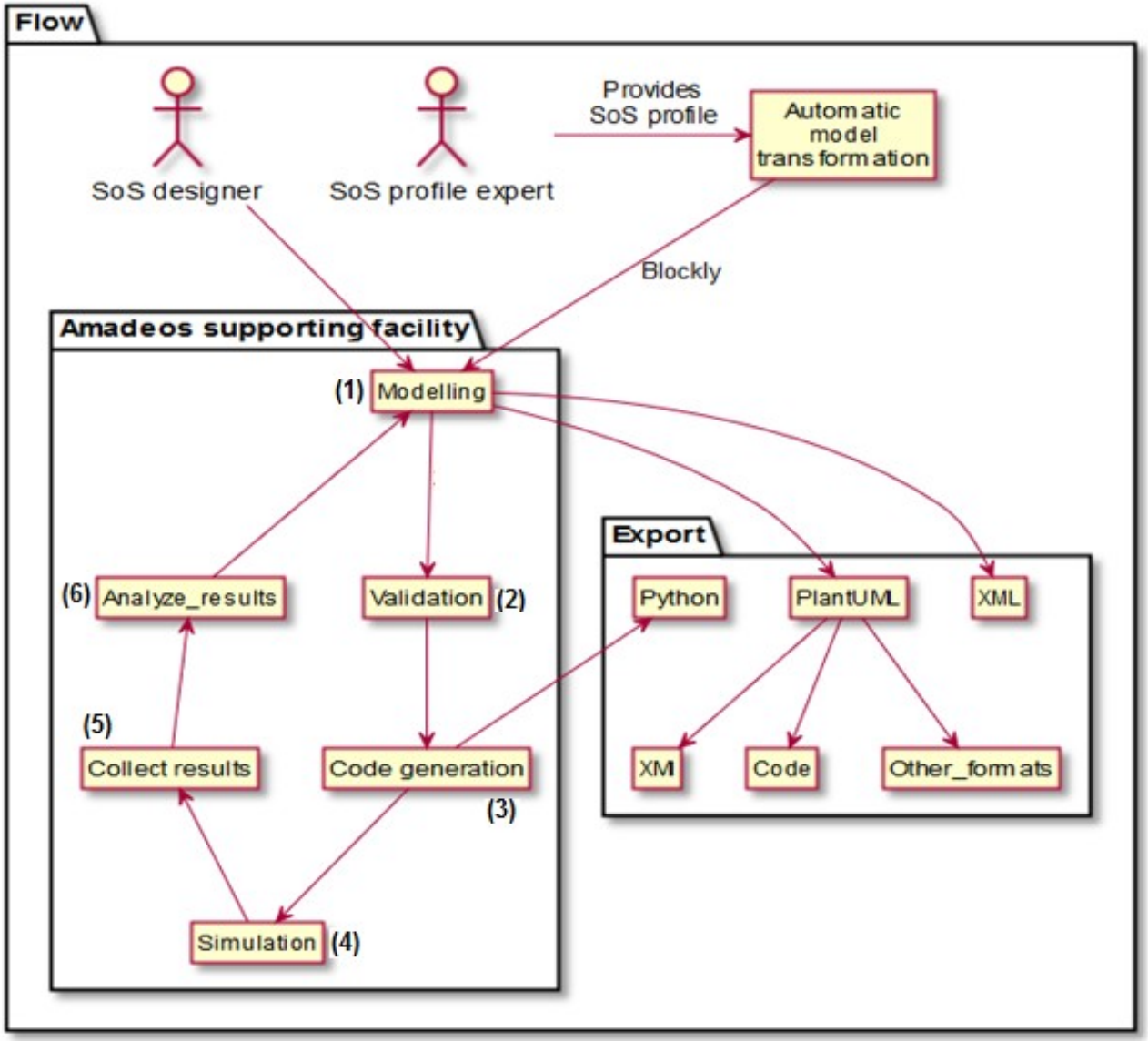
A system-of-systems

- Link to the homepage of tool
  - **http://blockly4sos.resiltech.com**
  - **Though, any modern browser would work fine,**
  - **Firefox is faster !**

SoS profile's integration within Blockly

- Blockly has been customized to be used for SoS modelling by **importing** the SoS SysML profile of AMADEOS.
- The SoS designer need not have any knowledge of SysML/UML;
  - the only prerequisite is high-level knowledge about the profile and knowledge of the supporting facility tool.

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

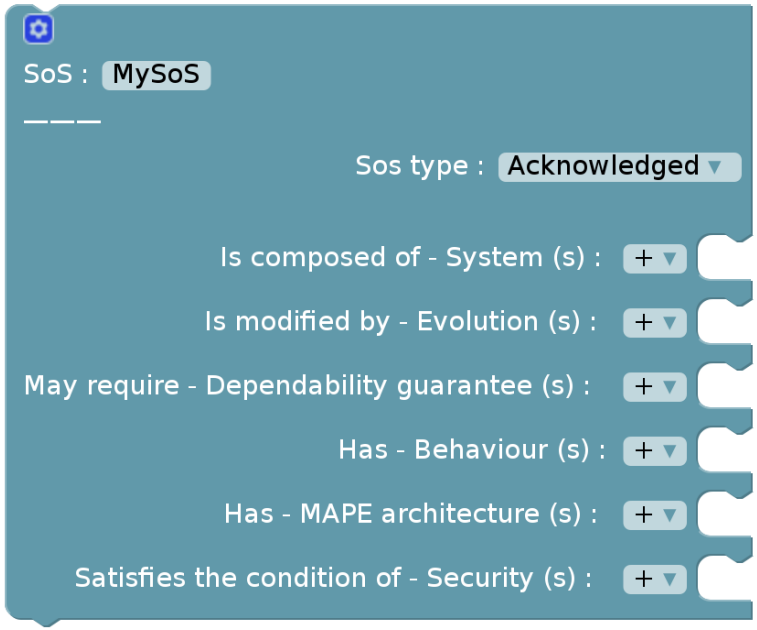
Model transformation

PlantUML

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



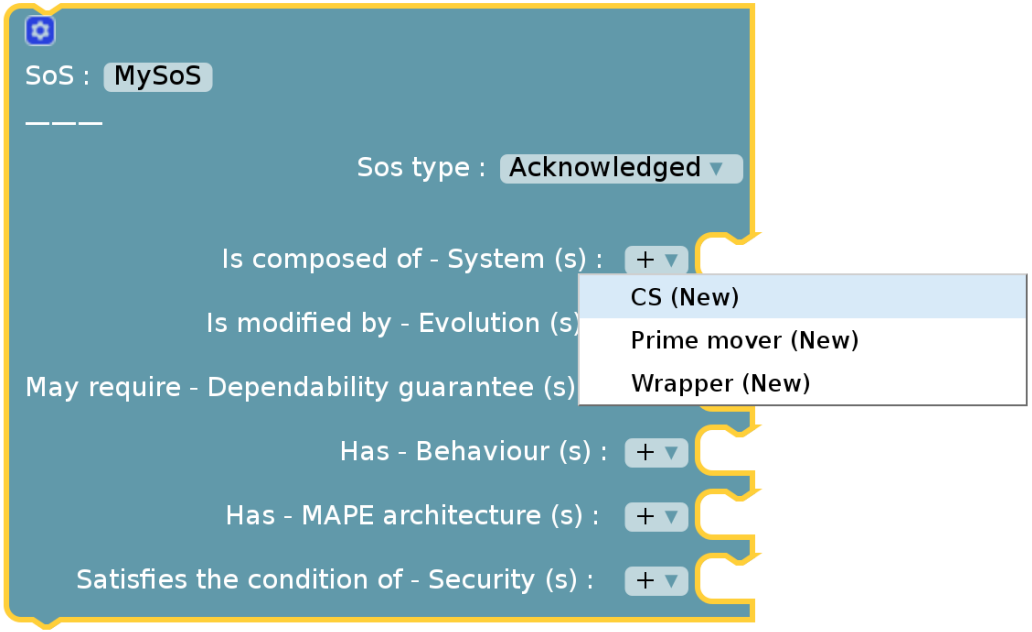
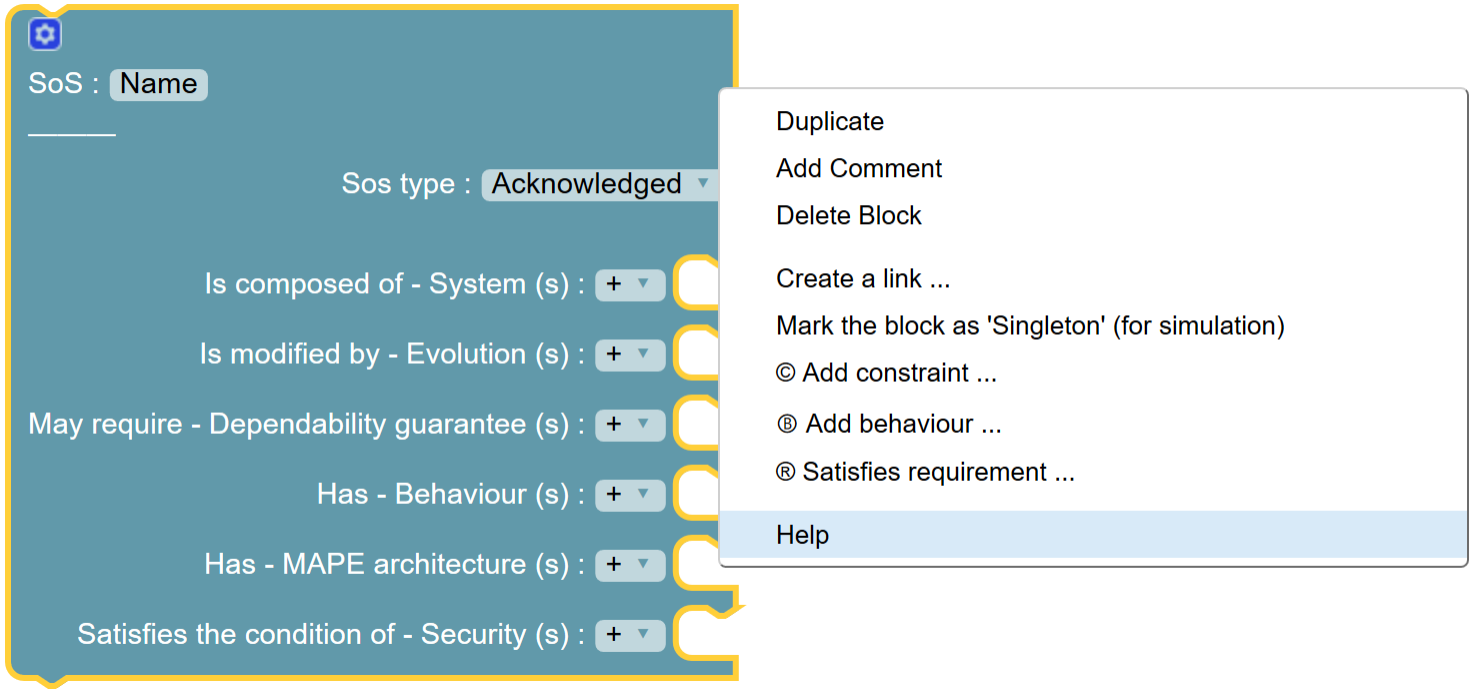


Figure: Add new blocks by clicking on the (+) drop-down/from left-hand side toolbox

Help/Glossary



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

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

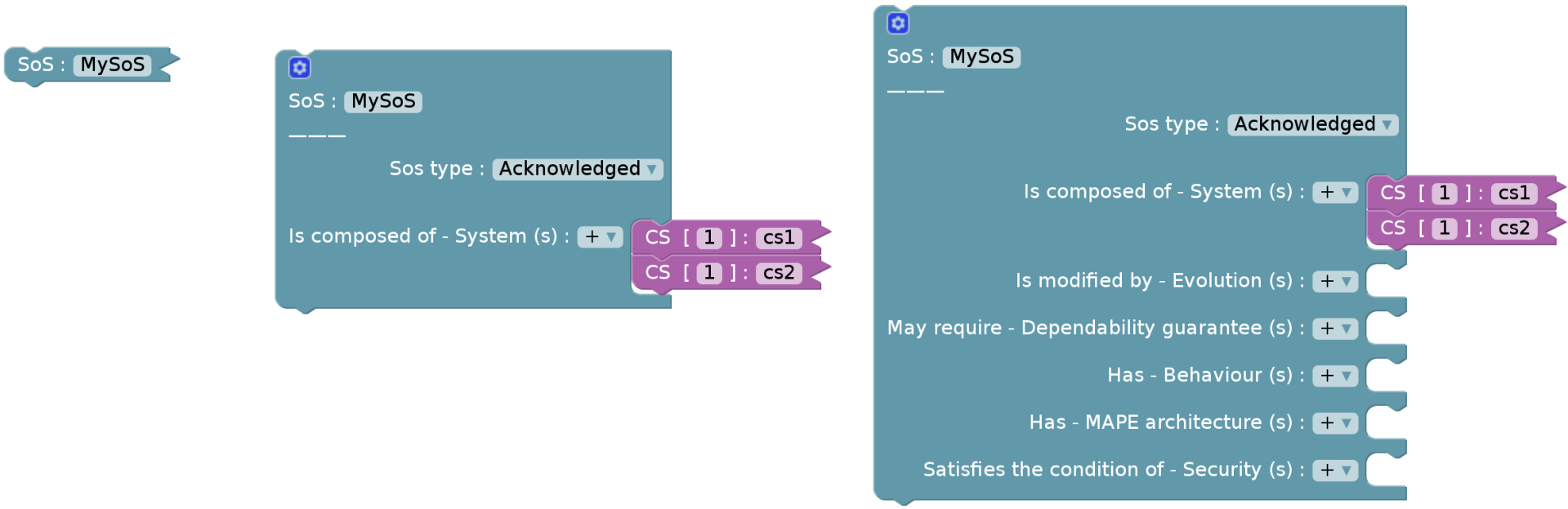
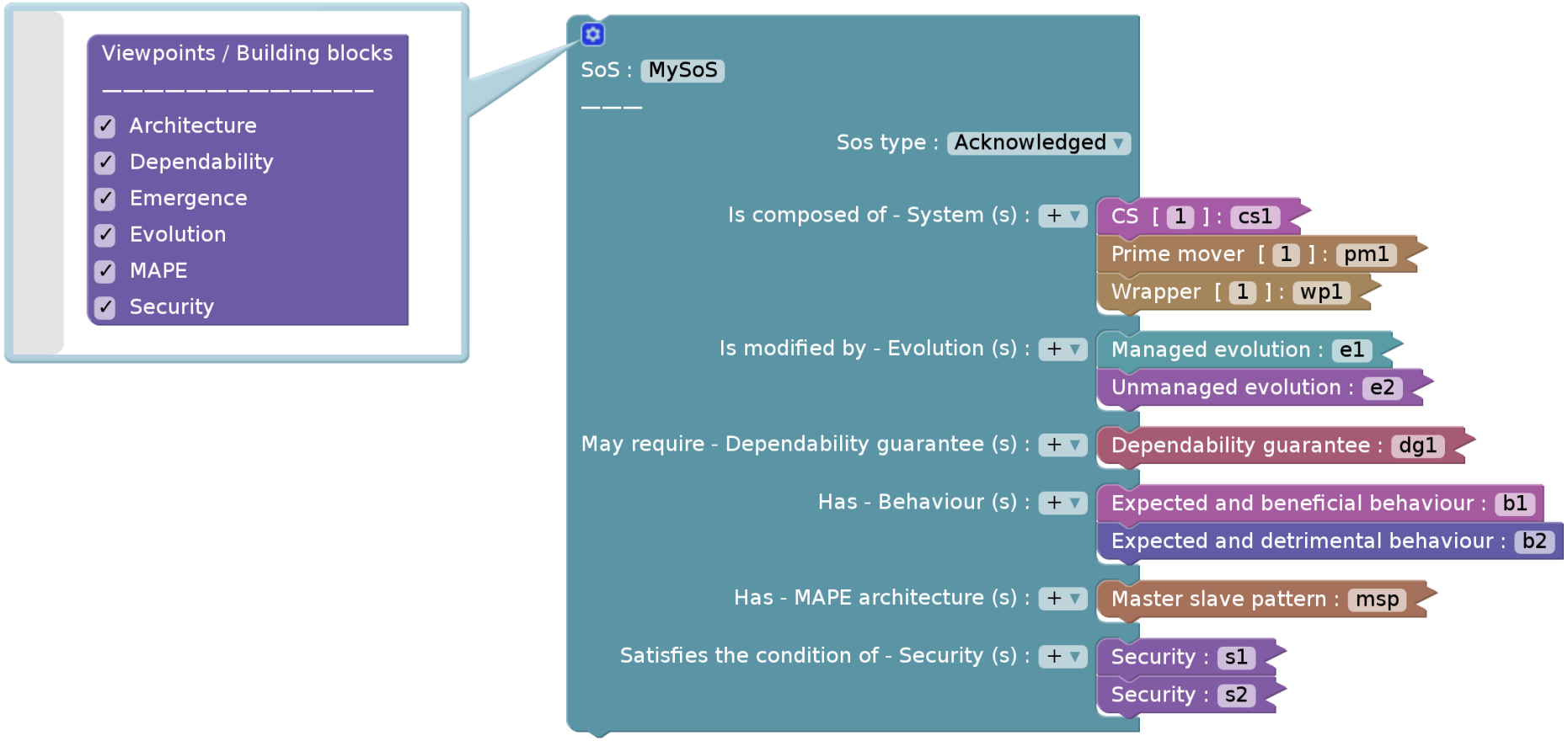
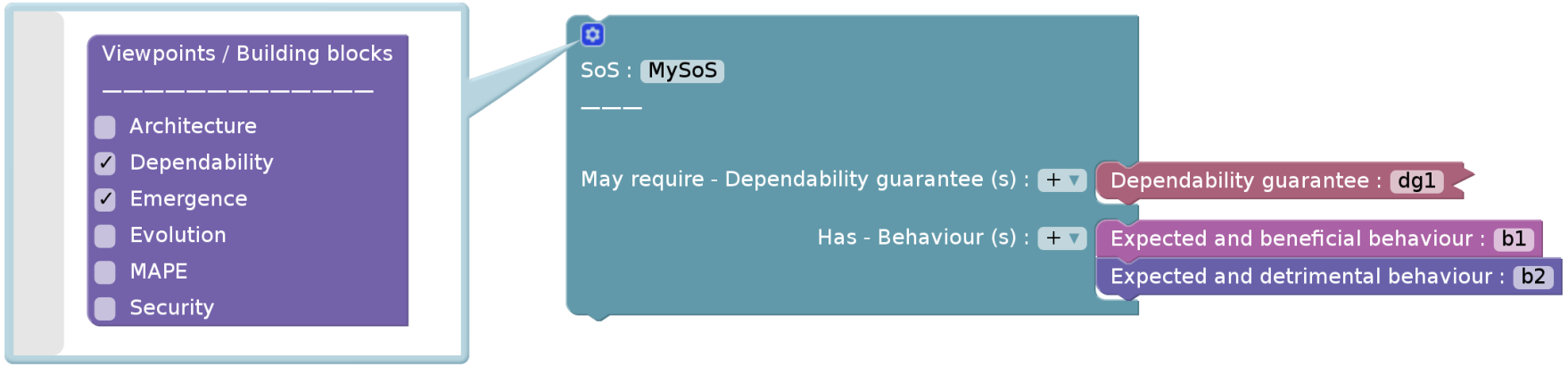


Figure: 3 ways of viewing a block - cycle between views by **double clicking** the block

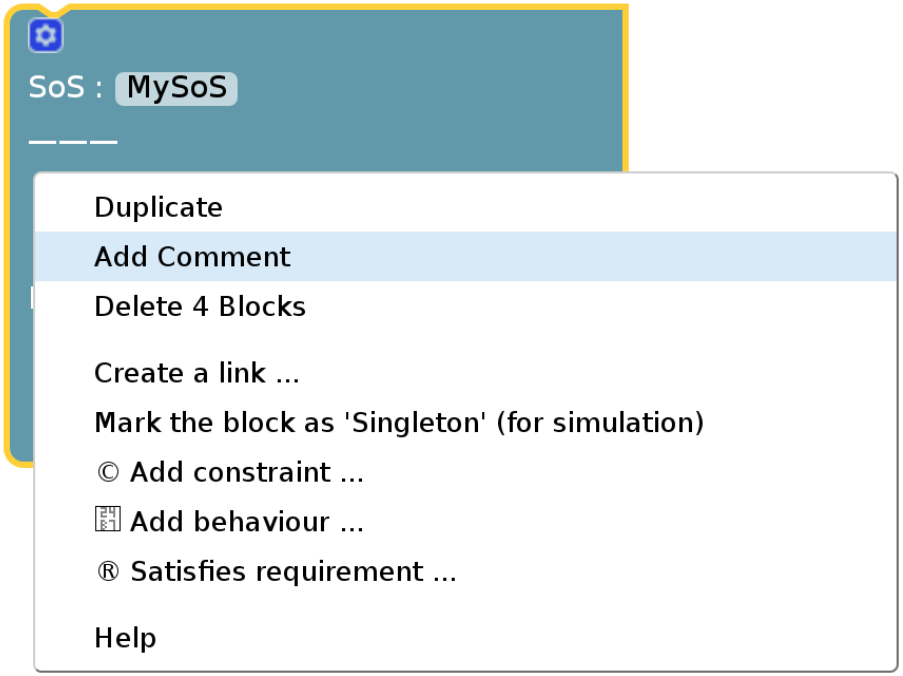
All viewpoints and building blocks of a block

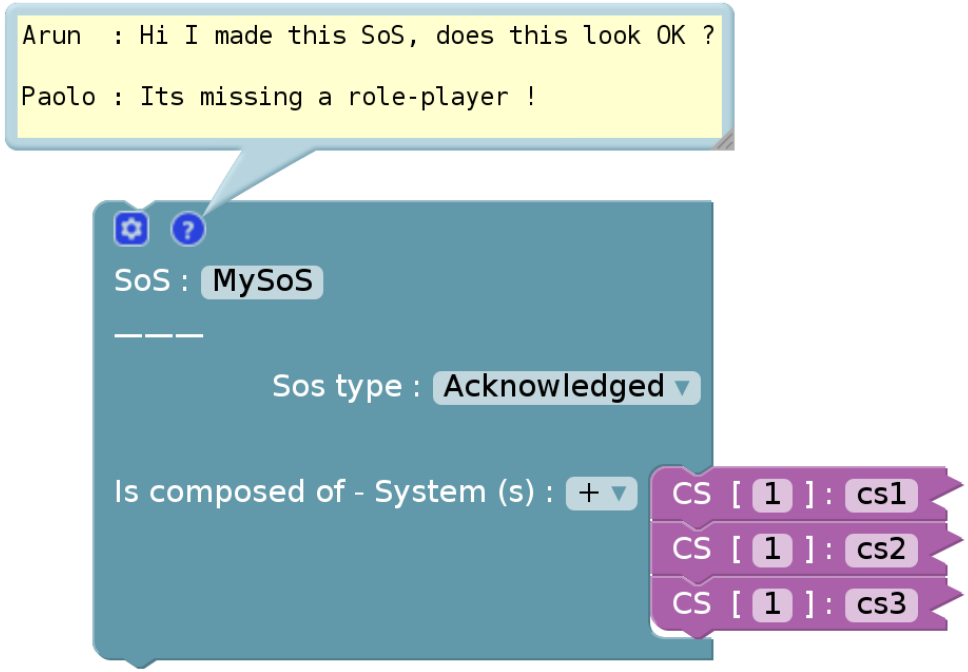


Filter some of the viewpoints/building blocks



Comment your design





Modularize the design by grouping

[BLOCKS]

Group

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

Group

Group : My_CSs

CS [1] : cs1

CS [1] : cs2

CS [1] : cs3

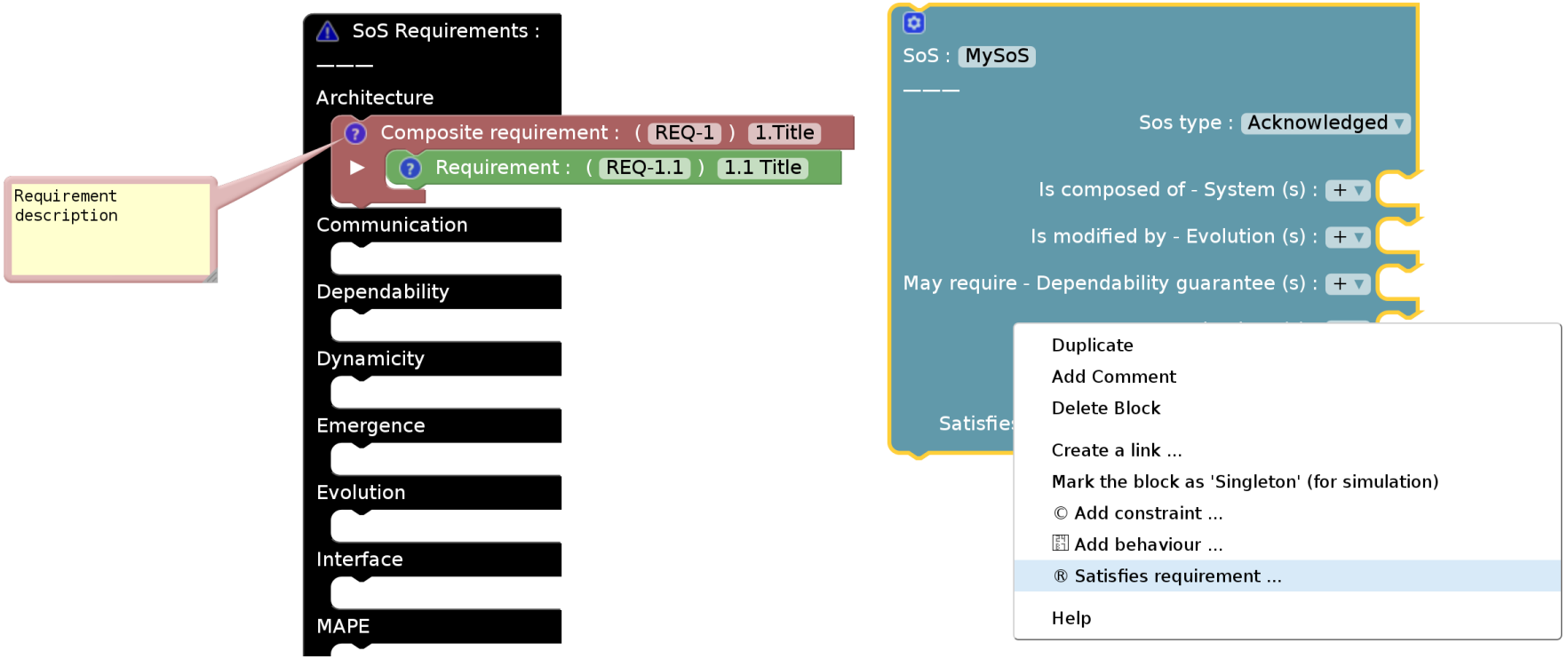
Group : My_PrimeMovers

Prime mover [1] : pm1

Prime mover [1] : pm2

Prime mover [1] : pm3

Manage requirements for each viewpoint



Model validation

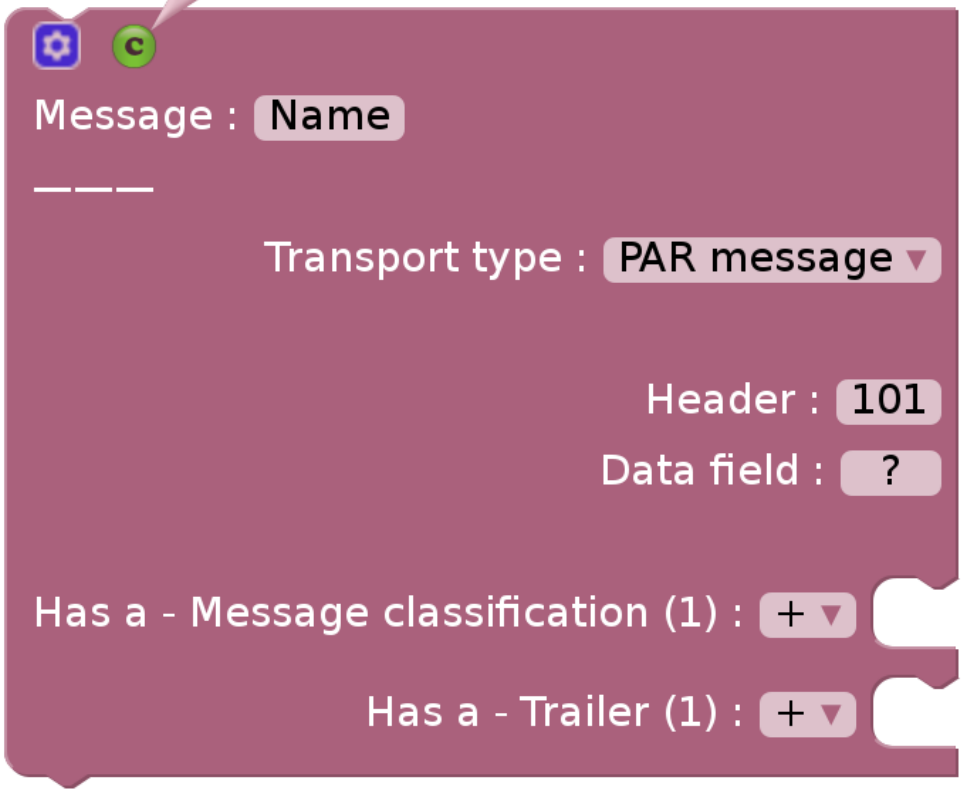
- **Blockly by default model validation by letting only compatible blocks to be connected with each other.**
- **User can add custom validation in JavaScript by using the below constraint functions:**

```
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")
```



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 : ?

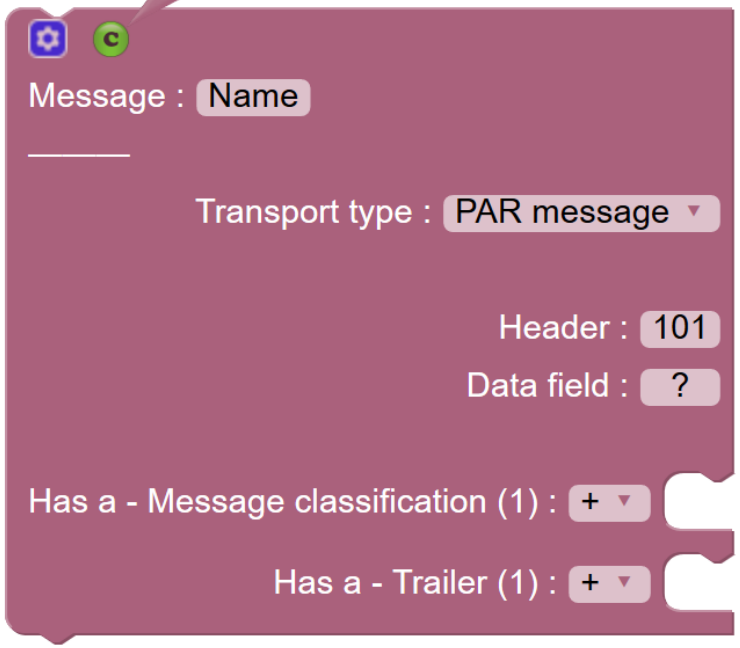
Has a - Message classification (1) : + ▾

Has a - Trailer (1) : + ▾

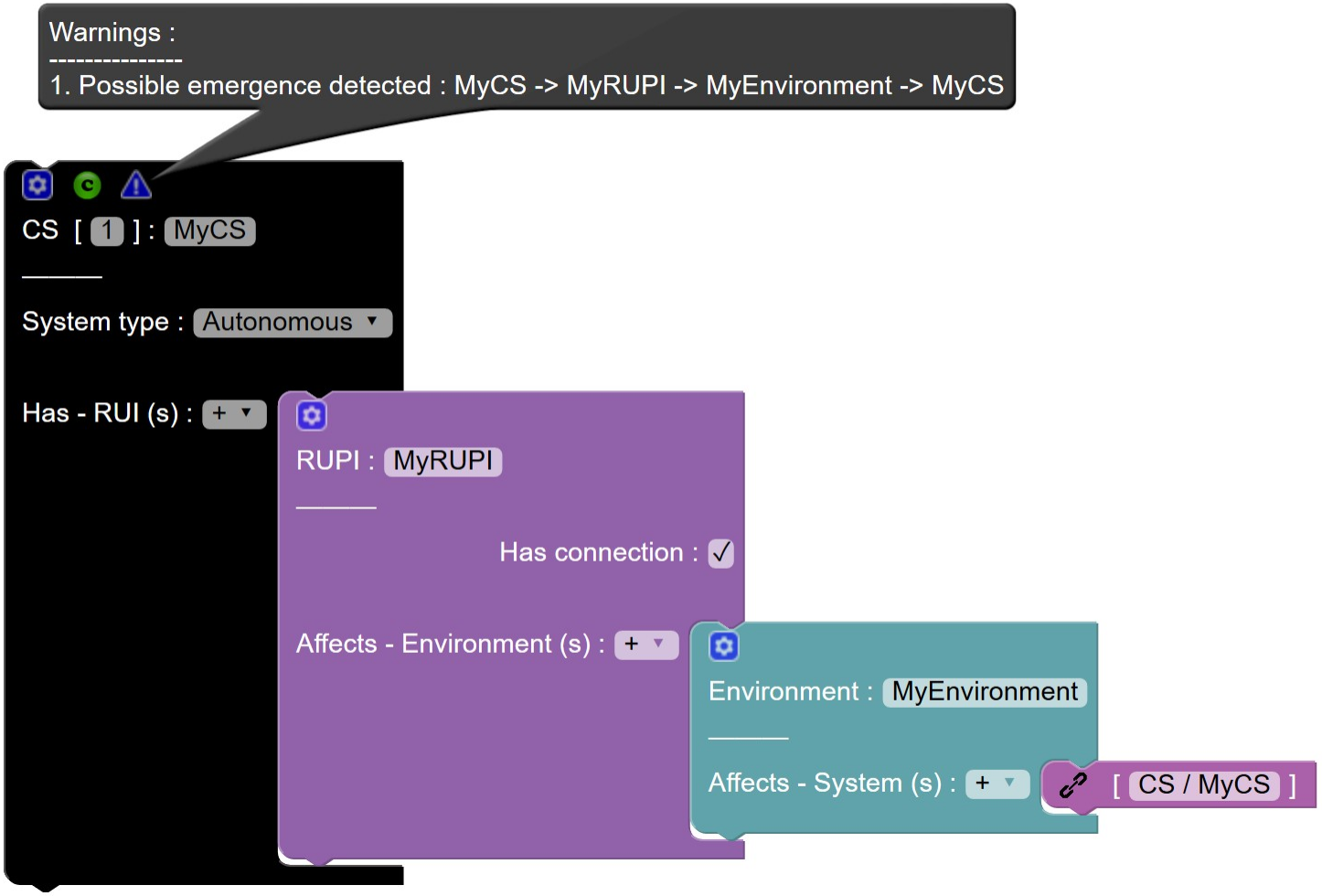
Forcing values !

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

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



Causal loops detection



Constraints can be used to detect possible emergence

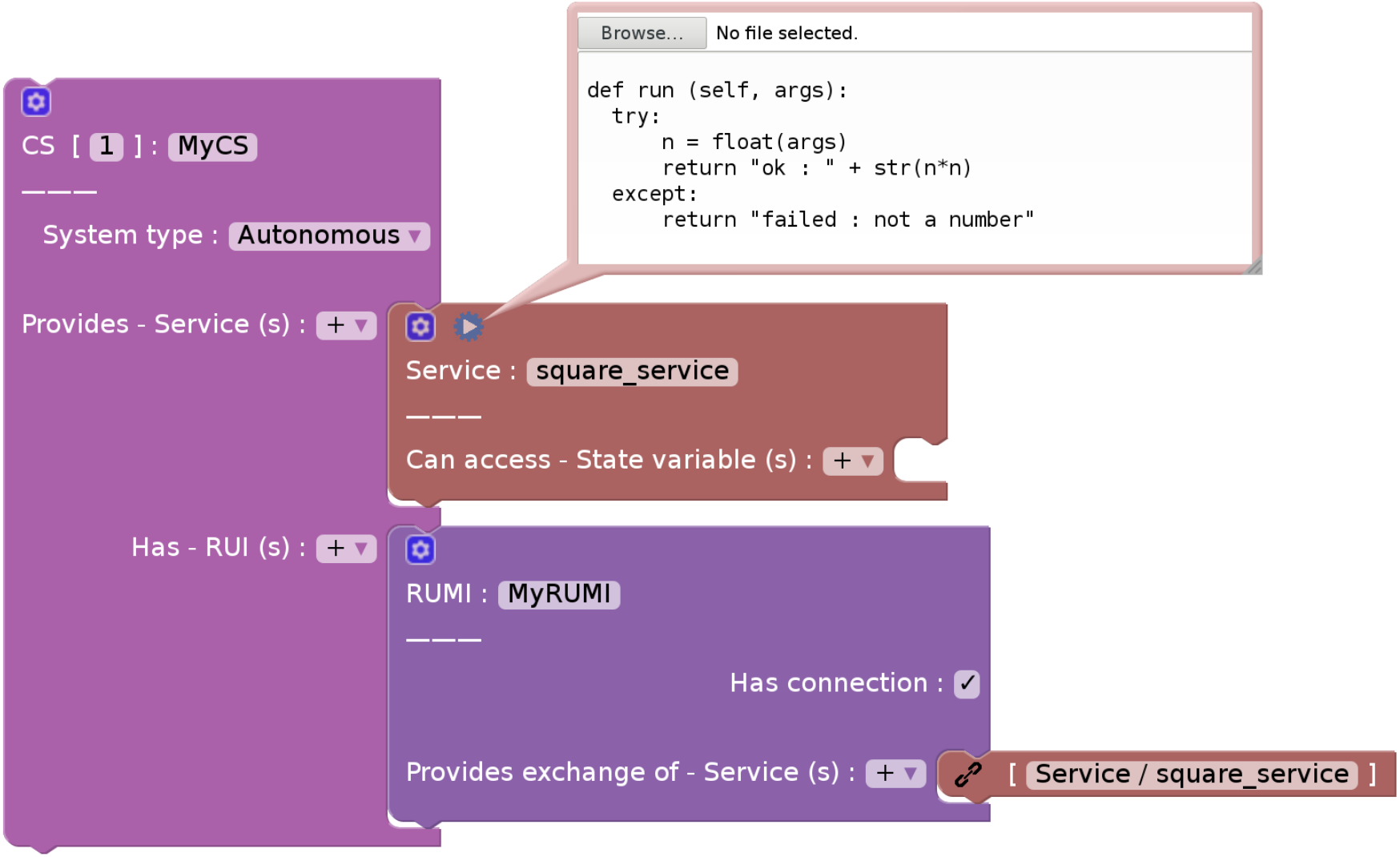
EXAMPLES OF SIMULATIONS

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 !**

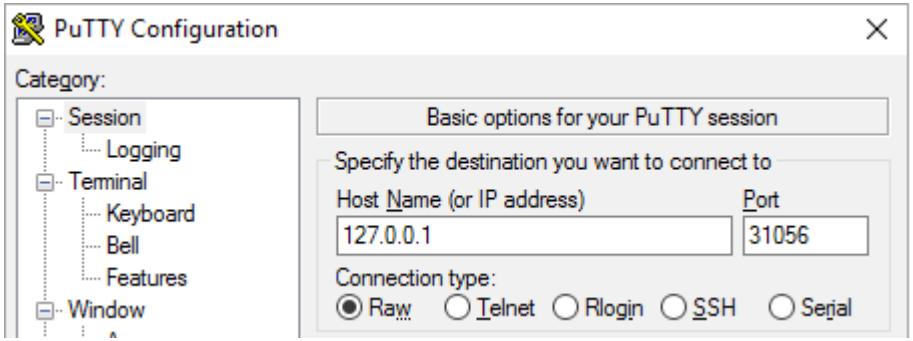
Add behavior - Dynamic part of model



Simulation run

SoS Log!

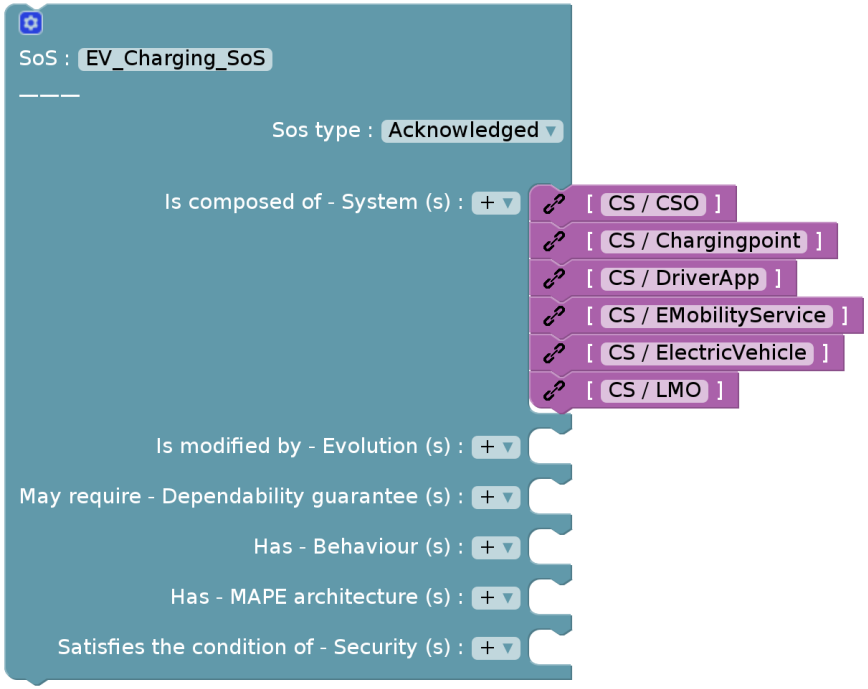
```
2016-10-05T14:53:25.485000 | MyCS -> MyCS                                  | Started !
2016-10-05T14:53:25.486000 | MyCS / MyRUMI -> MyRUMI                       | Listening on 31056
```

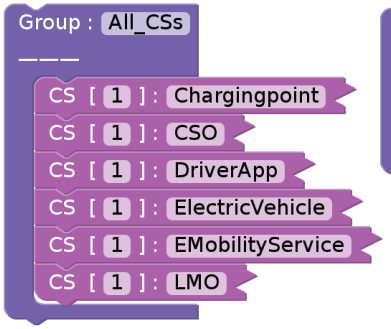


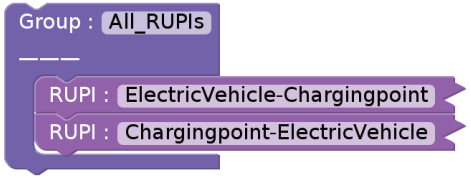

```
---> Connected to RUMI : MyRUMI
square_service abac
failed : not a number


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

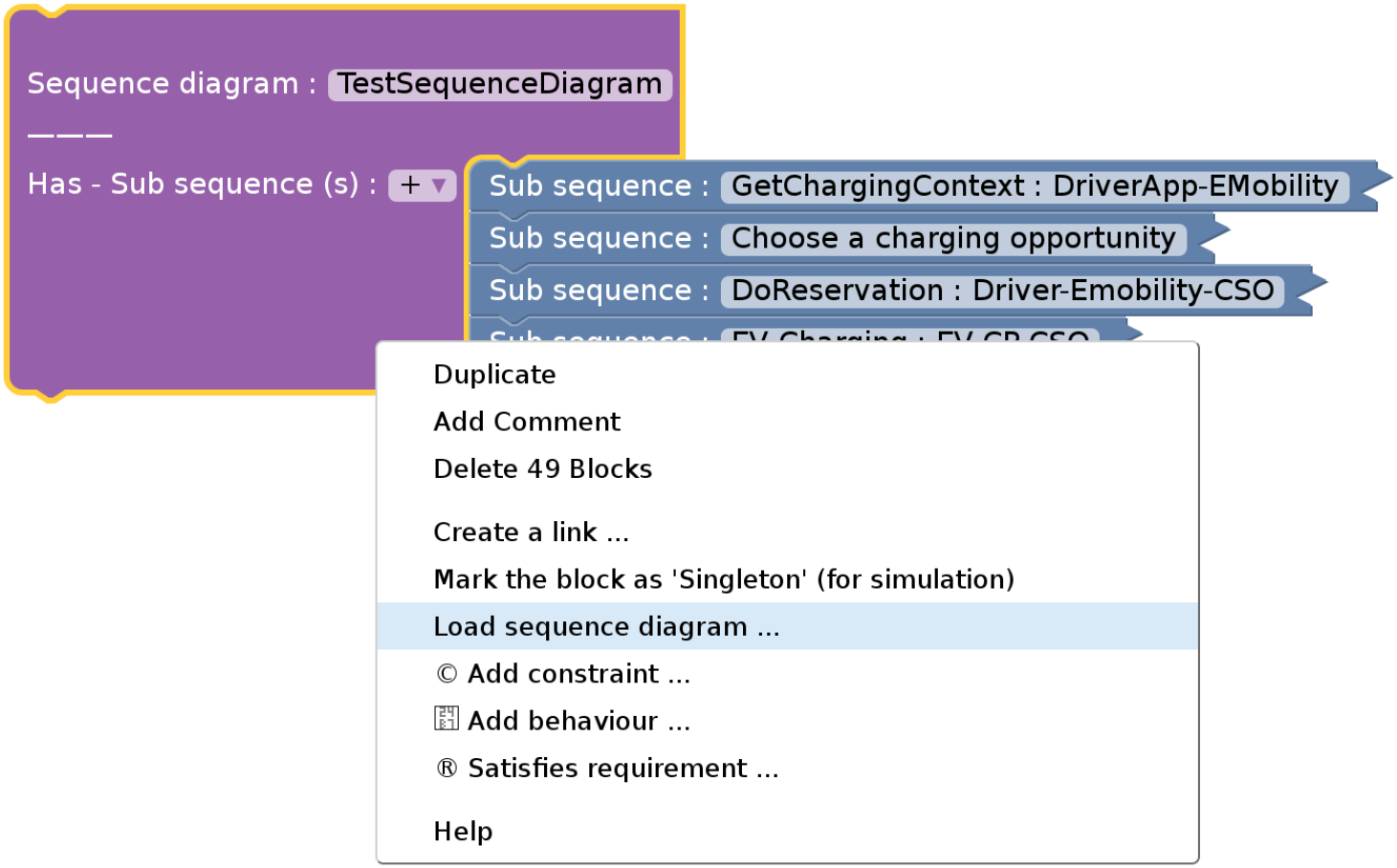
Load the example model





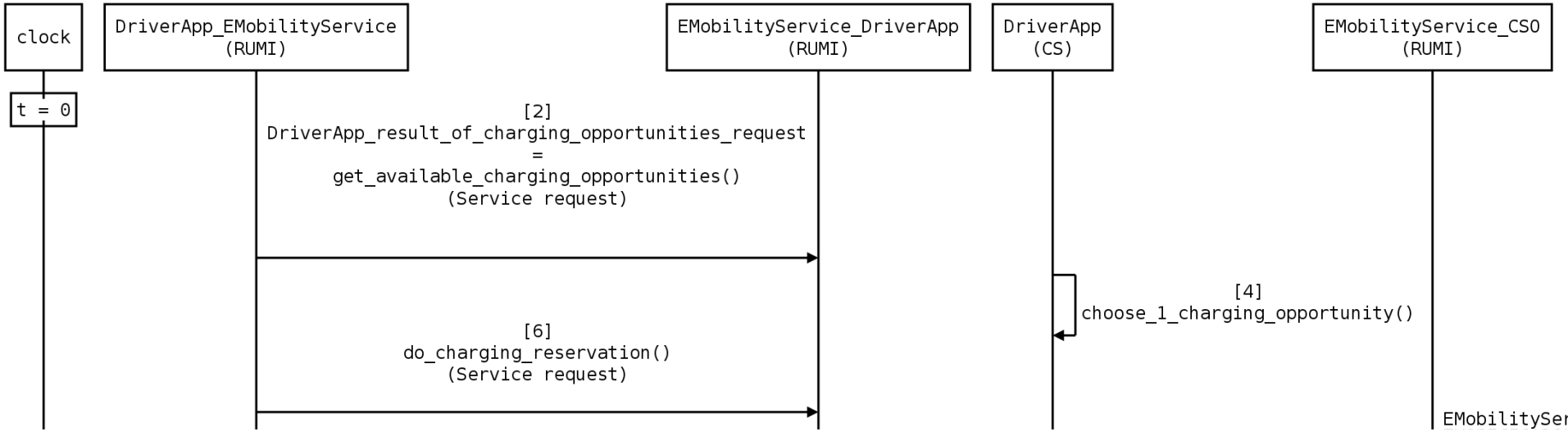


Load sequence diagram



Right click on <u>workspace</u> to view the sequence diagram menu

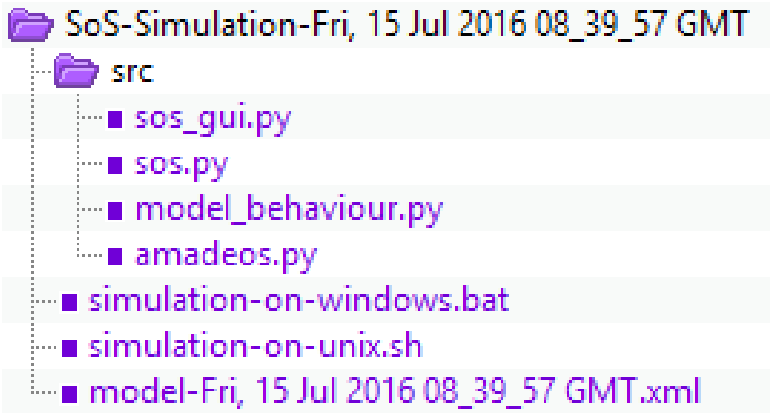
Load sequence diagram



Auto generated sequence diagram

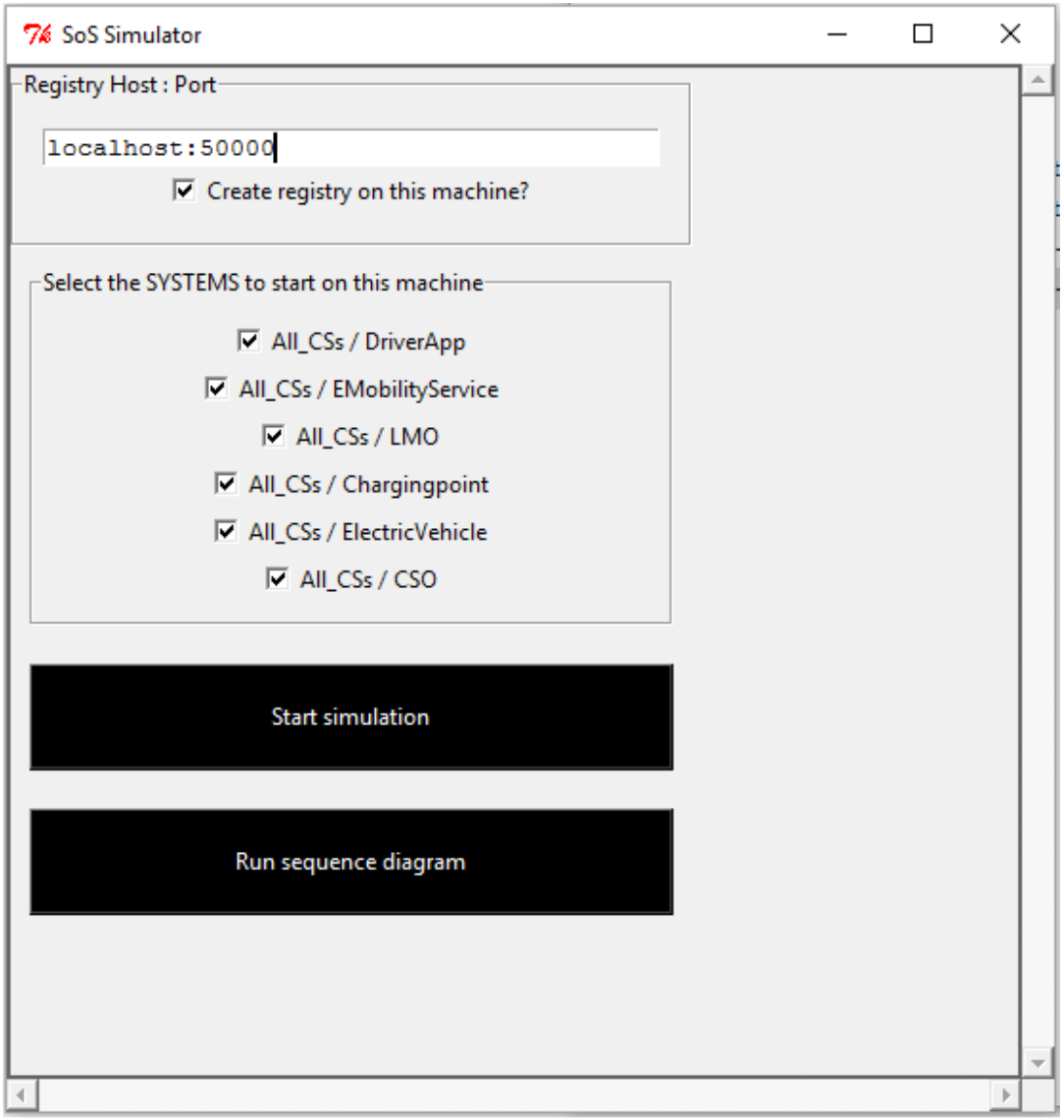
Code generation

The simulation code is generated in the following format:

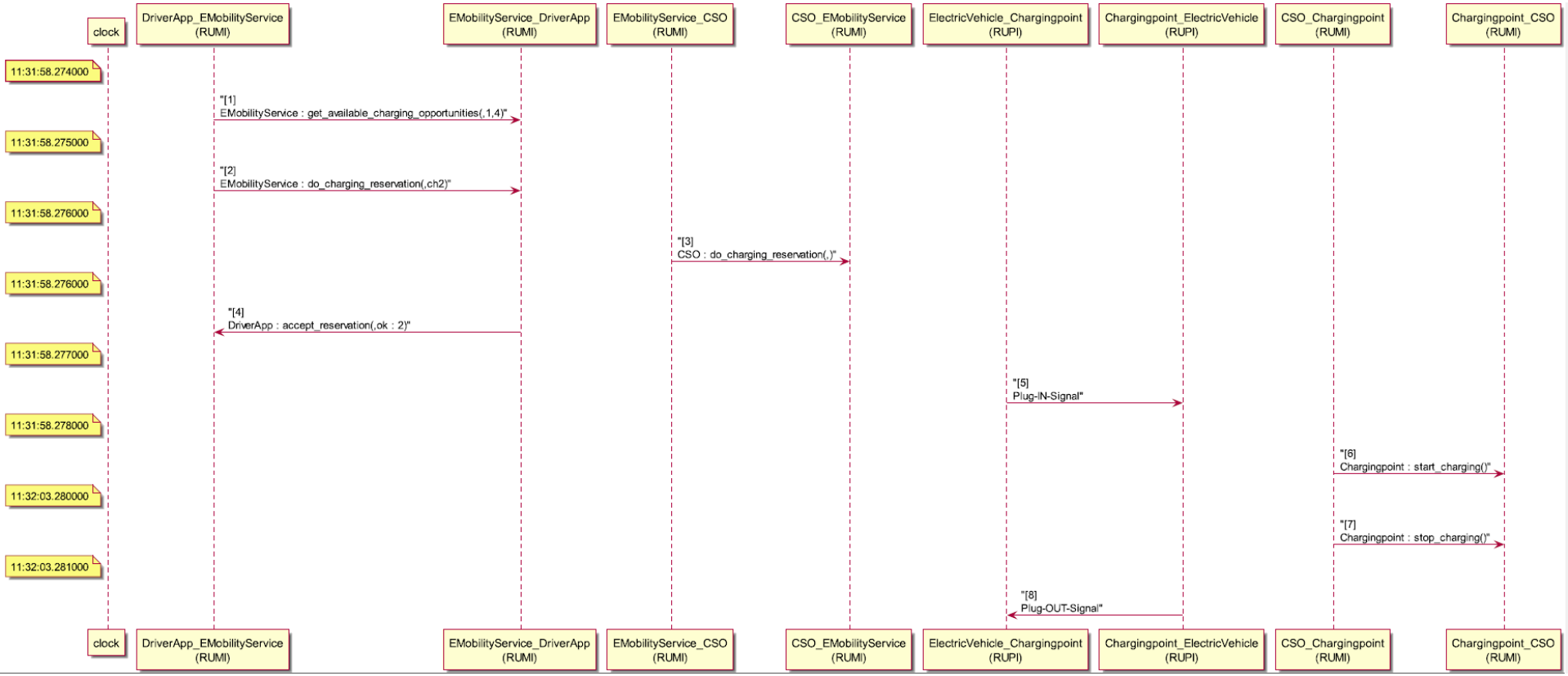


The simulation can be started by clicking on simulation-on-windows.bat or simulation-on-unix.sh depending on the platform of execution

Run simulation



Example simulation result with timestamp



The result of simulation is found in the “result.seq” file, which is the run-time sequence diagram with timestamp.

This result should be compliant with the <u>designed sequence diagram</u>

Blockly XML

- Export the model to XML and have a look at it.

- If you want to transform this model to a custom format you may need to use the .xml file

Model querying

Search inside a large model !

SoS : Smart_Grid_SoS

Sos type : Acknowledged

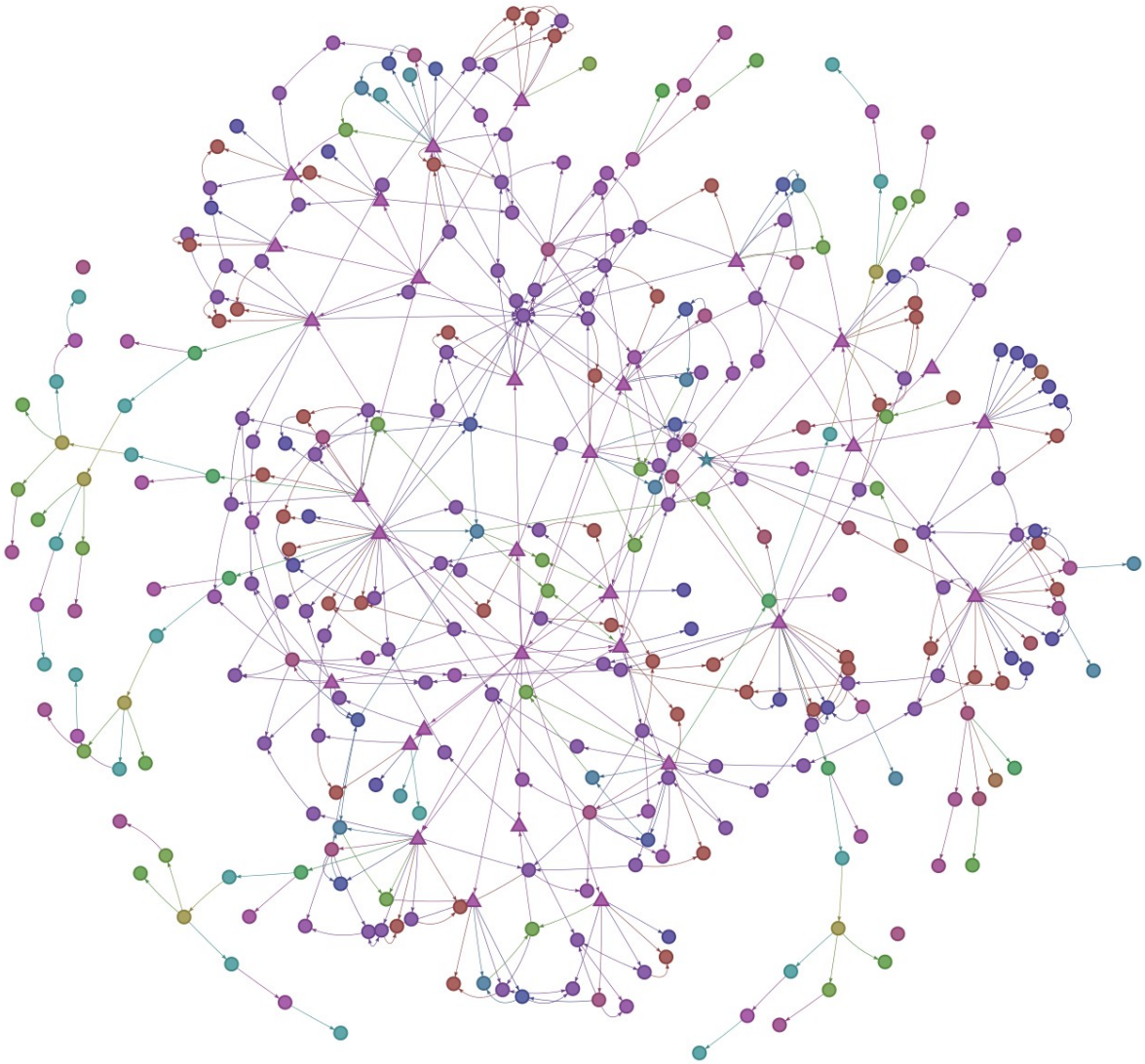
Is composed of - System (s) : +
- CS [1] : EV_Charging
- CS [1] : Medium_Voltage_Control
- CS [1] : Household

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

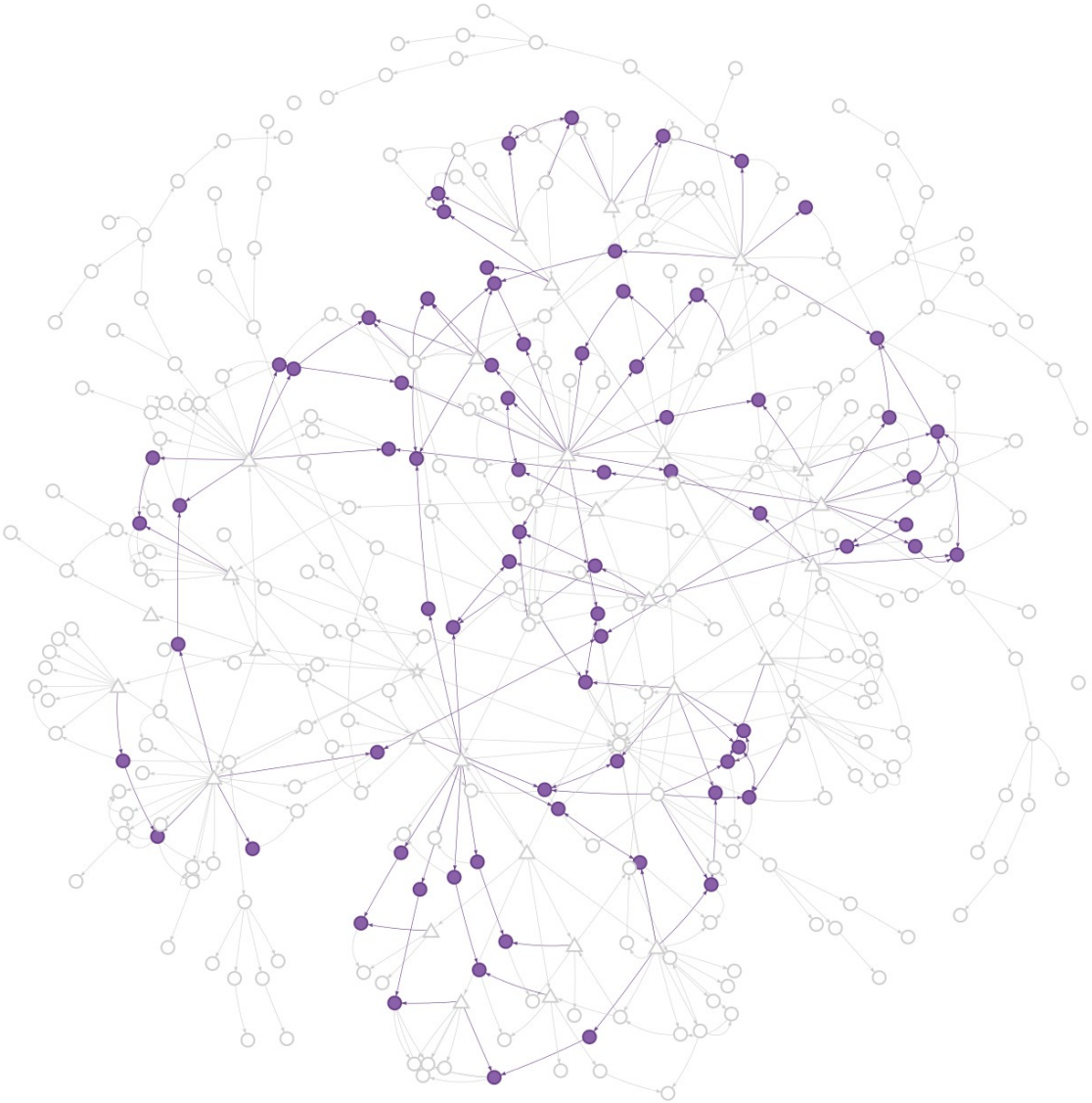
Has - Behaviour (s) : +
- [Expected and beneficial behaviour / Sos Interconnetcion]
- [Unexpected and detrimental behaviour / Ev_connection_disconnection]

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

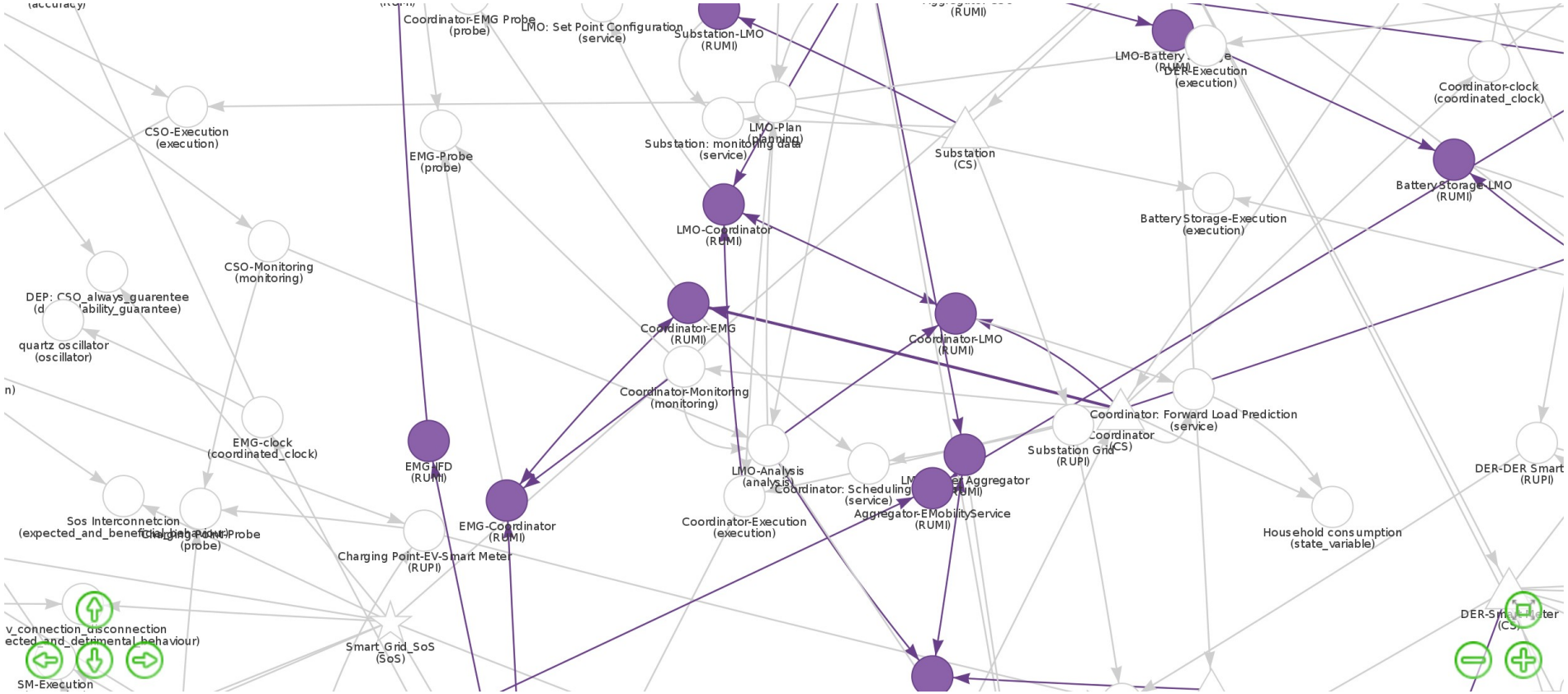
Model query - <u>return true;</u> (i.e. get all blocks)



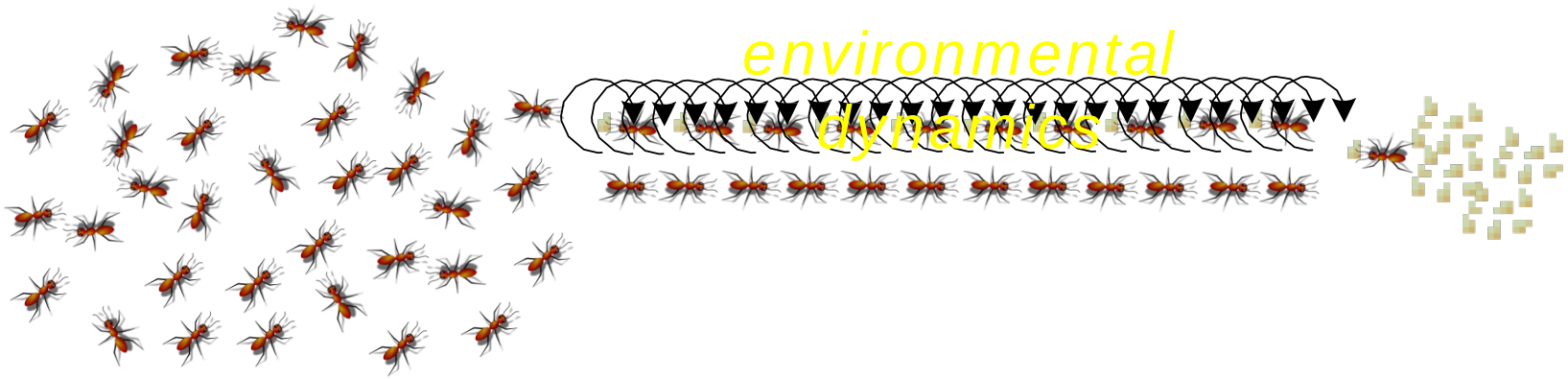
Model query - return block.of type == 'RUMI';



Model query - zoomed results

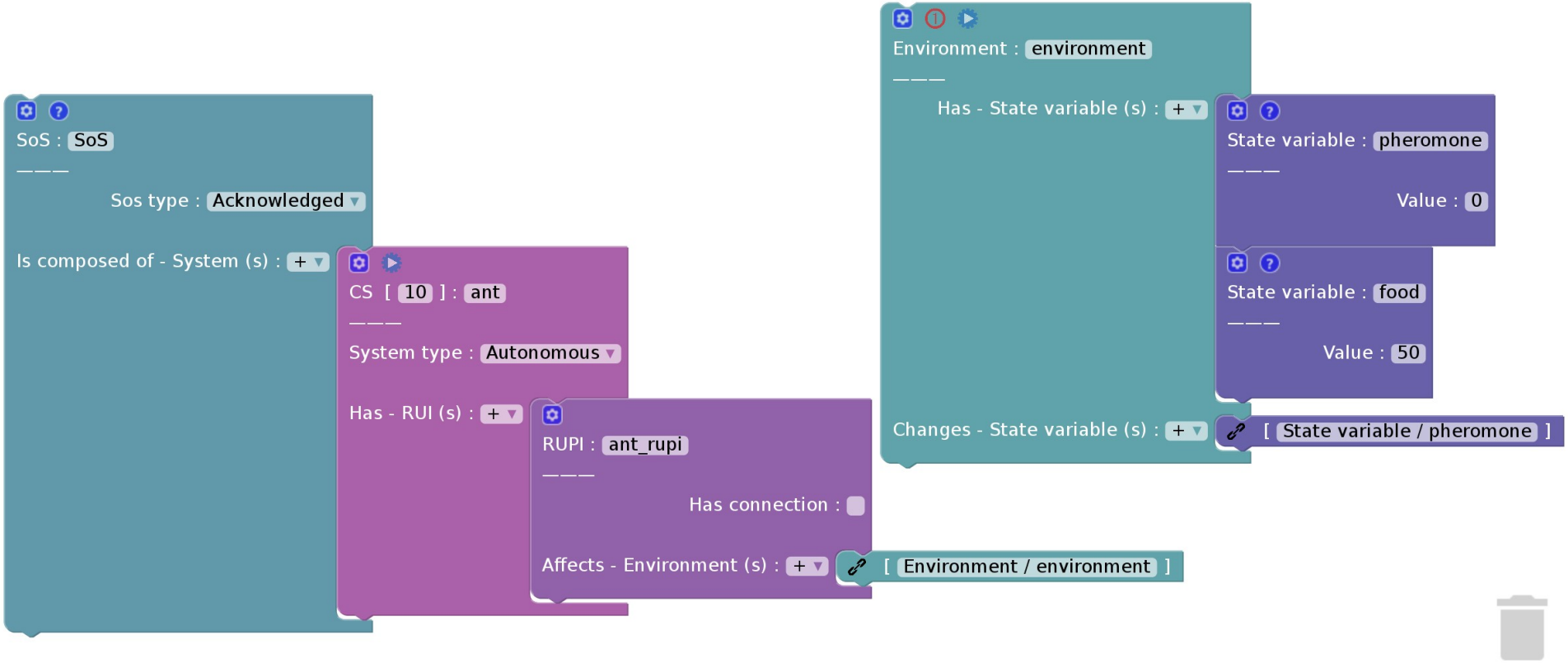


Stigmergic Channels



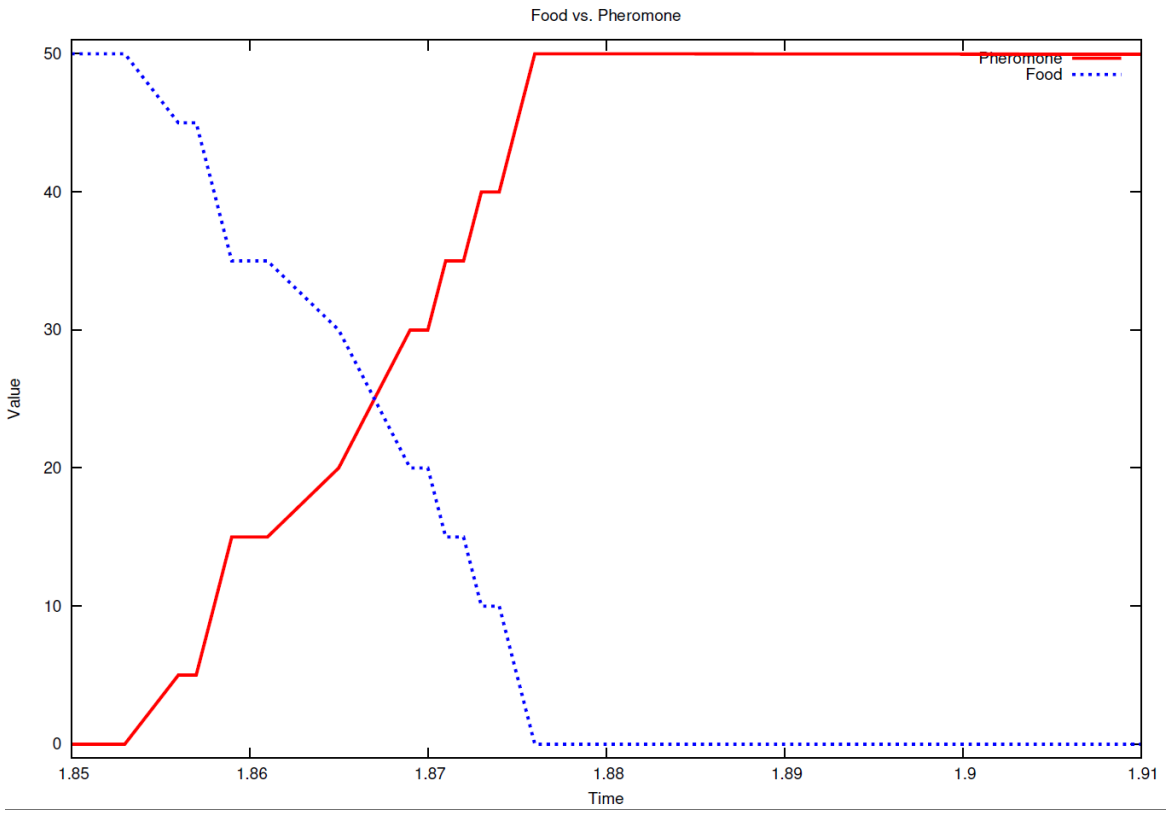
- **Ants find food and build/enforce trail by leaving traces (*pheromone*) in environment on way back.**
- **In case food source depleted,**
  - **ants stop leaving traces,**
  - **The environment evaporates traces autonomously**
    ⇒ *environmental dynamics*.
  - **the trail disappears.**

Ants model



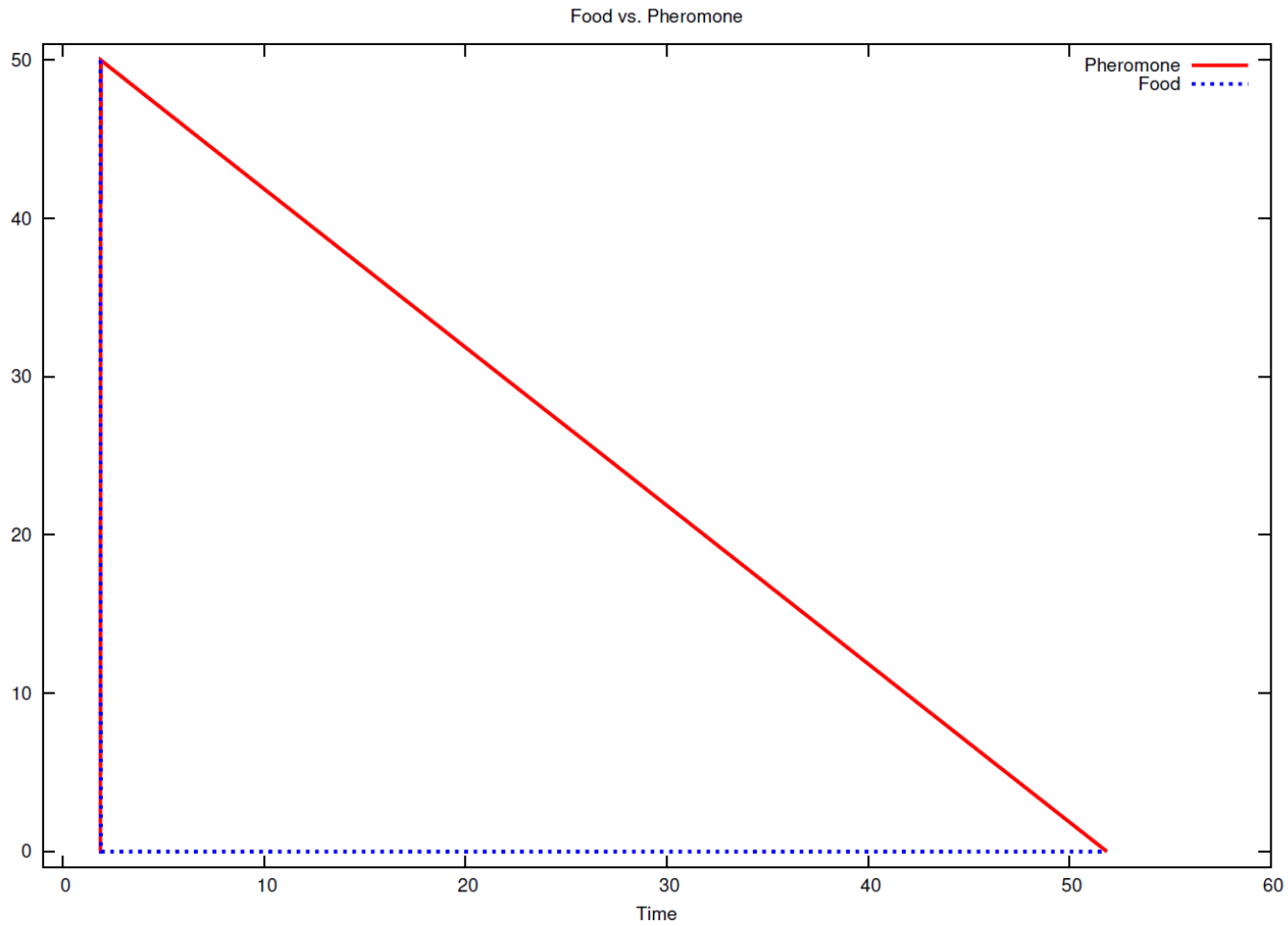
Please note the cardinality (of ants) and singleton (of environment) in the model !

Pheromone vs. Food simulation - 1



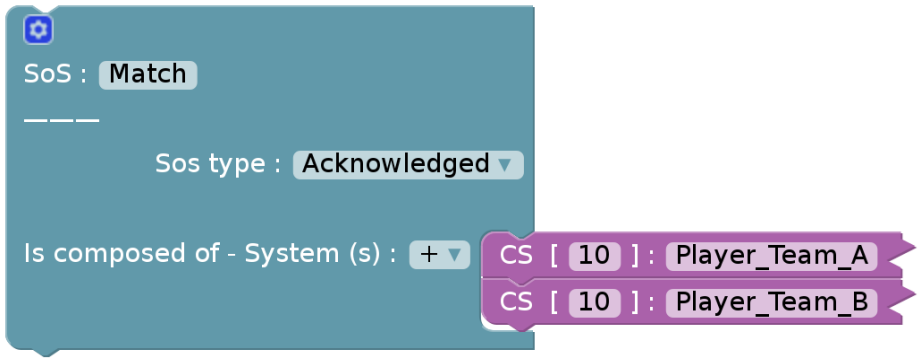
Change in pheromone and food as ants find food

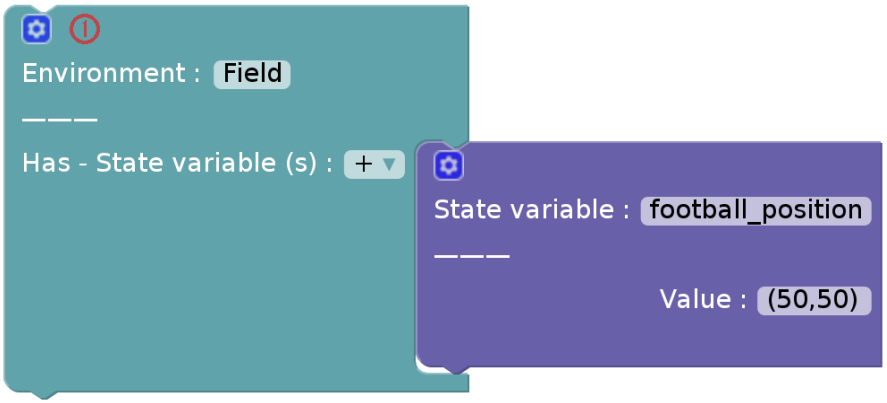
Pheromone vs. Food simulation - 2



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

Football model*

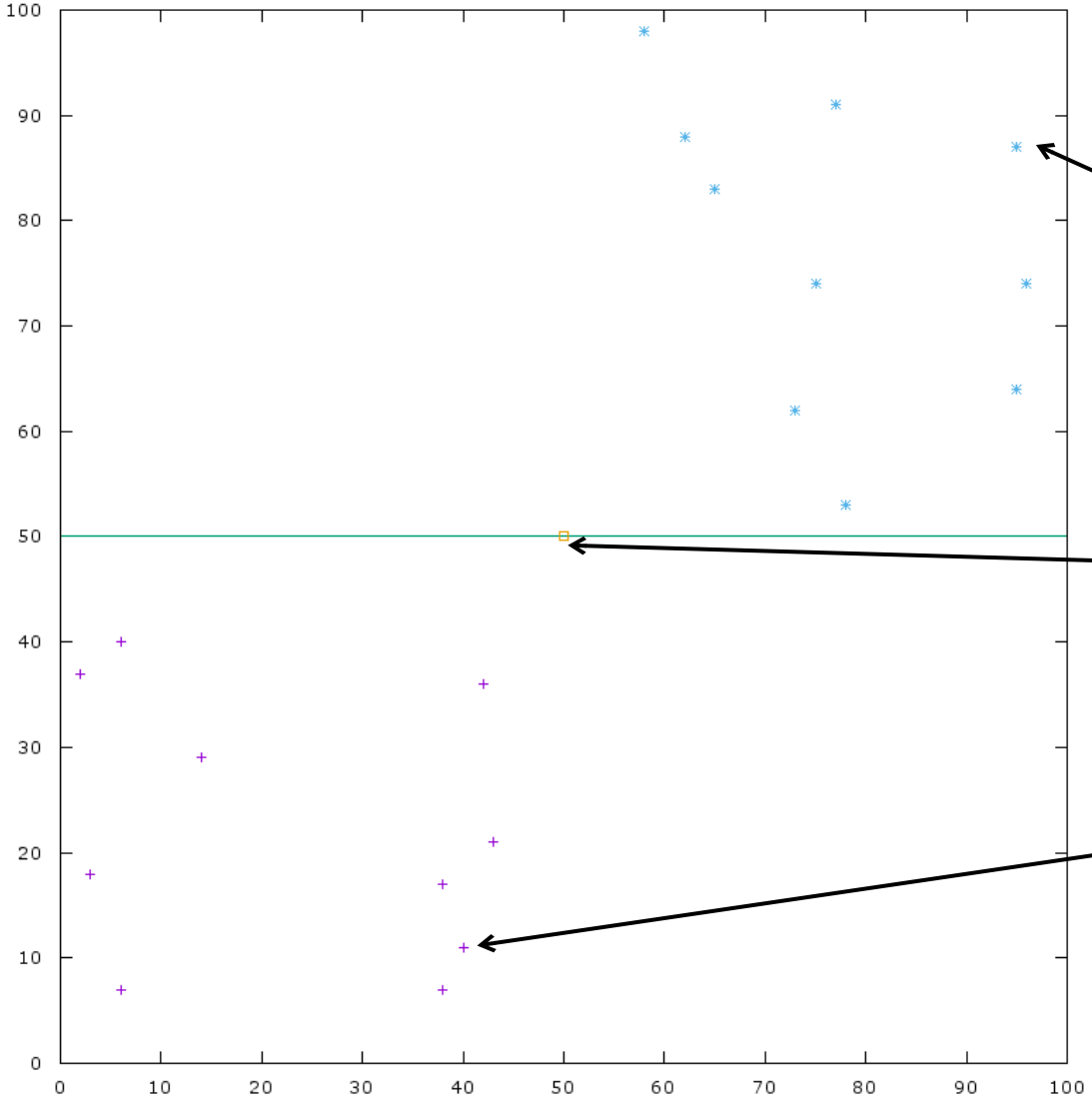




***Not formally a SoS yet - for this we need to add strategies for players.**

In this model, the position of players is random

Food ball simulation results - At 0th second

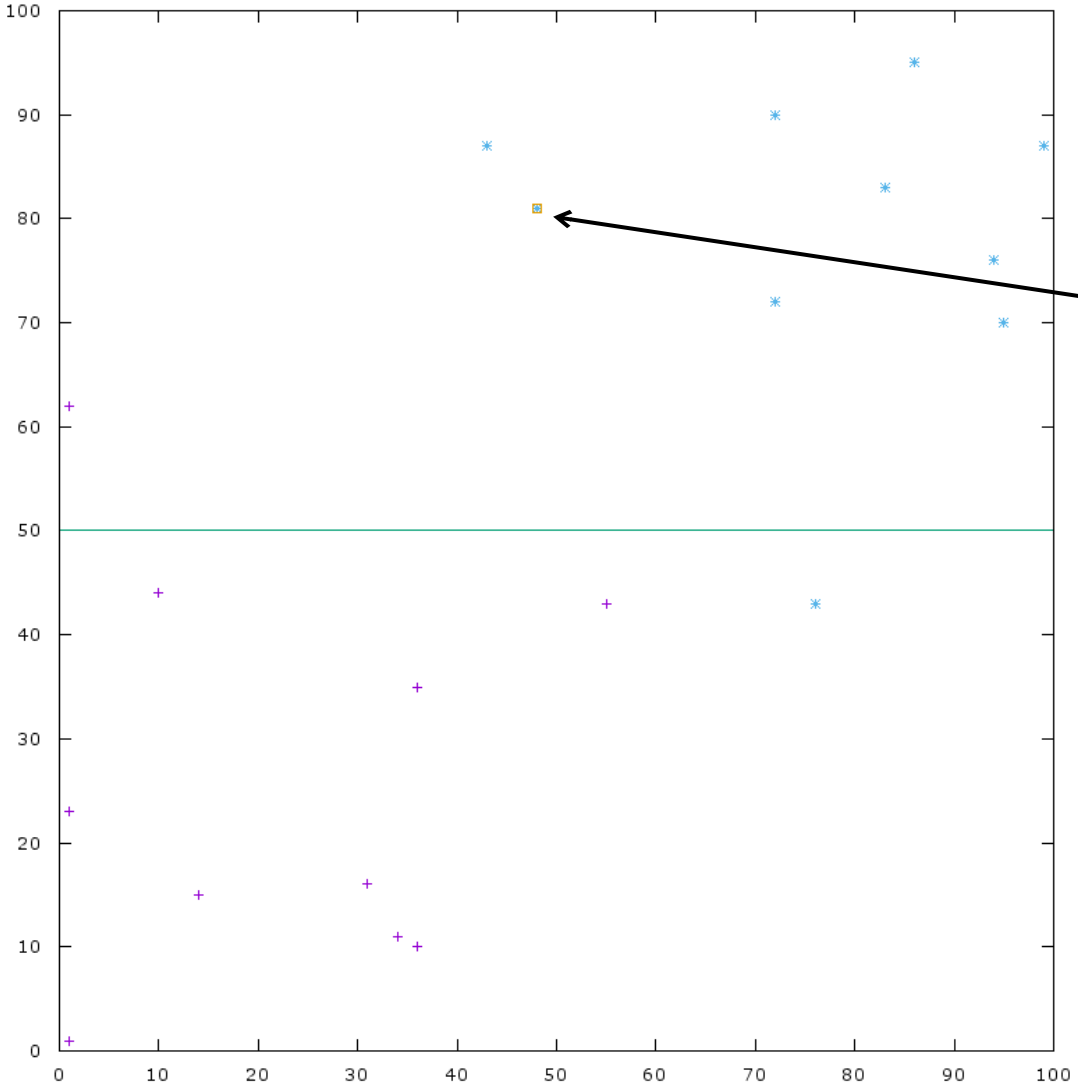


Team A

Football at position (50,50)

Team B

Food ball simulation results - At 25th second



Foot ball is with a team A player

Custom validation/code for Blocks

- **While building the supporting facility tool, if the build tool finds:**

1. **<block-name>.code.js it is added to the block's code for the SoS designer in Blockly**
2. **<block-name>.code.py it is added to the block's code for simulation**

Example: ***SoS.code.js*** **and** ***SoS.code.py***

Beyond SysML …

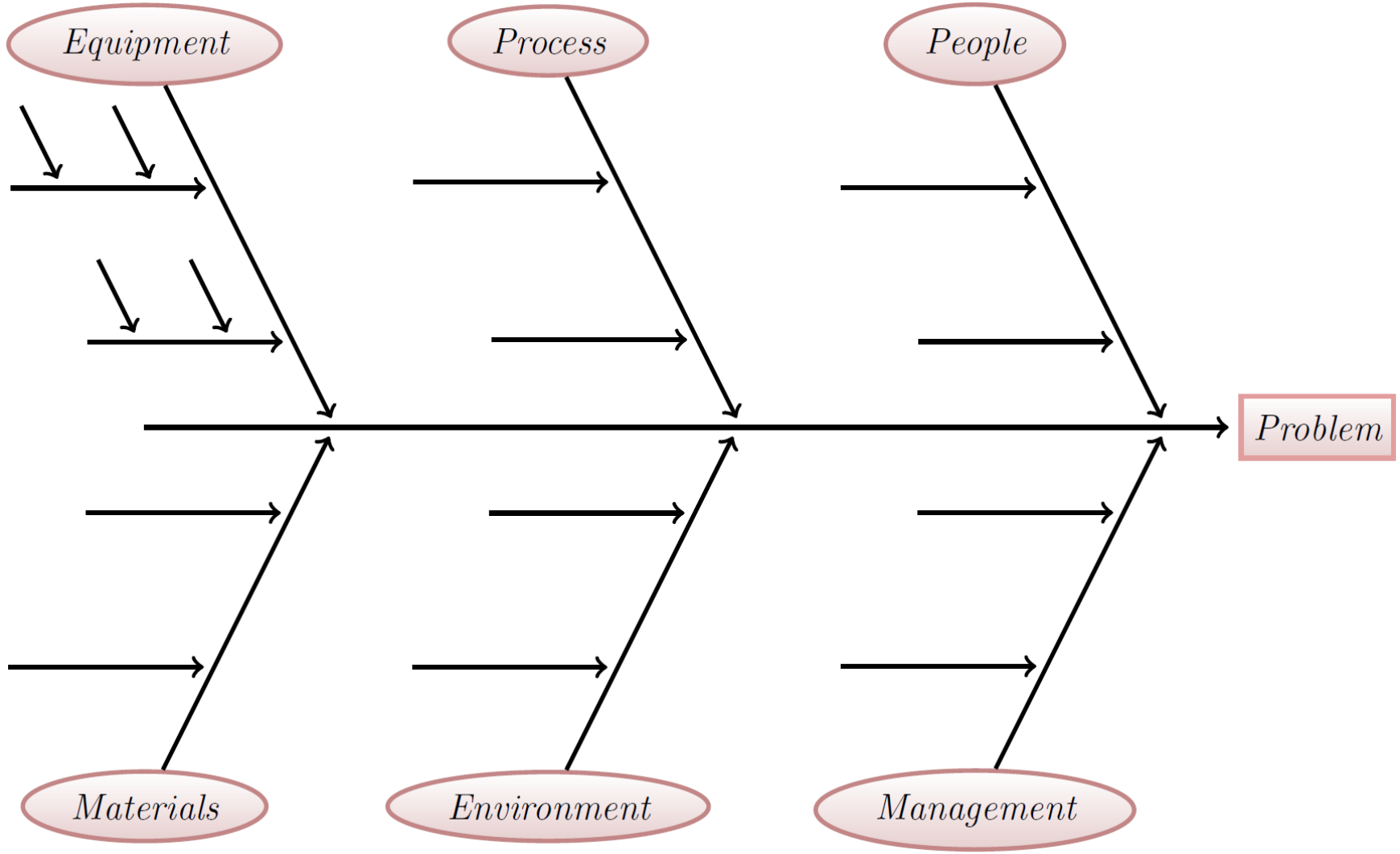
Systems engineering would require other kinds of diagrams

(where SysML may not be the perfect way to represent them)

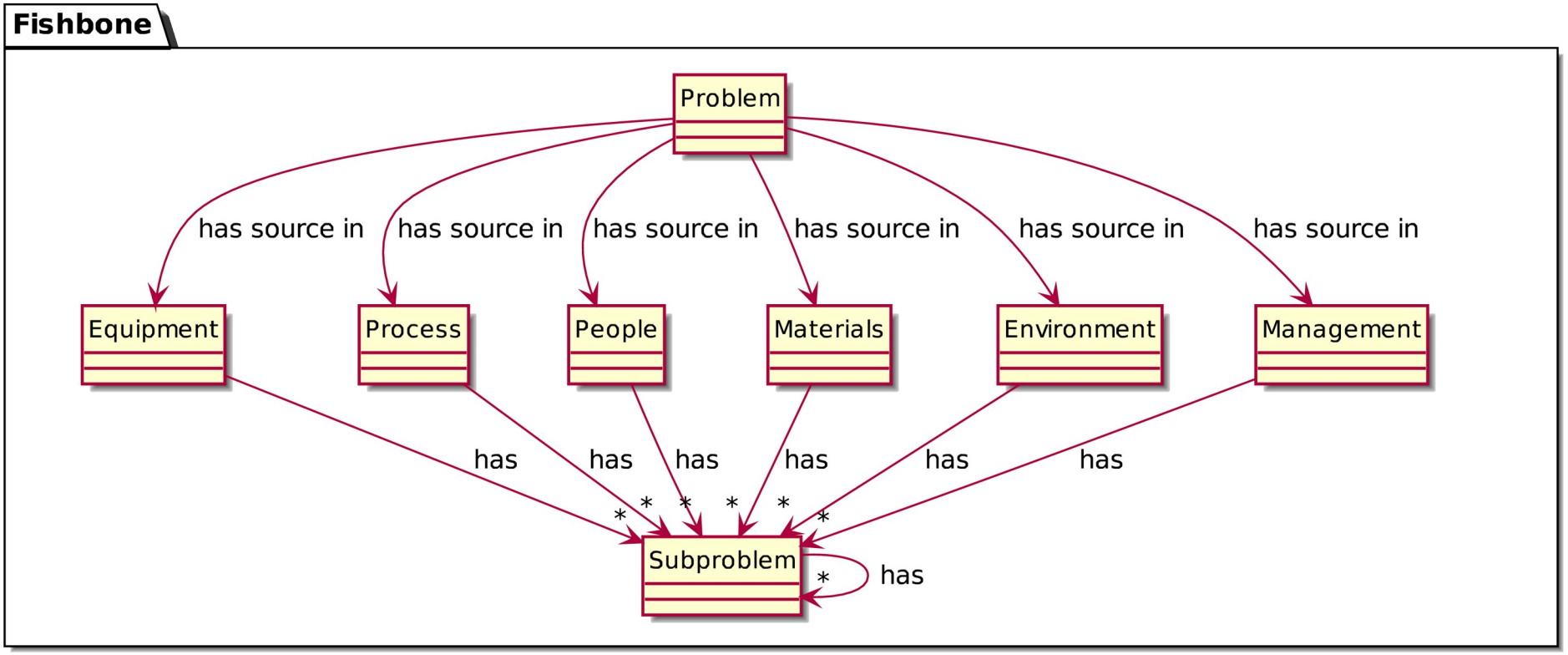
- **Functional block diagram**
- **N2 chart**
- **House of Quality**
- **Ishikawa diagram (fishbone)**
- **Parameter diagram**

Other <u>future</u> diagrams !

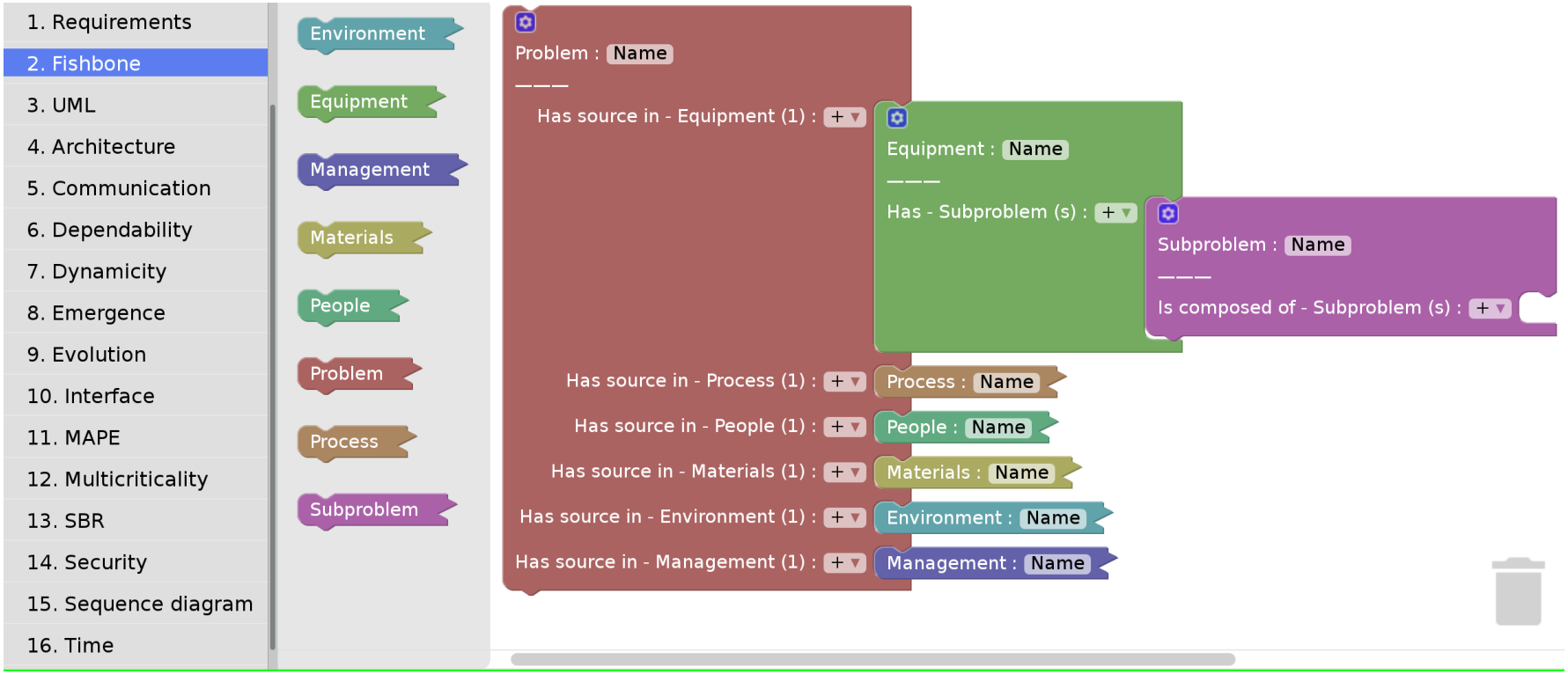
Custom diagrams



Ishikawa/Fishbone diagram – PlantUML diagram



Ishikawa/Fishbone diagram - Blockly



Future ! – Use of image based blocks

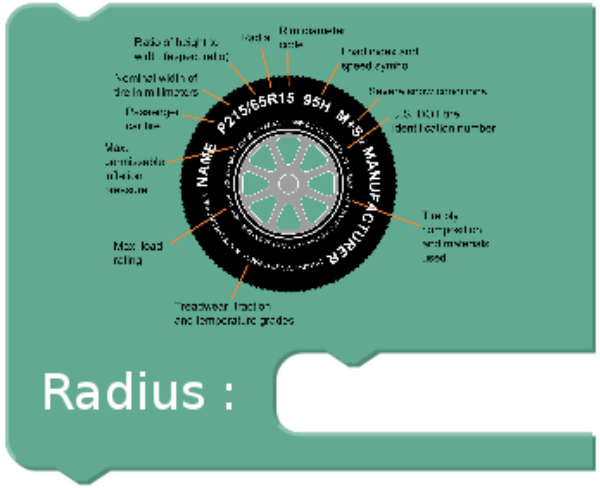


Image based blocks could provide an intutive interface for designing models

Image source : Wikipedia

Conclusion

- Showcased a tool for MDE for SoS developed for the AMADEOS project
- Demonstrated a tool for SoS
  - Design
  - Validation
  - Querying
  - Simulation
- There is good scope for improvements to bring MDE tools to masses

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