

# Using Petri nets within Cyber-Physical System's development

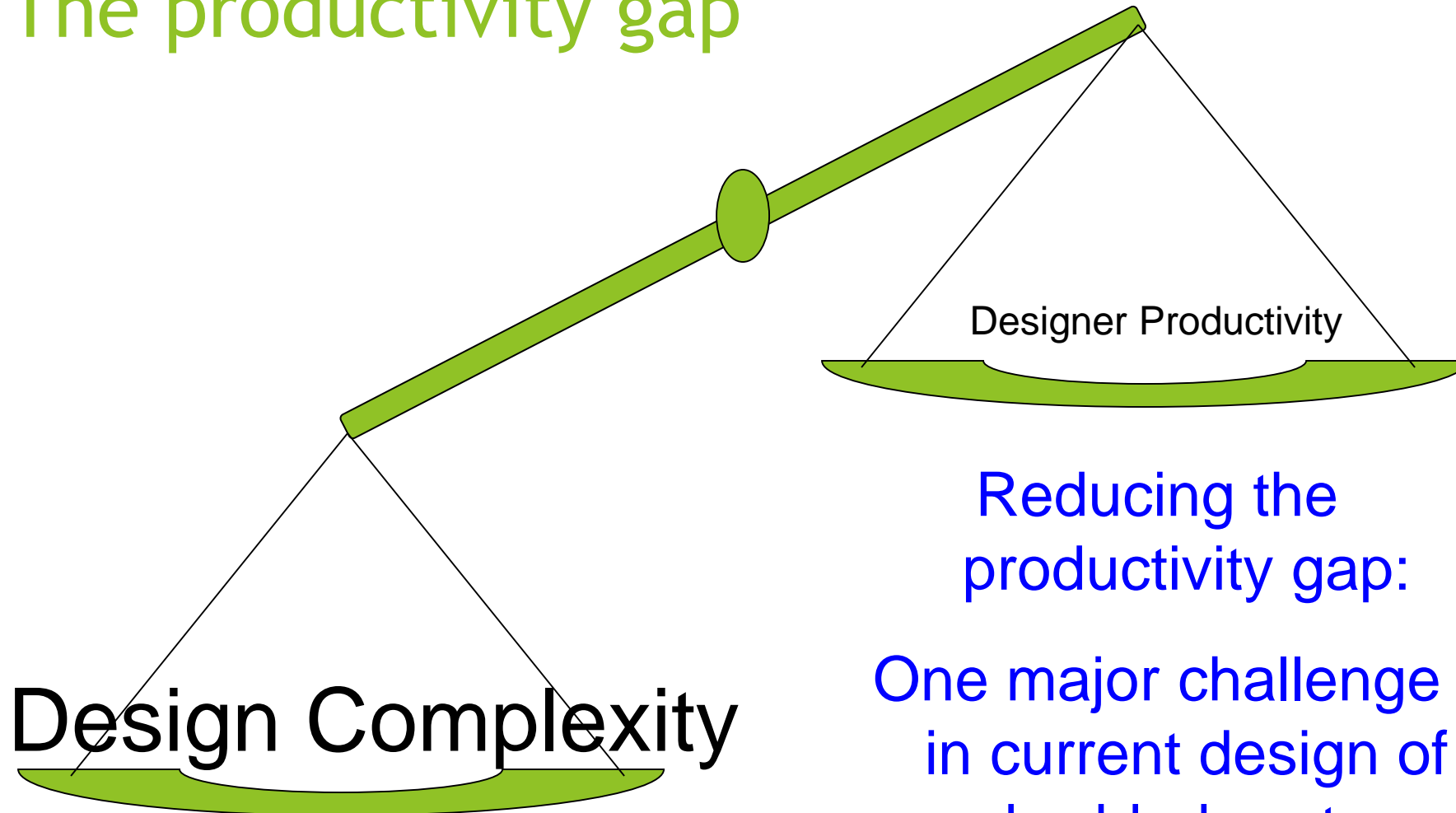
IOPT-Tools - Web-based platform for embedded controllers development based on IOPT Petri nets

Anikó Costa - [akc@fct.unl.pt](mailto:akc@fct.unl.pt)

# Outline

- ▶ Motivation to move towards model-based development
- ▶ Petri nets - a brief overview
- ▶ IOPT-Tools framework
- ▶ Conclusions

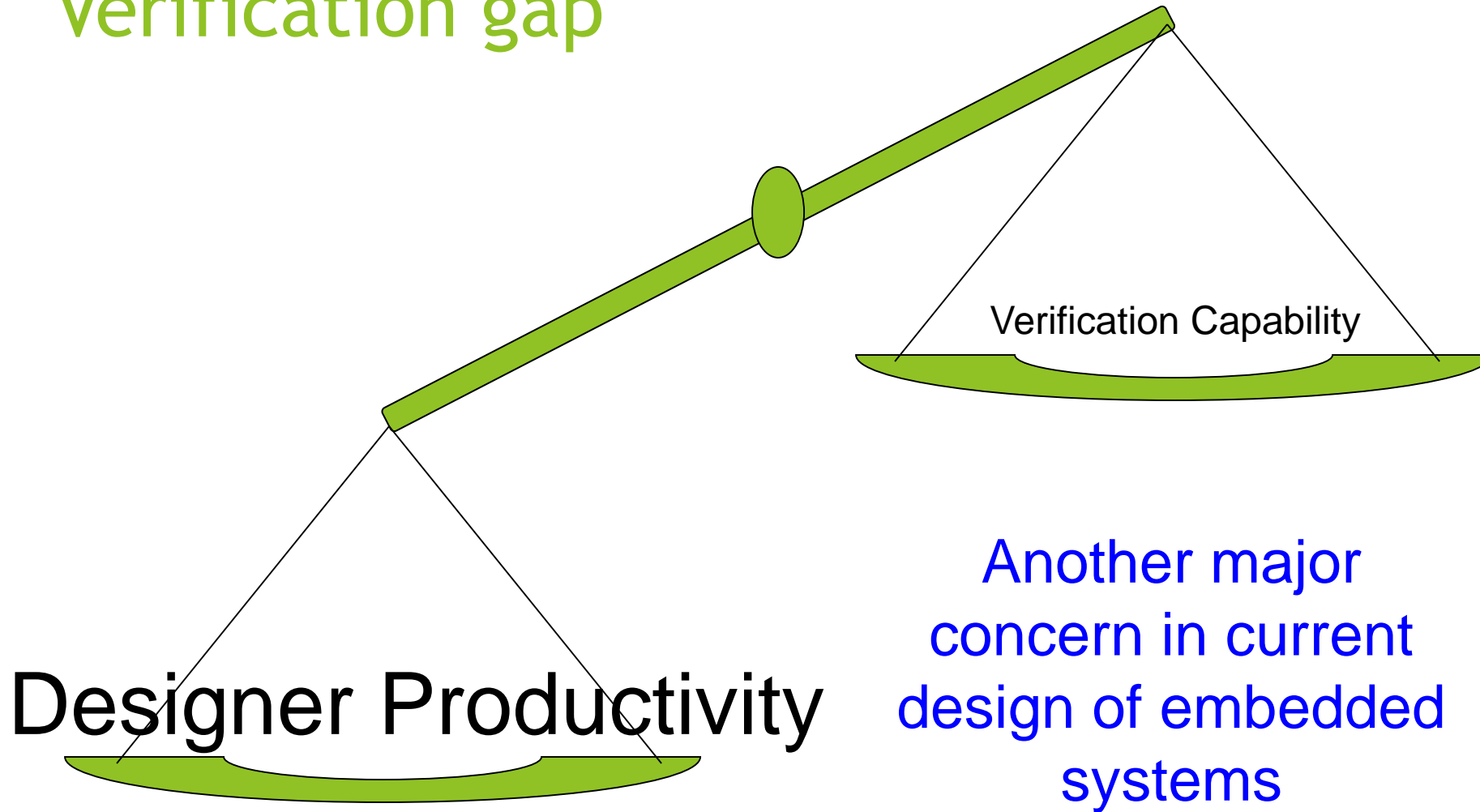
# The productivity gap



Reducing the productivity gap:

One major challenge in current design of embedded systems

# Verification gap



# The performance gap

- ▶ More performance always needed (at least wanted)
- ▶ Increasing clock frequency is not enough
- ▶ Exploiting concurrency and distributed computing and control is one major option to support improvements
- ▶ Although, we need mechanisms to allow robust synchronization, sharing of resources, mutual exclusion, and so on ...

# Open issues and challenges

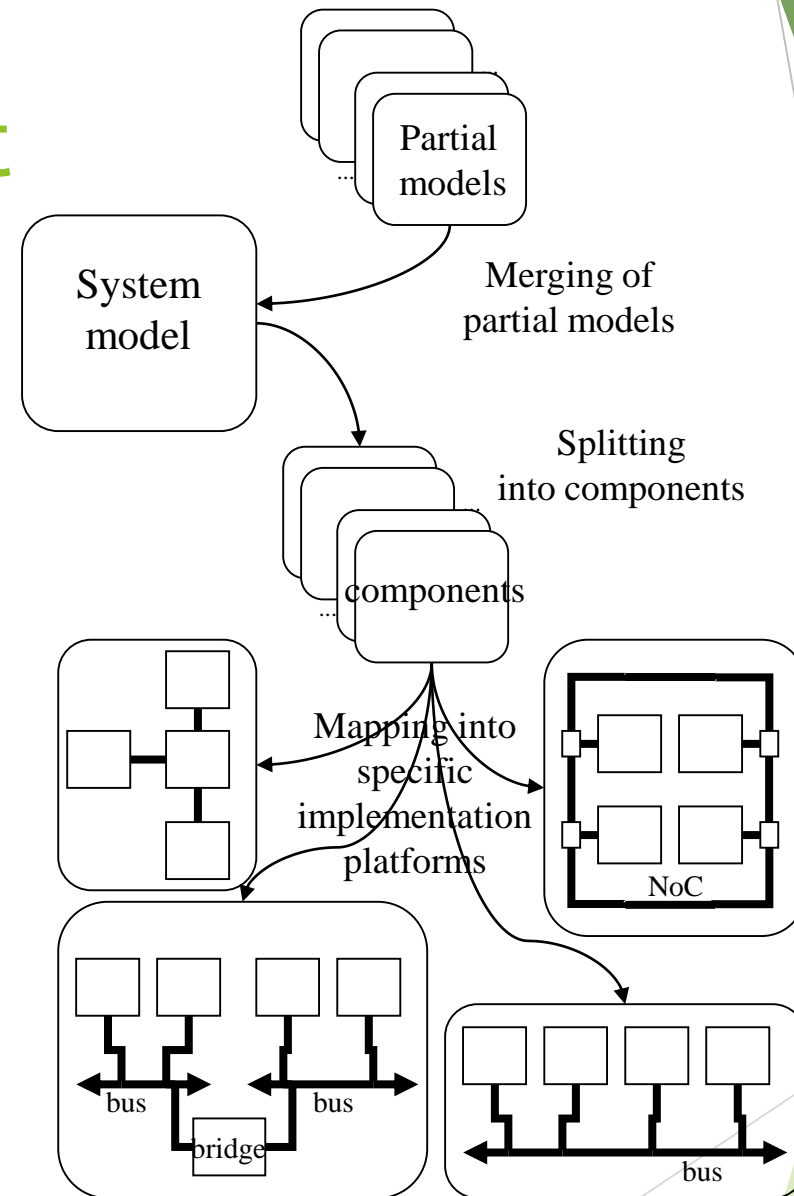
- ▶ How to handle design complexity?
- ▶ How to reduce the productivity gap?
- ▶ How to reduce the verification gap?
- ▶ How to cope with the performance gap?
- ▶ How to support reliable distributed execution?
- ▶ **Contribution to the answers:**
  - ▶ Relying more and more on Model-based Development
  - ▶ Increasing usage of design automation tools (including specification, simulation/validation, verification, code generation, and test)

# Moving to model-based development

- ▶ Models are used not only for describing specifications of the system at earlier phase of development, but also intended to be used along the whole development process, including automatic code generation (verification and implementation)
- ▶ Start with platform independent specification, “easily” supporting porting/implementation into specific platforms.
- ▶ For that end, an operational model having a precise execution semantics needs to be selected, allowing usage of the model at the different stages of the development process.

# Underlying development methodology

- ▶ Starting from partial models
- ▶ System model by merging partial models
- ▶ System components using model splitting
- ▶ Mapping into specific implementations platforms





# Selection of modeling formalism

- ▶ Among those eligible most common formalisms, it is worth to mention state diagrams, hierarchical and concurrent state diagrams, statecharts, and Petri nets.
- ▶ All of them can have:
  - ▶ Rigorous computational model
  - ▶ Precise execution semantics
  - ▶ Graphical representation
  - ▶ Formal representation

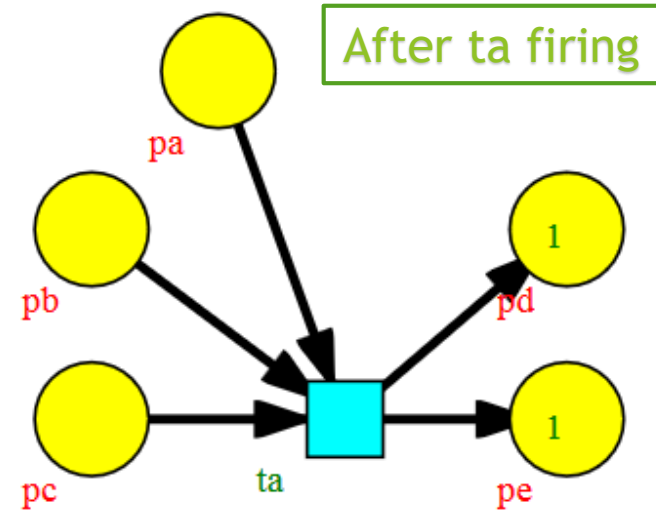
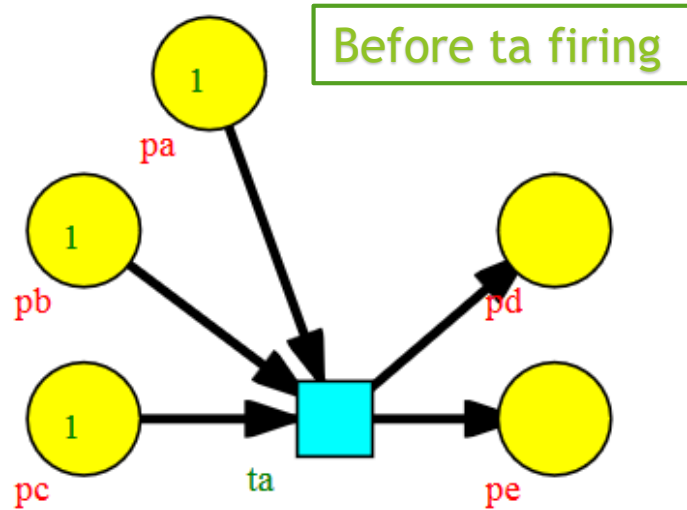
# Petri nets : what ? (I)

- ▶ Petri nets allow the modelling of system's behavior, starting from the concept of event and condition (close to the state concept).
- ▶ A first characterization can see Petri nets as a generalization of state diagrams.
- ▶ Graphical formalism, allowing an easy understanding of system's behavior (strong point for designers), with formal representation capabilities (strong point for tool developers).

# Petri nets: what ? (II)

- ▶ Bipartite graph, composed by two types of nodes:
  - ▶ Conditions or places, represented as circles or ellipses;
  - ▶ Events or transitions, represented by bars, squares or rectangles;
- ▶ Directed arcs that can interconnect nodes of different types;
- ▶ Model dynamics is associated with transitions, while the places represents the static component.

# Firing rules

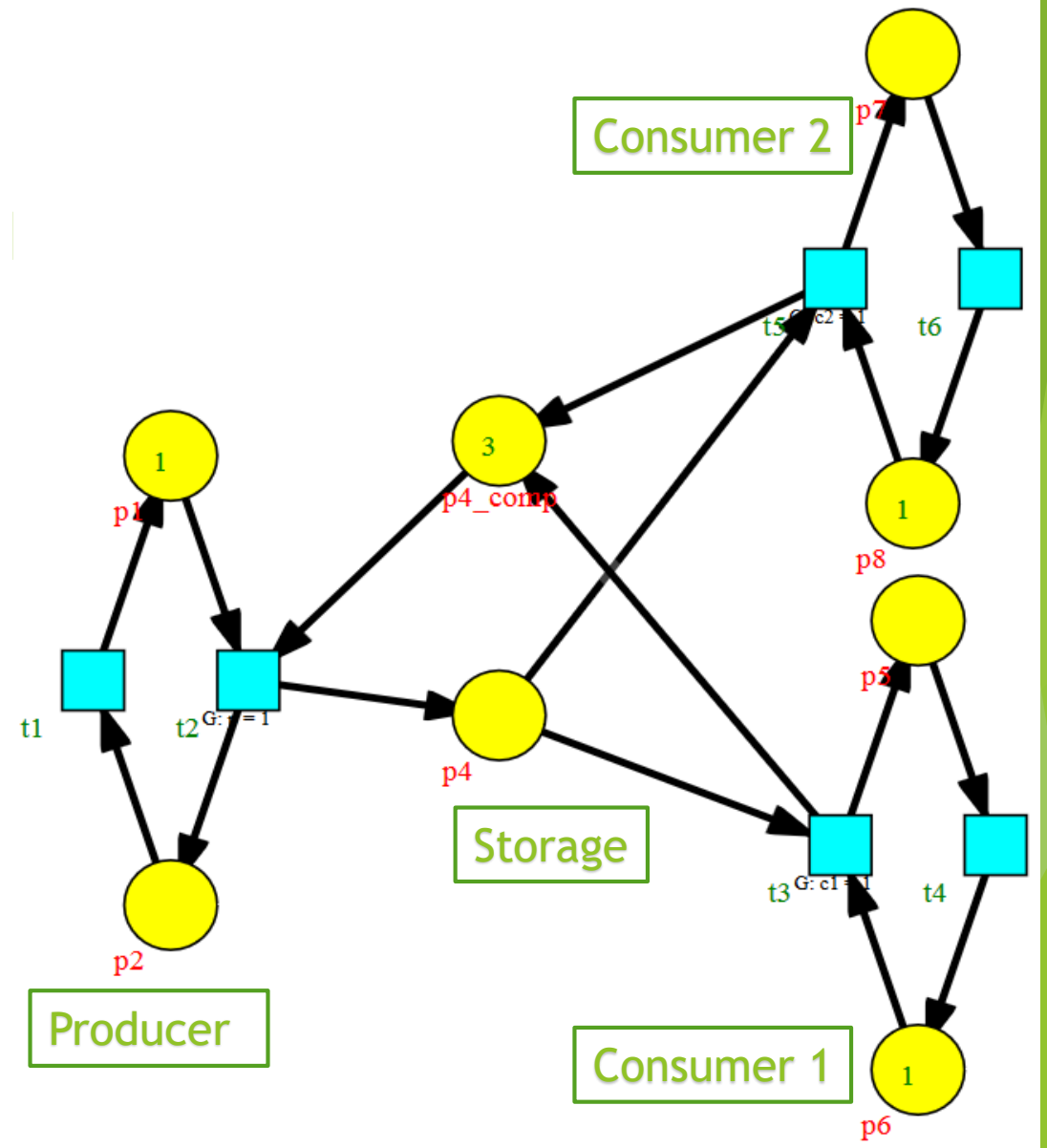


- ▶ Enabled transitions can be fired
- ▶ To be enabled, necessary to comply with enabling pre-conditions (input places marked).
- ▶ For some types of Petri nets, also necessary to comply with post-conditions (output places unmarked).
- ▶ Destruction of input tokens and creation of output tokens -> Atomic action

# Example: Producer-consumer system

Typical modeling situations:

- ▶ Concurrency
- ▶ Local evaluation
- ▶ Synchronization
- ▶ Conflict



# Petri net classes

- ▶ Low-level Petri nets versus High-level Petri nets
  - ▶ Place-Transition nets vs Coloured Petri nets
  - ▶ Safe, bound vs unbound nets
- ▶ Autonomous Petri net vs Non-autonomous Petri nets

Operational semantics / implementation issues:

- ▶ Synchronous execution
- ▶ Asynchronous execution
- ▶ Globally asynchronous locally synchronous (GALS)

# Petri nets for controller modeling

- ▶ Starting with autonomous classes of Petri nets...
- ▶ Extremely important to have the possibility to add dependencies to the environment under control, namely input and output signals and events.
- ▶ In those cases, Petri nets classes become non-autonomous.
- ▶ Several classes of non-autonomous Petri nets have been referred in the literature (some having strong links with automation systems ex. Silva 1985, Frey & Wagner 2000)

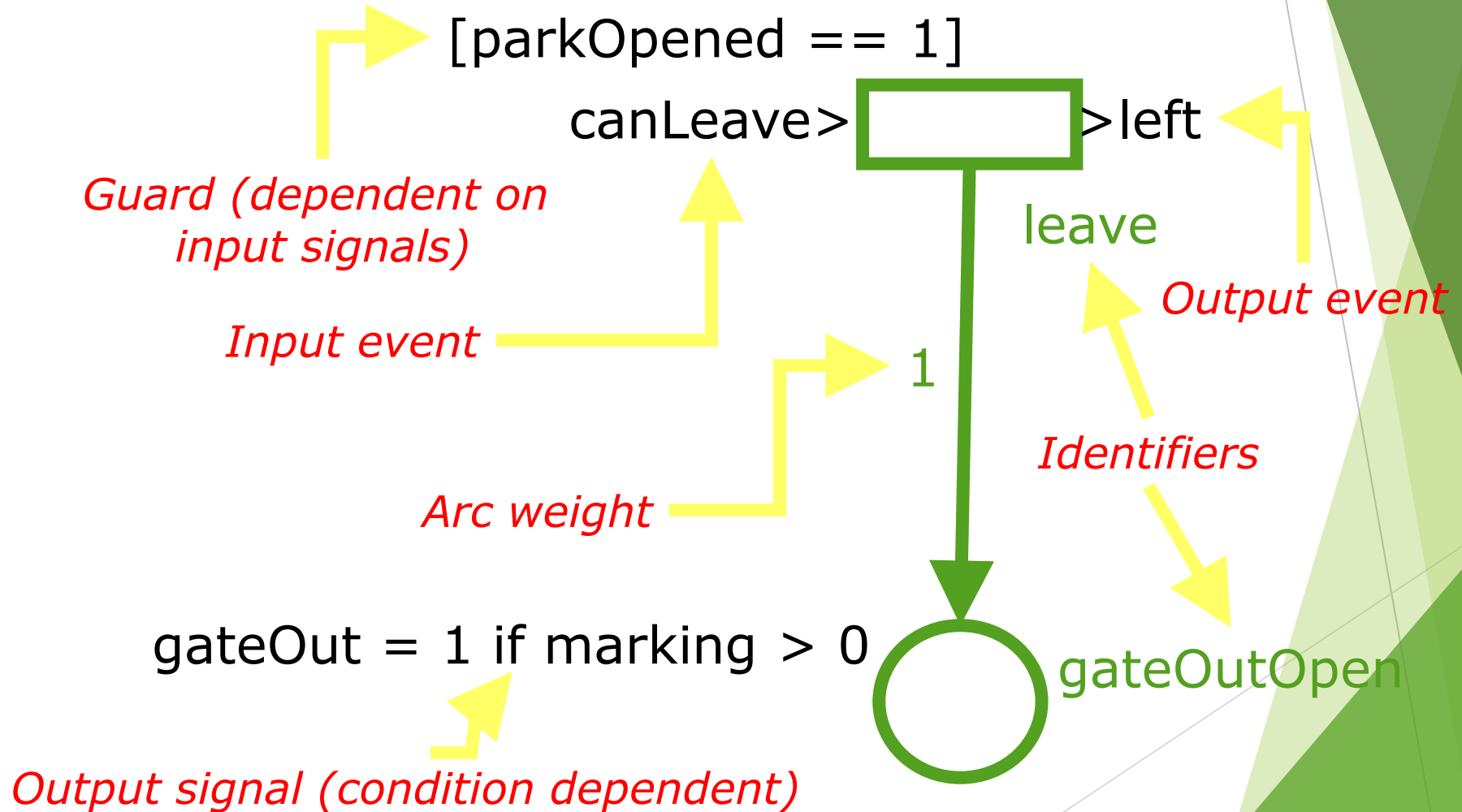
# The Input-Output Place-Transition Petri net class (IOPT nets)

- ▶ Extended from the Place/Transition net class with non-autonomous dependencies:
  - ▶ Input and output signals, Input and output events
  - ▶ Transition firing conditioned by input events and guard function constrained by input signals
  - ▶ Transition firing can generate output event and/or update output signals
  - ▶ Output signals can also be associated with places
  - ▶ Introduction of time domains and communication channels
  - ▶ Includes transition priorities and Test arcs



# IOPT – Input-Output Place-Transition nets

(Syntax)

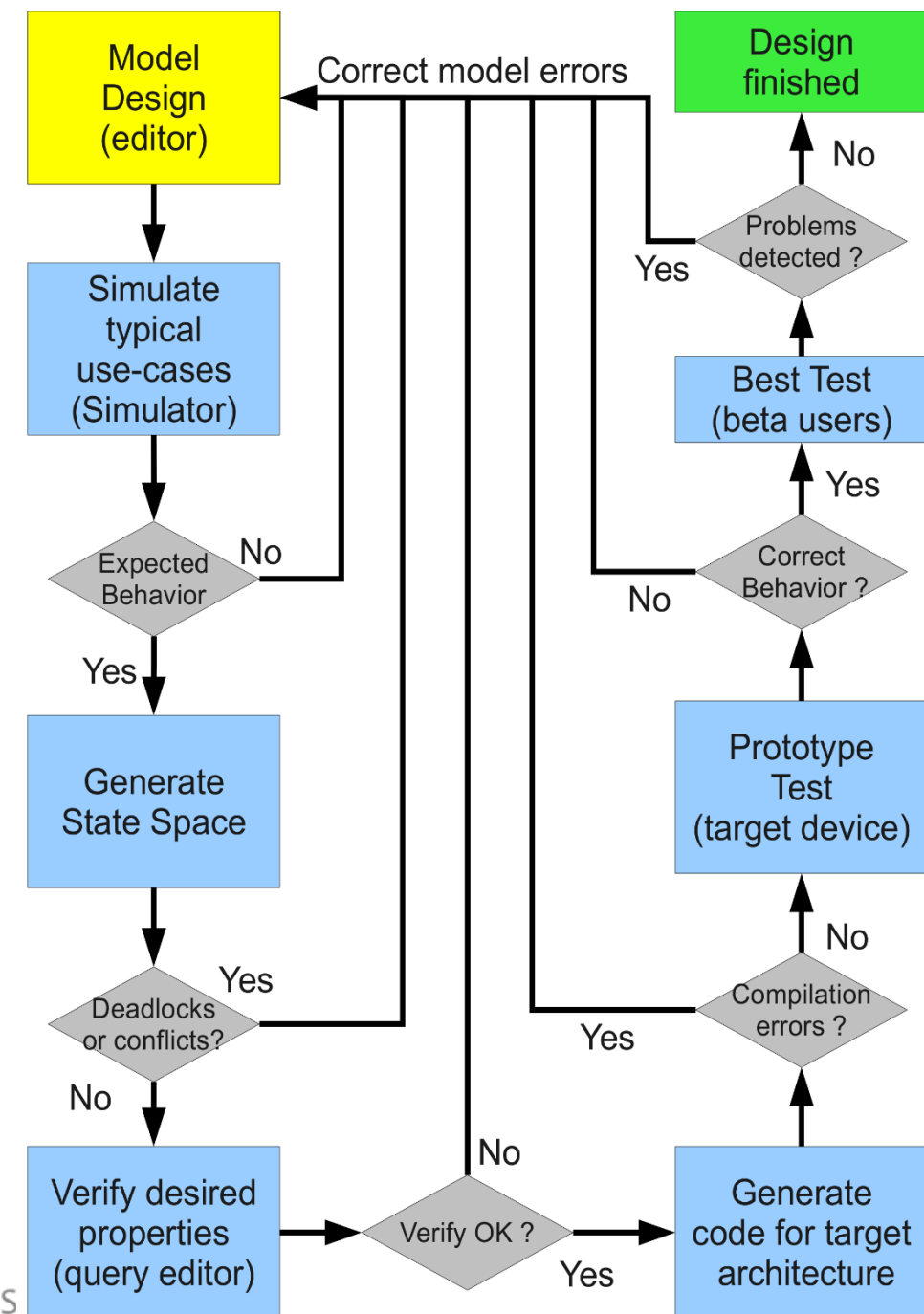


# The IOPT-Tools - a cloud-based framework

- ▶ Petri nets already have a set of supporting tools mostly covering specification and verification.
- ▶ However, Petri nets need additional tools, mostly covering automatic code generation, to be fully integrated in engineering development flows.
- ▶ A contribution using IOPT nets is available at <http://gres.uninova.pt/IOPT-Tools>
- ▶ IOPT-Tools is an integrated development tool framework covering the whole phase of embedded controllers development (including automatic code generation), testing (including simulation and verification) and maintenance
- ▶ IOPT-Tools have been extensively validated within engineering courses at NOVA University of Lisbon (and used by others)

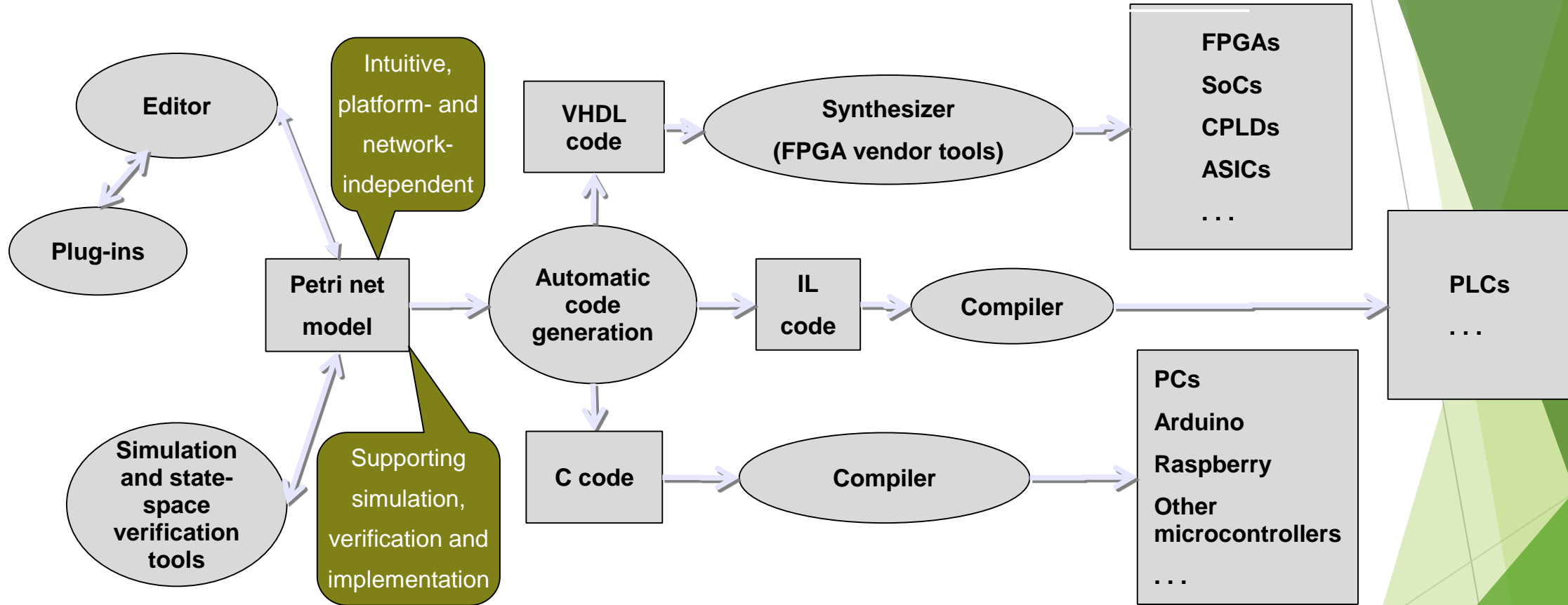
# IOPT-Tools cloud-based framework

- ▶ Tools are offered under a cloud-based user interface
- ▶ Web User Interface (<http://gres.uninova.pt/IOPT-Tools/>)
- ▶ AJAX Based IOPT Petri Net Editor
- ▶ Simulation
- ▶ Remote Debugger
- ▶ State Space Generation Tool
- ▶ Model-checking using a Query System
- ▶ Automatic controller C code generator
- ▶ Automatic controller VHDL hardware synthesis
- ▶ Automatic IL code generator for PLCs



# Development flow

Demo video available at <https://goo.gl/MxFHti>

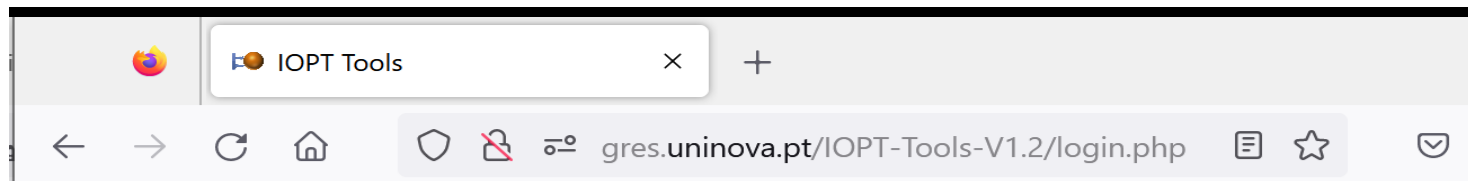


**Design effort**

**Automatic code generation**

**Implementation platforms**

# IOPT-Tools entry page



## IOPT Tools

### User Login:

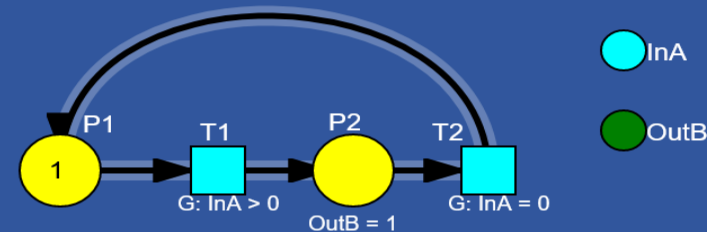
Username:

Password:

Login

New User

Cancel



Copyright (C) 2012-22 GRES Research Group

NOTES: Example models available under username "**models**" and password "**models**".

Anonymous users can login into the system with username "**guest**" and password "**guest**", to create new IOPT models and test all system functionalities. However, these models will be openly accessible to all users and may be modified by other users or deleted at any time. Therefore, creating a personal user account [free] is highly recommended.

For information about the tools please download the [User Manual](#).

[IOPT-Tools](#) have been developed by several members of the R&D Group on Reconfigurable and Embedded Systems ([GRES](#)). At current development phase, some changes and improvements will occur in the near future. Comments or requests can be directed to [gres@uninova.pt](mailto:gres@uninova.pt)

**Important note: IOPT-Tools require the latest Browser versions: Firefox >= 10, Chrome >= 12, Safari, Opera**

# IOPT-Tools Overview

**IOPT Tools (akc)** [Logout](#)  
Net file: wagon\_1\_2.pnml

Net: wagon\_12

Legend:

- GO (blue arrow), BACK (blue arrow), A1 (cyan arrow), A2 (cyan arrow), B1 (cyan arrow), B2 (cyan arrow), M1 (green arrow), M2 (green arrow), DIR1 (green arrow), DIR2 (green arrow)

Model actions:

Start new model:

Upload model file:  Nenhum ficheiro selecionado.

Important note: IOPT-Tools require the latest Browser versions: Firefox >= 10, Chrome >= 12, Safari, Opera

# Editor

The screenshot displays the Petri net editor interface for a Petri net titled "Net: wagon\_12". The interface includes a toolbar on the left with various editing tools, a central workspace for the Petri net, and a "Place Properties" panel on the right.

**Legend:**

- GO (blue arrow), BACK (blue arrow), A1 (blue arrow), A2 (blue arrow), B1 (blue arrow), B2 (blue arrow), M1 (green arrow), M2 (green arrow), DIR1 (green arrow), DIR2 (green arrow)

**Petri Net Structure:**

- Places (Yellow Circles):** Four places with capacity 12: "Carl\_Ready 12", "Car1\_Move\_Forward", "Car2\_Ready 12", and "Car2\_Move\_Forward". Two places with capacity 1: "M1 = 1 DIR1 = 0" and "M2 = 1 DIR2 = 0".
- Transitions (Blue Squares):** "GO" (G: GO = 1), "A1" (G: A1 = 1), "B1" (G: B1 = 1), "A2" (G: A1 = 1), "B2" (G: B2 = 1), and "BACK" (G: BACK = 1).
- Transitions (Red Squares):** "M1 = 1 DIR1 = 1" and "M2 = 1 DIR2 = 1".

**Transitions and Connections:**

- "GO" is connected to "Carl\_Ready 12", "Car1\_Move\_Forward", "Car2\_Ready 12", and "Car2\_Move\_Forward".
- "A1" is connected to "Carl\_Ready 12" and "M1 = 1 DIR1 = 1".
- "B1" is connected to "Carl\_Ready 12" and "M1 = 1 DIR1 = 1".
- "A2" is connected to "Car2\_Ready 12" and "M2 = 1 DIR2 = 1".
- "B2" is connected to "Car2\_Ready 12" and "M2 = 1 DIR2 = 1".
- "M1 = 1 DIR1 = 1" is connected to "Carl\_Move\_Back" and "M1 = 1 DIR1 = 0".
- "M2 = 1 DIR2 = 1" is connected to "Car2\_Move\_Back" and "M2 = 1 DIR2 = 0".
- "BACK" is connected to "M1 = 1 DIR1 = 0" and "M2 = 1 DIR2 = 0".

**Place Properties Panel (Right):**

- ID: 77\_20
- Name: Car1\_Move\_Forward
- Initial Marking: 0
- Bound: 1
- Time Domain: [empty]
- Comment: 12
- Port: [dropdown]
- Output: M1 = [dropdown]
- Action 1: 1
- When: [empty]
- Output: DIR1 = [dropdown]
- Action 2: 1
- When: [empty]
- Output: [dropdown]
- Action 3: [empty]
- When: [empty]
- Buttons: Save, Cancel

**Bottom Status Bar:**

- Select place 77\_20
- X: 430 Y: 760
- Grid: [checked] Snap: [checked] Mod.I/O: [checkbox]



### Transition Properties:

ID:

Name:

Priority:

Guard:

Input Events:   
Multiple Selection Shift/Ctrl Key

Output Events:   
Multiple Selection Shift/Ctrl Key

Action 1:

Time Domain:

Port:

Comment:

### Place Properties:

ID:

Name:

Initial Marking:

Bound:

Time Domain:

Comment:

Port:

Output Action 1:   
When:

Output Action 2:   
When:

### Signal Properties:

Name/ID:

Mode:

Type:

Value:

Min:

Max:

Physical I/O Nr:

Used in transition-12:tr\_12, transition-13:tr\_13;

### Event Properties:

ID:

Mode:

Autonomous:

Edge:

Level:

Signal:

Used by transitions tr-10:got\_ticket;

### Arc Properties:

ID:

Type:

Inscription:



# Space state generator

va.pt/IOPT-Tools-V1.2/ss\_init\_marking.php

## State-space Generator Initial Marking Editor

Model wagon\_12.pnml

Generate State Space

Cancel

Place:

Marking:

Car1\_Move\_Back [2]: 0

Car1\_Ready [3]: 1

Car1\_Arrived [4]: 0

Car2\_Arrived [5]: 0

Car1\_Move\_Forward [6]: 0

Car2\_Ready [12]: 1

Car2\_Move\_Back [18]: 0

Car2\_Move\_Forward [20]: 0

gres.uninova.pt/IOPT-Tools-V1.2/ss\_progress.php

User: akc Model: wagon\_12

Close

View Graph

Download File

Update Bounds

File size: 0.000 Mb

```
Cycle 1: 1 states + 0 links  
Cycle 2: 2 states + 0 links  
Cycle 3: 5 states + 0 links  
Cycle 4: 6 states + 2 links  
Cycle 5: 8 states + 3 links
```

```
MIN Bounds: Car1_Arrived=0 Car1_Move_Back=0 Car1_Move_Forward=0 Car1_Ready=0 Car2_Arrived=0 Car2_Move_Back=0  
Car2_Move_Forward=0 Car2_Ready=0
```

```
MAX Bounds: Car1_Arrived=1 Car1_Move_Back=1 Car1_Move_Forward=1 Car1_Ready=1 Car2_Arrived=1 Car2_Move_Back=1  
Car2_Move_Forward=1 Car2_Ready=1
```

```
#####  
Total States: 8  
Total Links: 5  
#####
```

```
Executing queries...  
Done: found 0 query matching states.
```

```
Generation time (sec): 0.00 (when 0.00 it is smaller than 0.01sec)
```

```
Generating output file.  
Done.
```

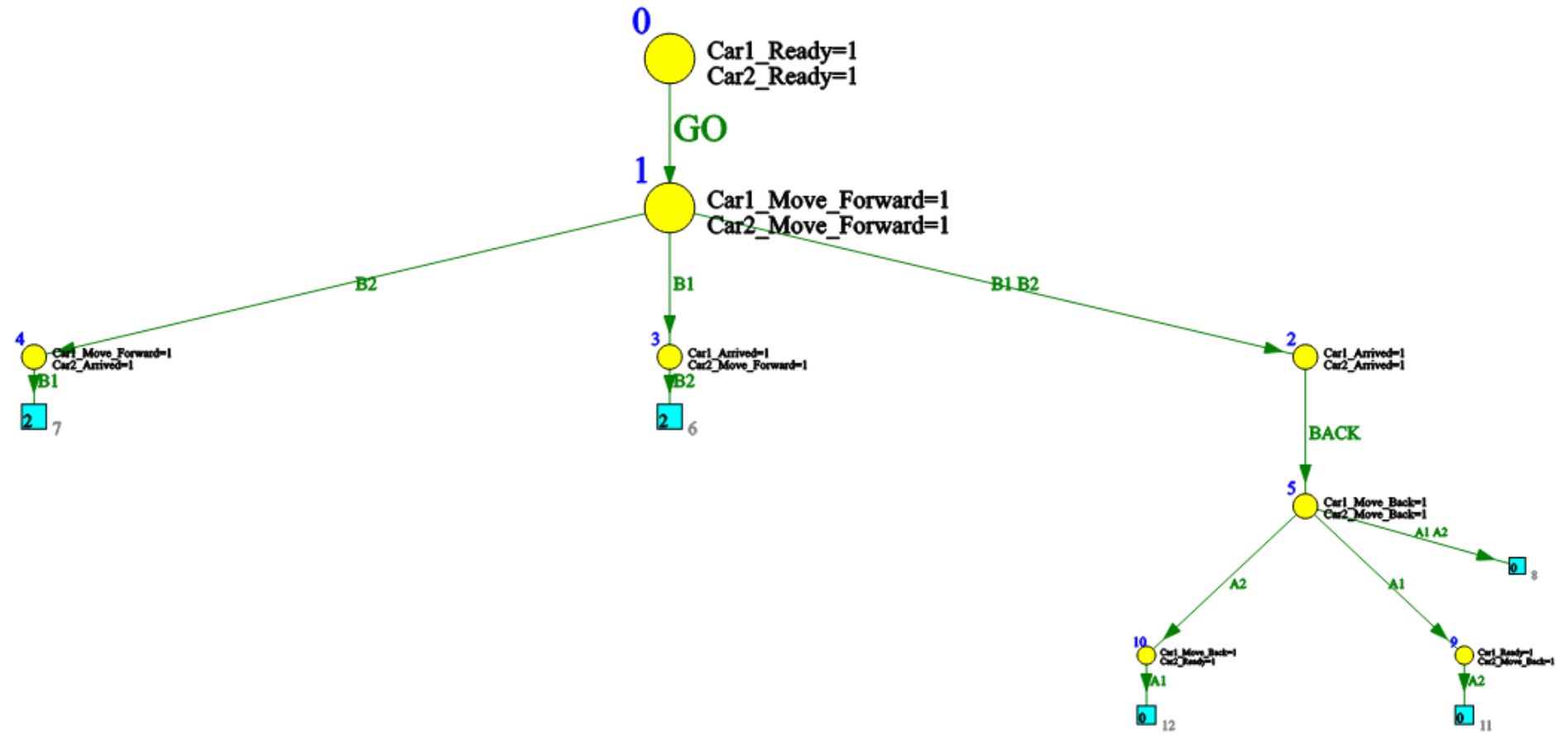
```
#####  
Total States: 8  
Total Links: 5  
Deadlock count: 0  
Conflict count: 0  
Invalid count: 0  
#####
```

### Net wagon\_12

8 (from 8) Nodes, 5 Loops, 0 Deadlocks, 0 Conflicts, Max. Depth = 6, 0 Invalid

Min Bound = [Car1\_Arrived=0 Car1\_Move\_Back=0 Car1\_Move\_Forward=0 Car1\_Ready=0 Car2\_Arrived=0 Car2\_Move\_Back=0 Car2\_Move\_Forward=0 Car2\_Ready=0]

Max Bound = [Car1\_Arrived=1 Car1\_Move\_Back=1 Car1\_Move\_Forward=1 Car1\_Ready=1 Car2\_Arrived=1 Car2\_Move\_Back=1 Car2\_Move\_Forward=1 Car2\_Ready=1]



# Query editor

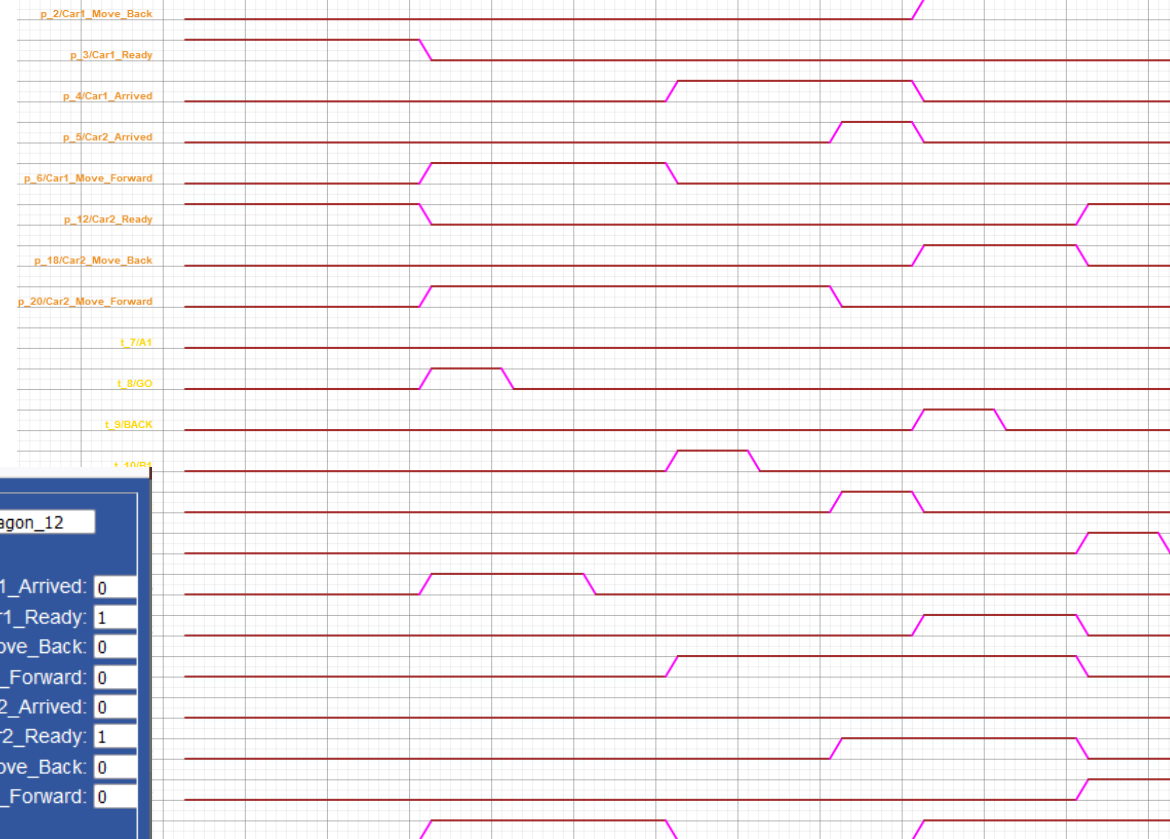
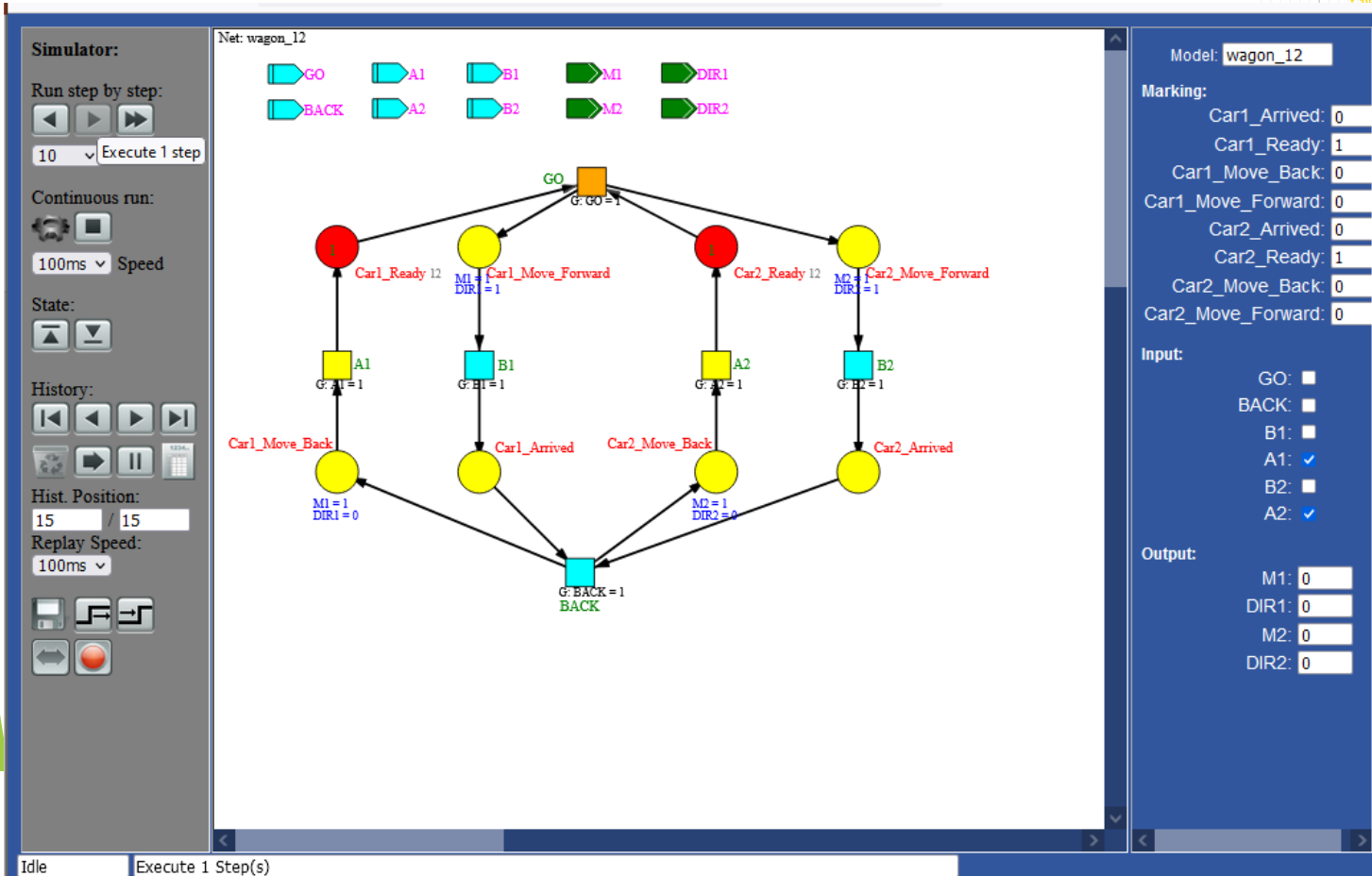
The screenshot shows the 'State-space Query Editor' web interface. At the top, the browser address bar shows 'gres.uninova.pt/IOPT-Tools-V1.2/query.php'. The main header is 'State-space Query Editor' with the model name 'Model wagon\_12.pnml'. Below this, there are several configuration sections: 'Values:' with dropdowns for 'Places', 'Transitions', and 'Event Output Signals'; 'Operators:' with dropdowns for 'Comparison', 'Arithmetic', and 'Logic', and a 'Sub-expression' dropdown; 'Reachability:' with a text input 'State [ ] is reachable'; and 'Edit:' with buttons for 'Erase', 'Clear All', and 'Undo'. A large text area for the 'Query:' contains the text 'Car2\_Arrived = 2'. Below the query area are 'Save' and 'Exit' buttons. At the bottom, there is a 'Query list:' section with a 'Selected query:' dropdown set to '2'. A scrollable list of queries is shown below, with the second query, 'Car2\_Arrived = 2', selected.

# Query results

The screenshot shows the 'Query Results' web interface. The browser address bar shows 'gres.uninova.pt/IOPT-Tools-V1.2/query\_res.php'. The main header is 'Found 1 matching query results for model wagon\_12'. Below this is a 'Summary:' section with a table showing the results for 'Query 1 ( Car1\_Arrived AND Car2\_Arrived )', which has '1 matching states'. At the bottom, there are radio buttons for 'Sort by Query' and 'Sort by State', along with 'Show Results' and 'Exit' buttons.

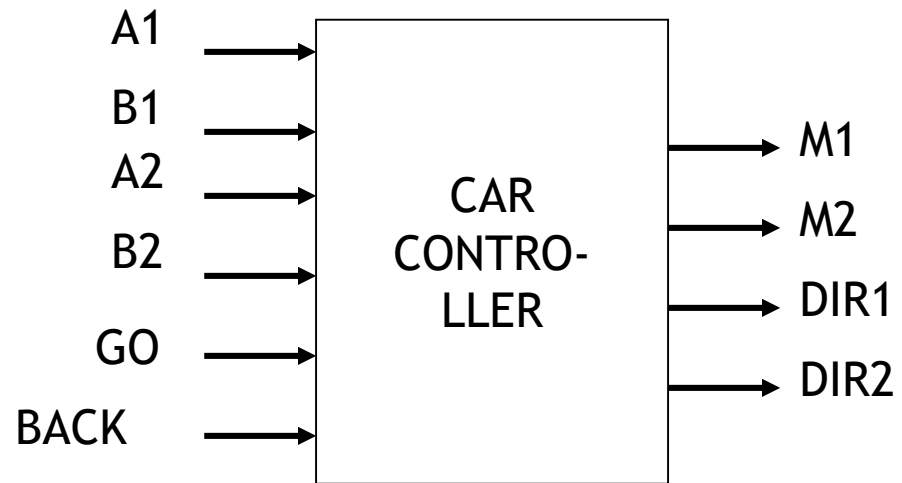
# IOPT-Tools - Simulator

## Token player



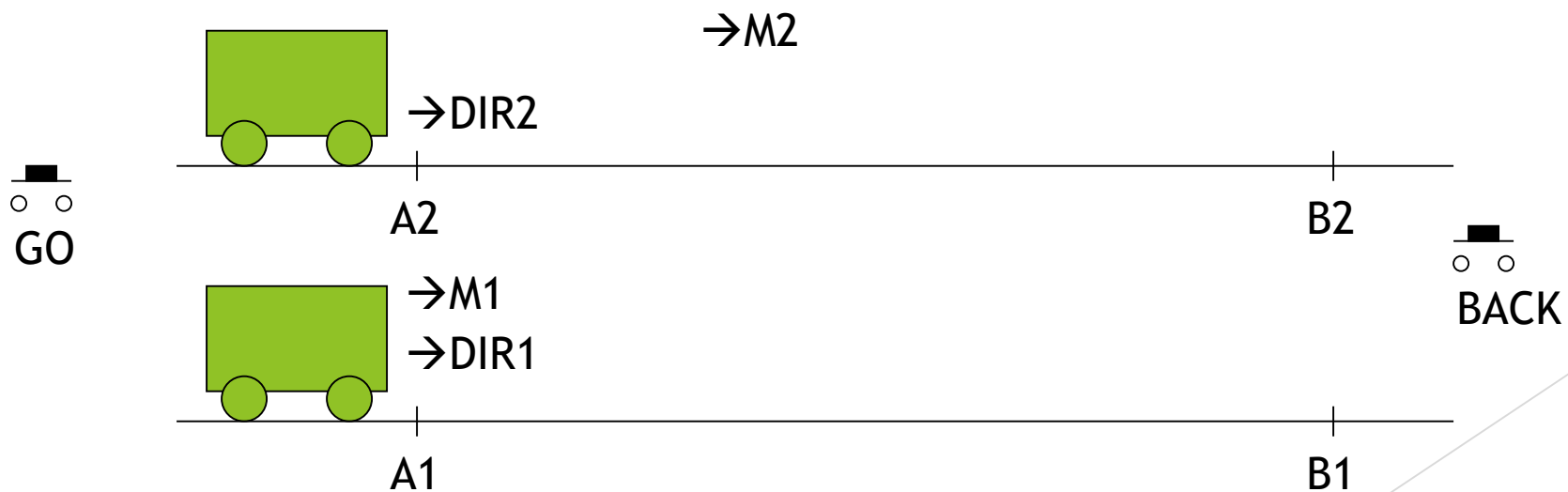
## Timing diagram

# An example

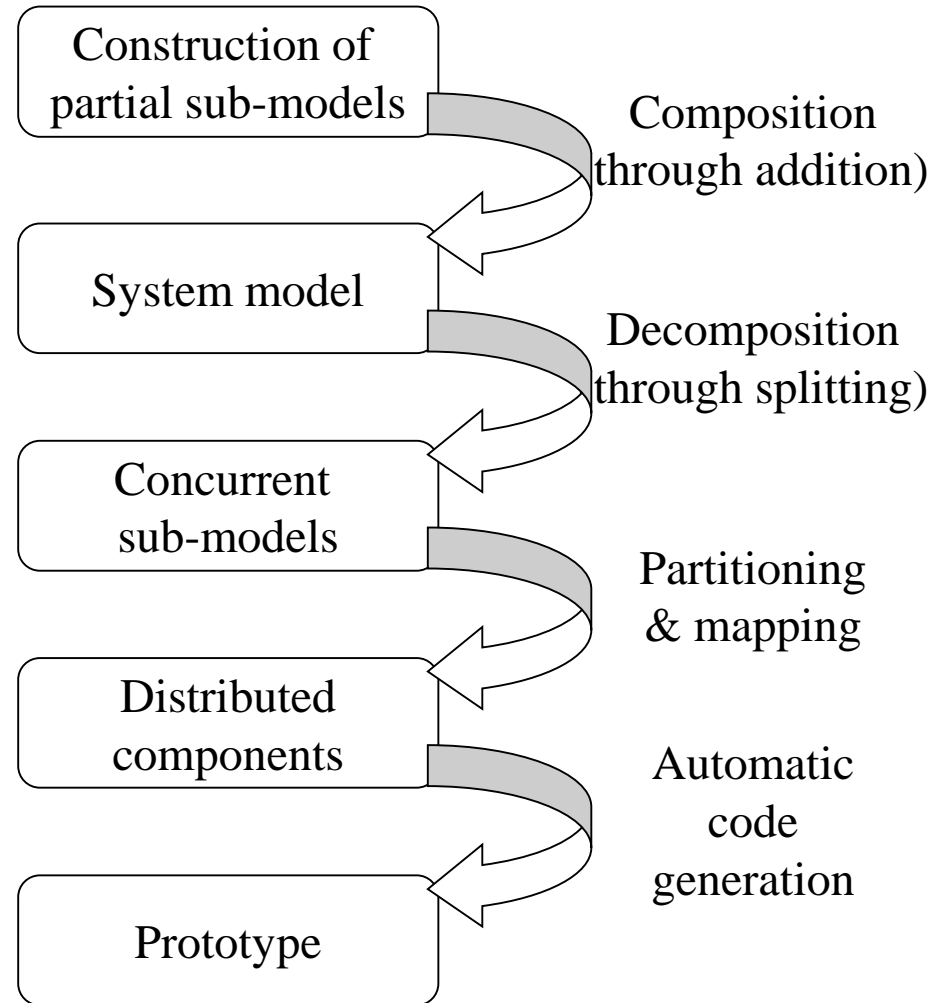


## Goal:

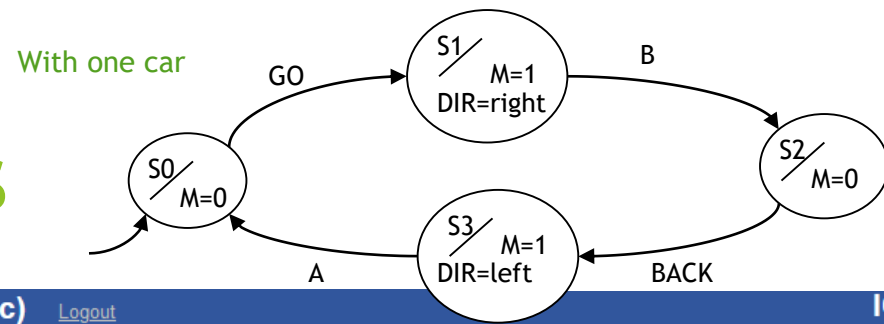
To model the behaviour of a two car transportation system; Cars are synchronized at the beginning and at the end.



# Underlying methodology



# Partial models



**IOPT Tools (akc)** Logout  
Net file: wagon\_1.pnml

Car1 Controller

**IOPT Tools (akc)** Logout  
Net file: wagon\_2.pnml

Car2 Controller

[Edit Model](#) [Simulator](#) [Debugger](#) [Generate State Space](#) [Query Editor](#) [Query Results](#) [C Code](#) [VHDL Code](#) [IL Code](#) [Simulink Code](#) [Model](#) [Download Model File](#) [Export P/T Net](#) [Decompose GALS](#) [HIPPO](#) [Model List](#) [User Manual](#)

[Model](#) [Simulator](#) [Debugger](#) [Generate State Space](#) [Query Editor](#) [Query Results](#) [C Code](#) [VHDL Code](#) [IL Code](#) [Simulink Code](#) [Model](#) [Download Model File](#) [Export P/T Net](#) [Decompose GALS](#) [HIPPO](#) [Model List](#) [User Manual](#)

**Model actions:**

Start new model:  [Create](#)

Upload model file:  Nenhum ficheiro selecionado. [Upload](#)

**Model actions:**

Start new model:  [Create](#)

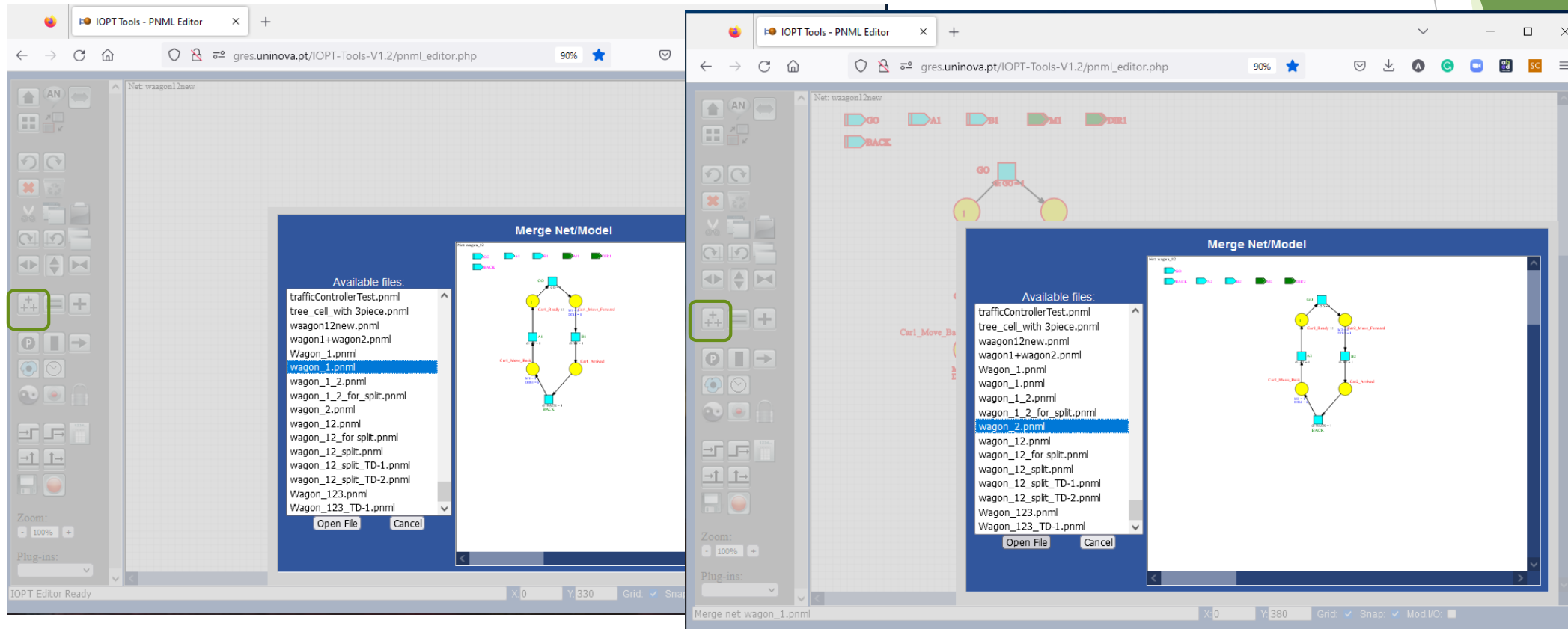
Upload model file:  Nenhum ficheiro selecionado. [Upload](#)

Important note: IOPT-Tools require the latest Browser versions: Firefox >= 10, Chrome >= 12, Safari, Opera



# Model composition

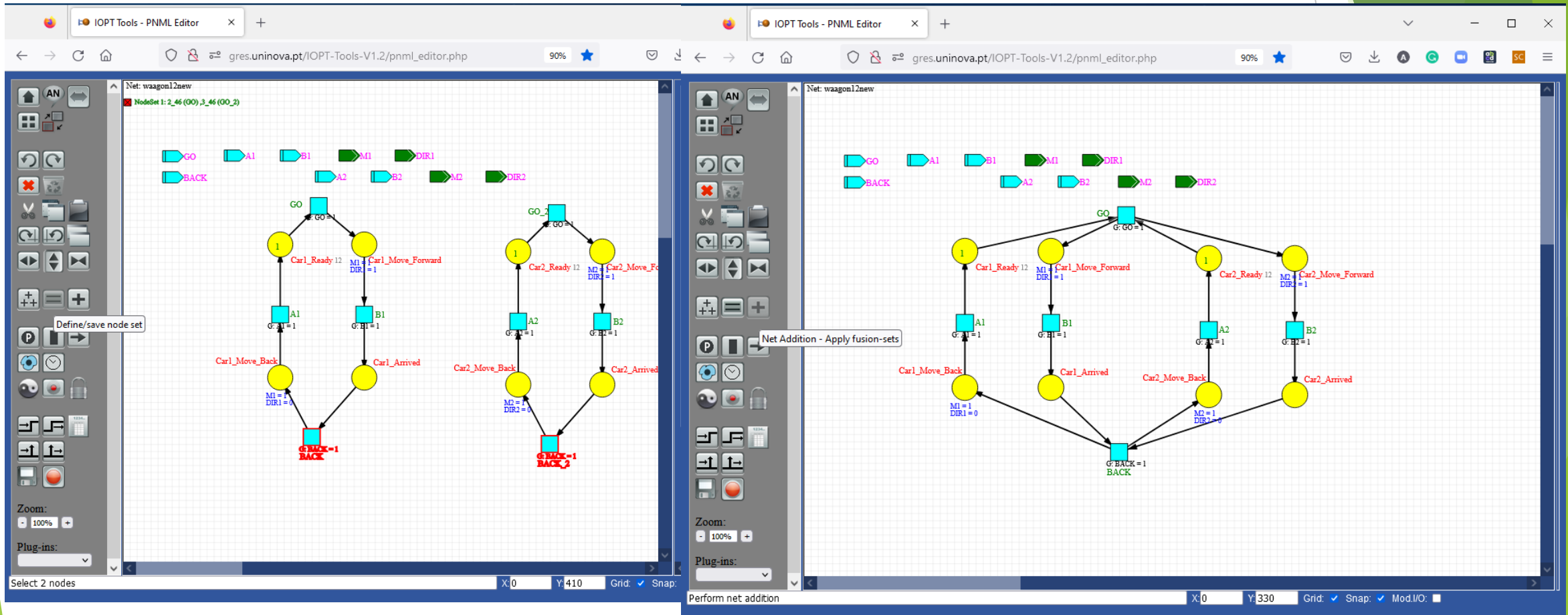
## Net Merging



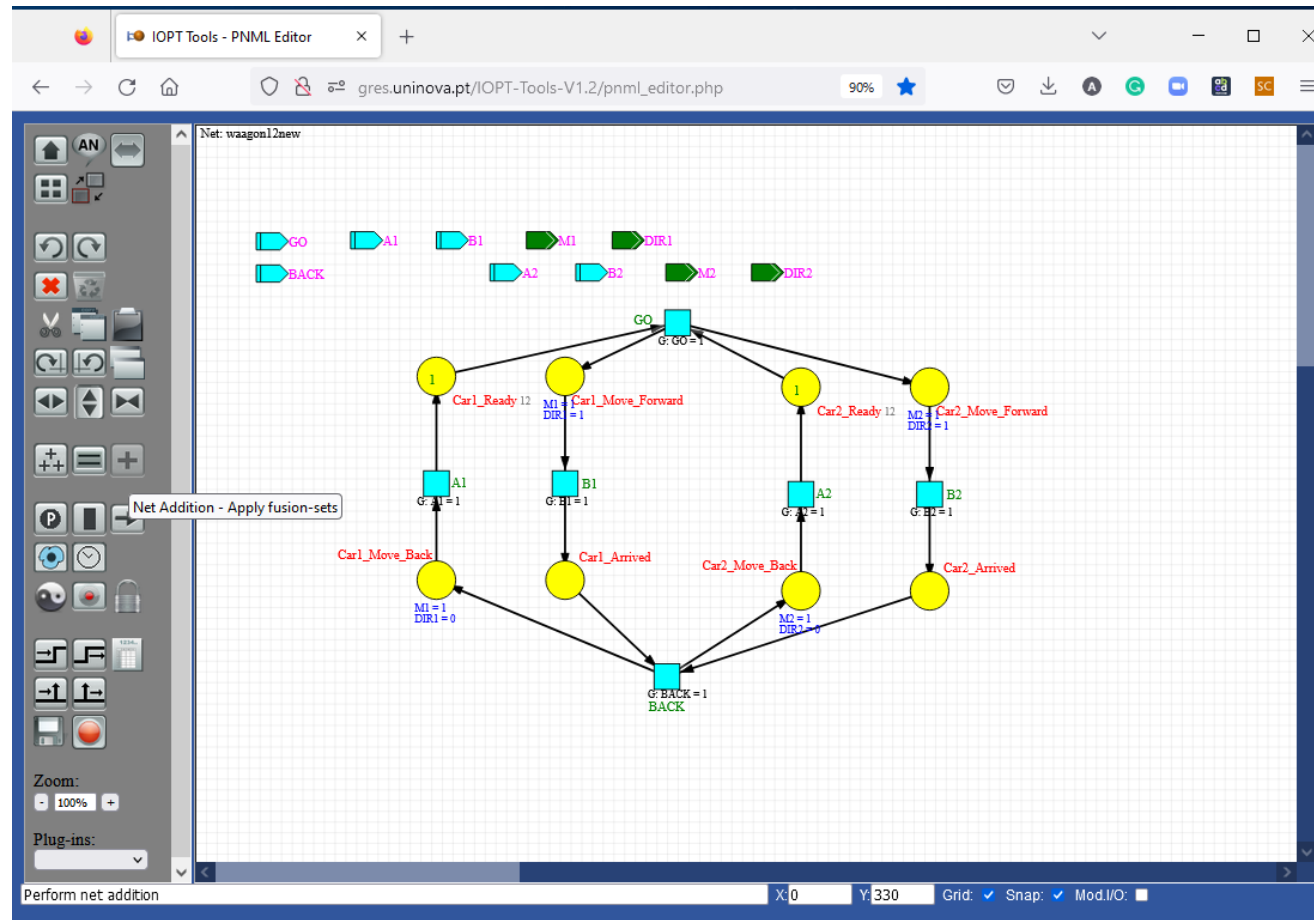


# Model composition

## Node fusion



# System model - results of net addition



# Space state generator

ges.uninova.pt/IOPT-Tools-V1.2/ss\_init\_marking.php

## State-space Generator Initial Marking Editor

Model wagon\_12.pnml

Generate State Space

Cancel

Place:

Marking:

Car1\_Move\_Back [2]: 0

Car1\_Ready [3]: 1

Car1\_Arrived [4]: 0

Car2\_Arrived [5]: 0

Car1\_Move\_Forward [6]: 0

Car2\_Ready [12]: 1

Car2\_Move\_Back [18]: 0

Car2\_Move\_Forward [20]: 0

ges.uninova.pt/IOPT-Tools-V1.2/ss\_progress.php

User: akc Model: wagon\_12

Close View Graph Download File Update Bounds File size: 0.000 Mb

```
Cycle 1: 1 states + 0 links
Cycle 2: 2 states + 0 links
Cycle 3: 5 states + 0 links
Cycle 4: 6 states + 2 links
Cycle 5: 8 states + 3 links

MIN Bounds: Car1_Arrived=0 Car1_Move_Back=0 Car1_Move_Forward=0 Car1_Ready=0 Car2_Arrived=0 Car2_Move_Back=0
Car2_Move_Forward=0 Car2_Ready=0

MAX Bounds: Car1_Arrived=1 Car1_Move_Back=1 Car1_Move_Forward=1 Car1_Ready=1 Car2_Arrived=1 Car2_Move_Back=1
Car2_Move_Forward=1 Car2_Ready=1

#####
Total States: 8
Total Links: 5
#####

Executing queries...
Done: found 0 query matching states.

Generation time (sec): 0.00 (when 0.00 it is smaller than 0.01sec)

Generating output file.
Done.

#####
Total States: 8
Total Links: 5
Deadlock count: 0
Conflict count: 0
Invalid count: 0
#####
```

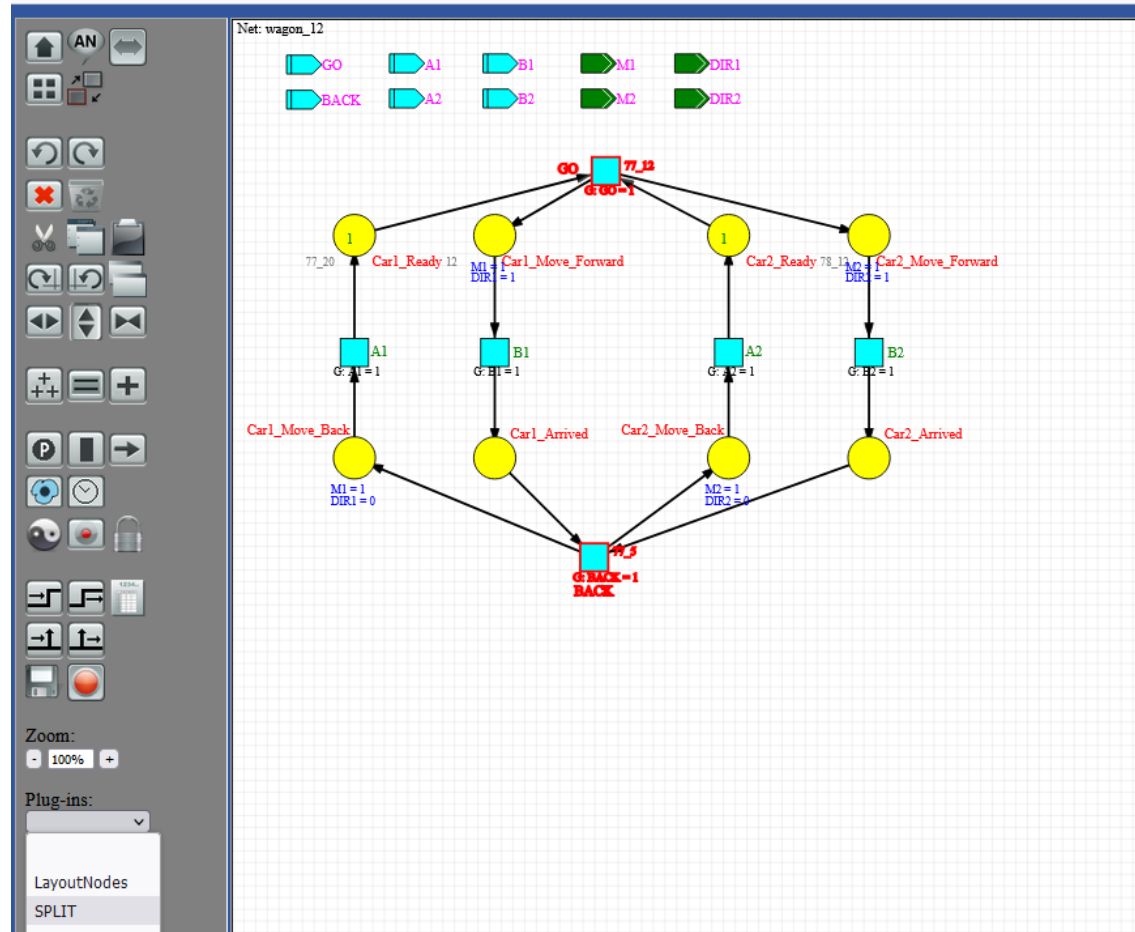
# Query editor

The screenshot shows the 'State-space Query Editor' web interface. At the top, the browser address bar shows 'gres.uninova.pt/IOPT-Tools-V1.2/query.php'. The main header is 'State-space Query Editor' with the model name 'Model wagon\_12.pnml' to its right. Below the header, there are several configuration sections: 'Values:' with dropdowns for 'Places', 'Transitions', and 'Event Output Signals'; 'Operators:' with dropdowns for 'Comparison', 'Arithmetic', and 'Logic', and a 'Sub-expression' dropdown; 'Reachability:' with a text input 'State [ ] is reachable'; and 'Edit:' with buttons for 'Erase', 'Clear All', and 'Undo'. A large text area labeled 'Query:' contains the text 'Car2\_Arrived = 2'. Below the query area are 'Save' and 'Exit' buttons. At the bottom, there is a 'Query list:' section with a 'Selected query:' dropdown set to '2'. The list contains five items: '1 Car1\_Arrived AND Car2\_Arrived', '2 Car2\_Arrived = 2', '3', '4', and '5'. Item 2 is selected.

# Query results

The screenshot shows the 'Query Results' web interface. The browser address bar shows 'gres.uninova.pt/IOPT-Tools-V1.2/query\_res.php'. The main header is 'Found 1 matching query results for model wagon\_12'. Below the header is a 'Summary:' section with a table. The table has two columns: the first column contains 'Query 1 ( Car1\_Arrived AND Car2\_Arrived )' and the second column contains '1 matching states'. At the bottom, there are radio buttons for 'Sort by Query' and 'Sort by State', and buttons for 'Show Results' and 'Exit'.

# Component decomposition - Net Splitting



**Plug-in: SPLIT**

Instructions: To use the Net Splitting tool go through the following steps: 1. In the PNML Editor select a set of nodes in which the splitting operation will occur (Cutting Set). (NOTE: A selected node must not have any arc linking it to another selected node and the removal of all the selected nodes should generate atleast two subnets). 2. The user can choose on which subnet the master transition of a selected cutting set transition element will be located by filling it's "Comment" field with the "id" of another net element that isn't a cutting set element. The "id" inserted must from an element that belongs to the intended subnet generated after the removal of the cutting set elements. 3 To merge two or more subnets, select a non-cutting set node of a subnet and fill it's "Comment" field with the "id" of an element that belongs to another subnet (this last one mustn't also be a cutting set node). 4. By selecting "Synchronous" or "Assynchronous" on "ImpType" (Implementation type) the SPLIT tool will generate the resulting subnets communication with the respective channel types.

ImpType: Synchronous

Apply - Cancel



# Net splitting

IOPT Tools (akc) [Logout](#)  
Net file: wagon\_1\_2\_for\_split.pnml

Net: wagon\_12

Legend:

- GO (blue arrow), A1 (blue arrow), B1 (blue arrow), M1 (green arrow), DIR1 (green arrow)
- BACK (blue arrow), A2 (blue arrow), B2 (blue arrow), M2 (green arrow), DIR2 (green arrow)

Buttons:

Edit Model | Simulator | Debugger | Generate State Space | Query Editor | Query Results | C Code | VHDL Code | IL Code | Simulink Code  
Download Model File | Export P/T Net | **Decompose GALS** | HIPPO | Model List | User Manual

Model actions:

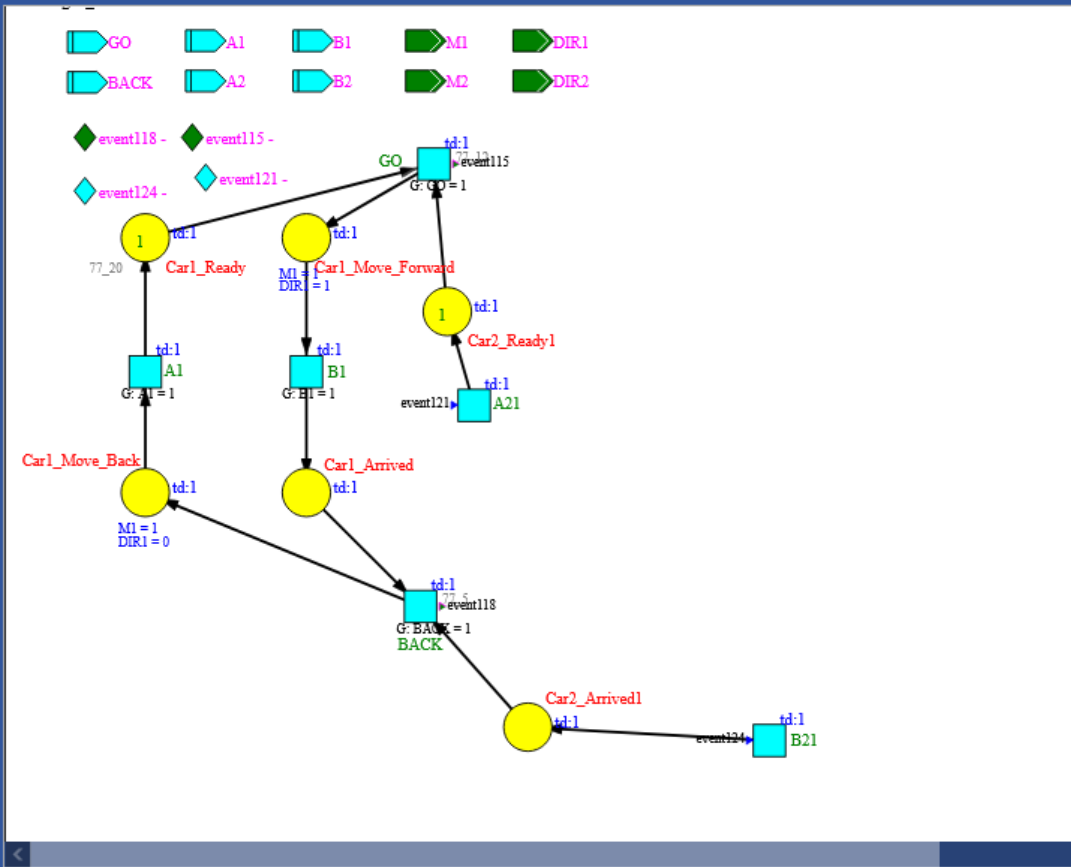
Start new model:  Create

Upload model file:  Nenhum ficheiro selecionado. Upload

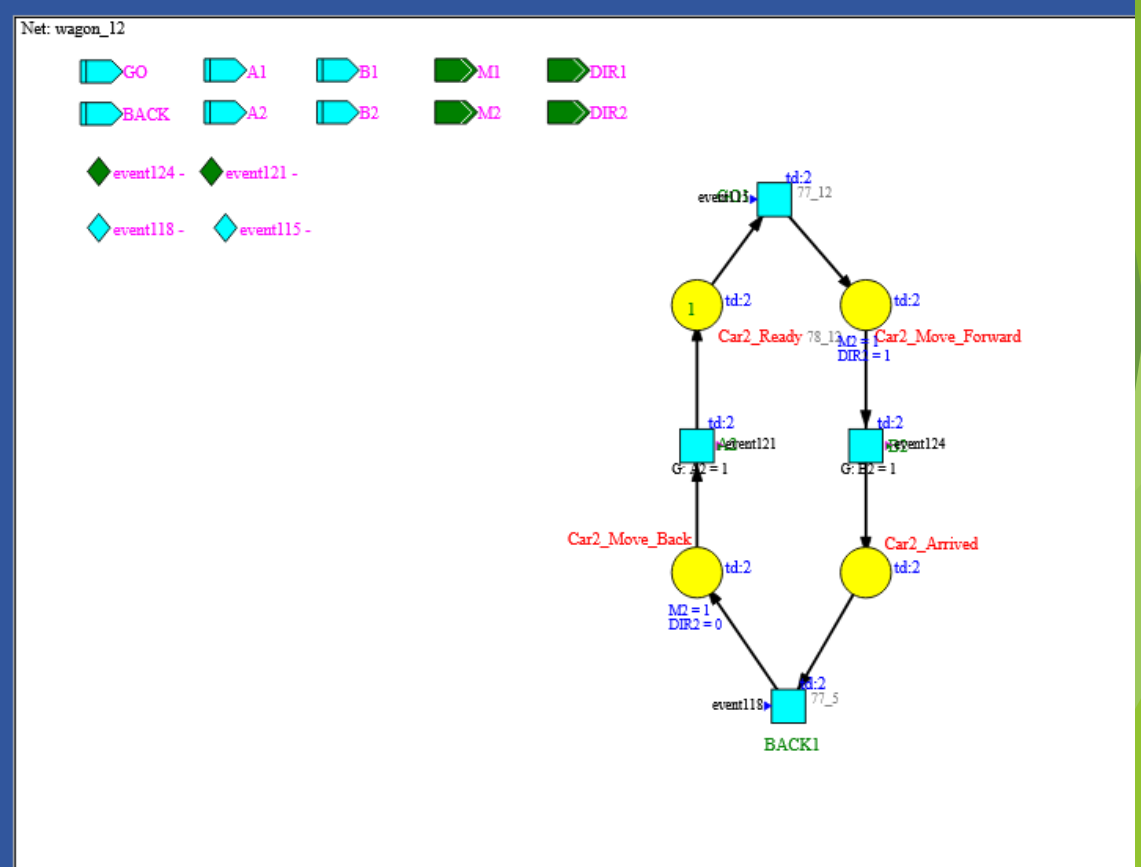
Important note: IOPT-Tools require the latest Browser versions: Firefox >= 10, Chrome >= 12, Safari, Opera

# Distributed component

IOPT Tools (akc) [Logout](#)  
Net file: wagon\_1\_2\_for\_split\_TD-1.pnml

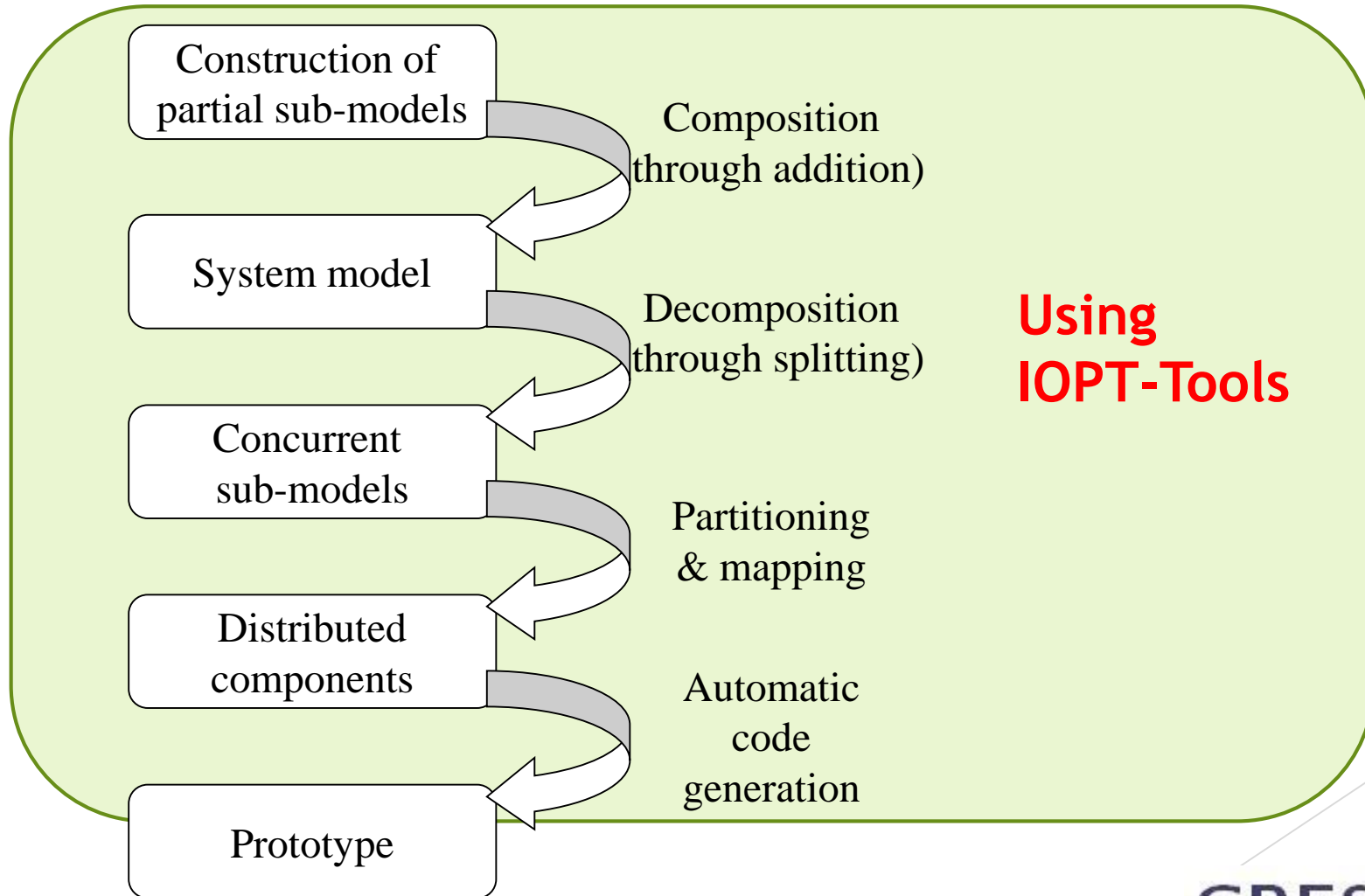


IOPT Tools (akc) [Logout](#)  
Net file: wagon\_1\_2\_for\_split\_TD-2.pnml



Edit Model Simulator Debugger Generate State Space Query Editor Query Results C Code VHDL Code IL Code Simulink Code  
Download Model File Export P/T Net Decompose GALS HIPPO Model List User Manual

# Underlying methodology





# IOPT-Tools - Remote Debugger

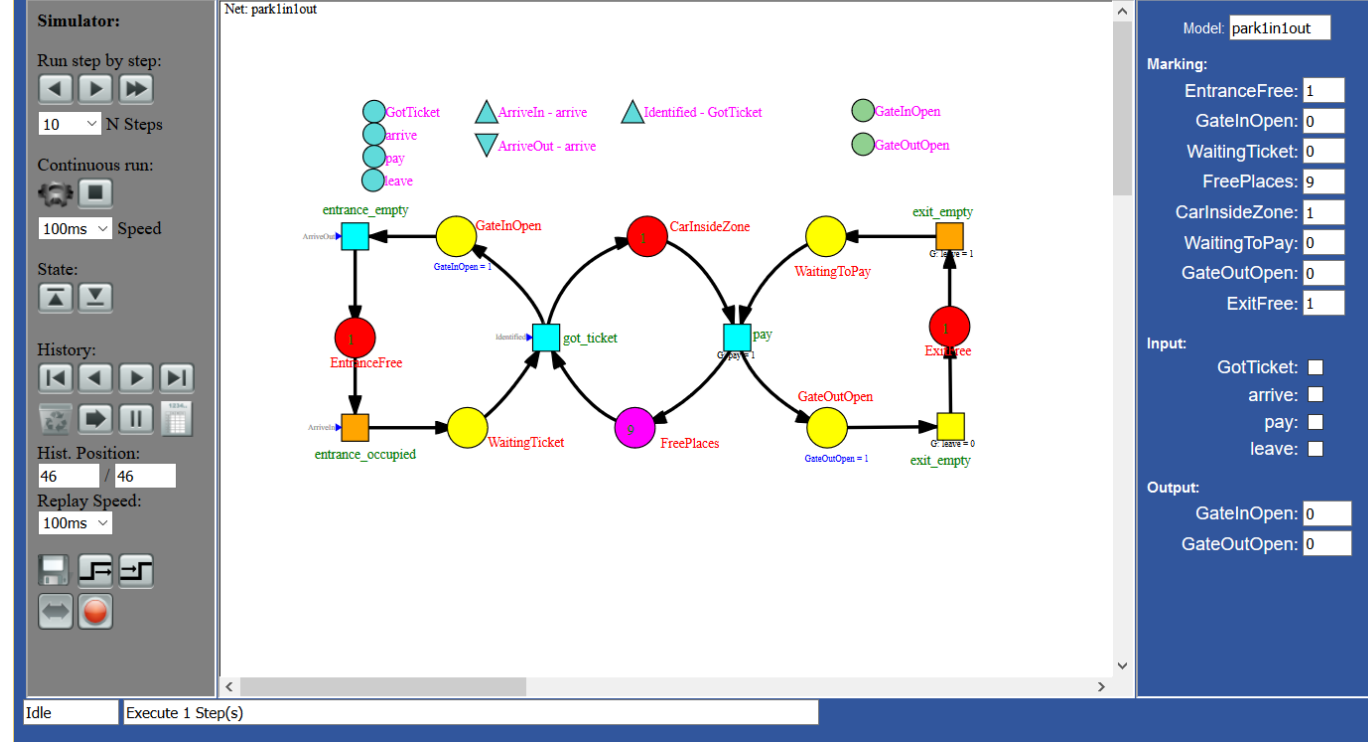
- ▶ Communication architecture to enable the remote control, monitoring and debug of embedded system controllers designed using IOPT Petri nets.
- ▶ The architecture adds Internet connectivity capabilities to the controllers produced by the automatic code generators, enabling online remote debugging and monitoring using the IOPT simulator tool.
- ▶ Furthermore, it enables the creation of web based graphical user interfaces for remote operation and the development of distributed systems where a Petri net model running on a central system supervises the actions of multiple remote subsystems.

# IOPT-Tools - Remote Debugger

Usage of a minimalist HTTP server,  
Implemented directly on the  
controller code.

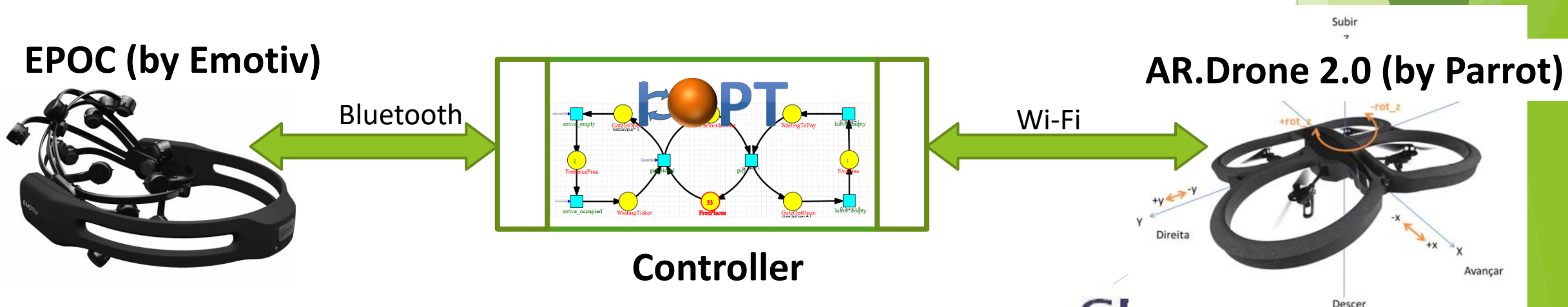
Supporting:

- Status monitoring functions,
- A tracing mechanism with step-by-step execution and breakpoints definition,
- The capability to remotely force the value of input and output signals, used to implement hardware-in-the-loop solution where the simulator takes full control of the physical embedded devices.



# Application example

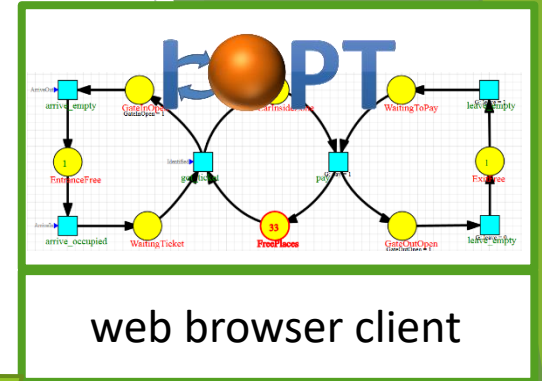
- ▶ An IOPT-based controller receiving signals from an EEG signals acquisition system and actuating movements of a quadcopter.



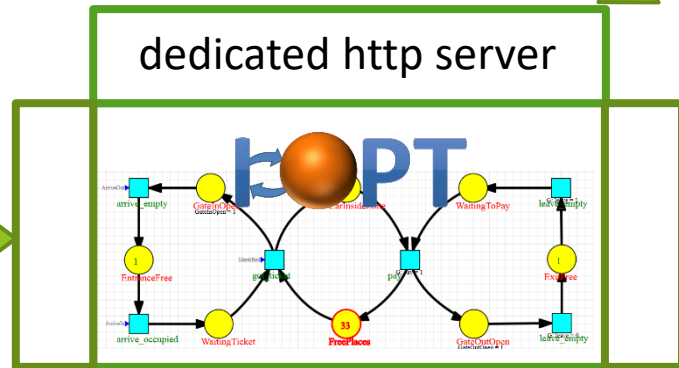
# Application example

Adding remote monitoring and control.

## Remote debugger



internet



Controller

EPOC (by Emotiv)



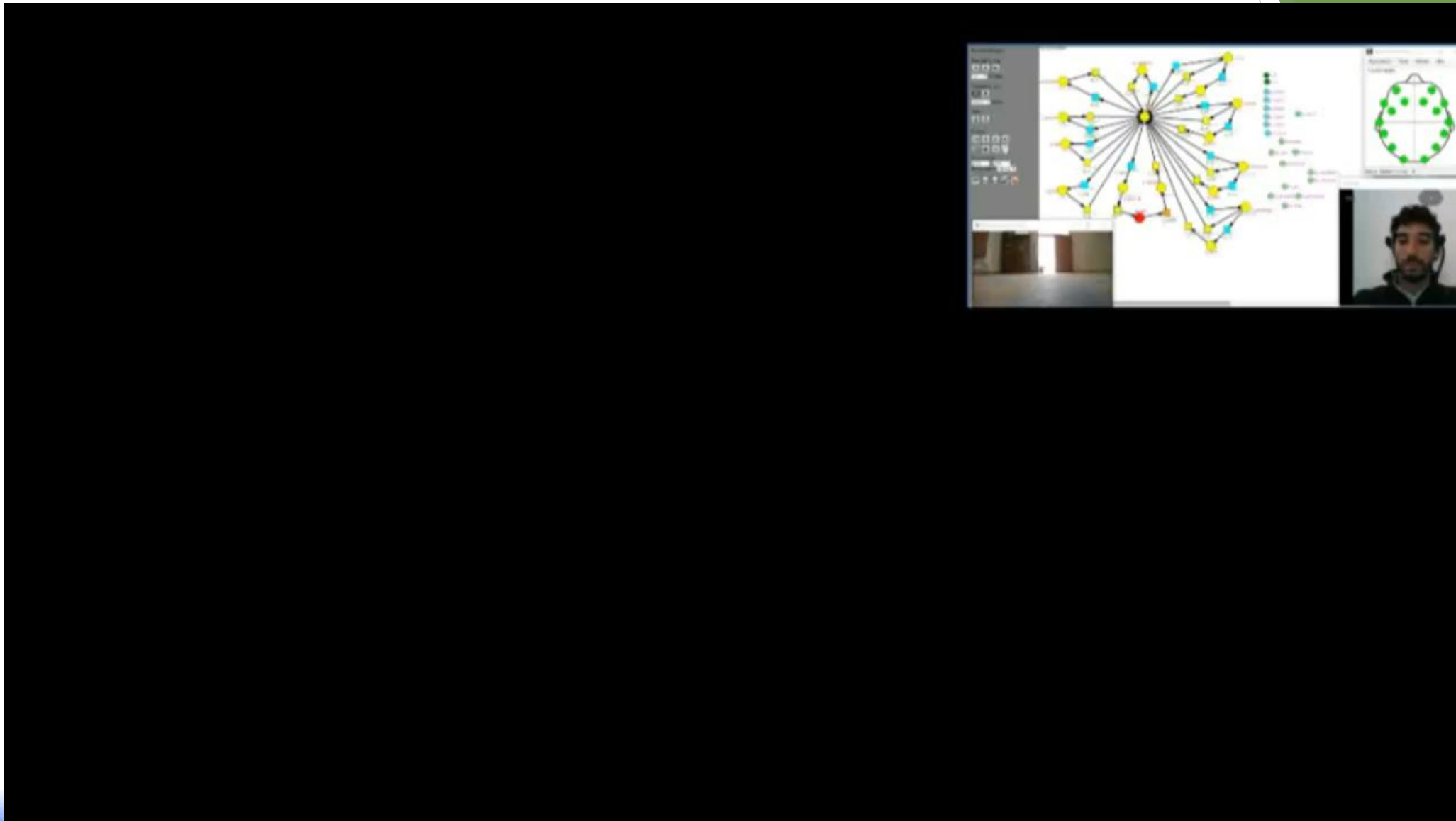
Bluetooth

AR.Drone 2.0 (by Parrot)

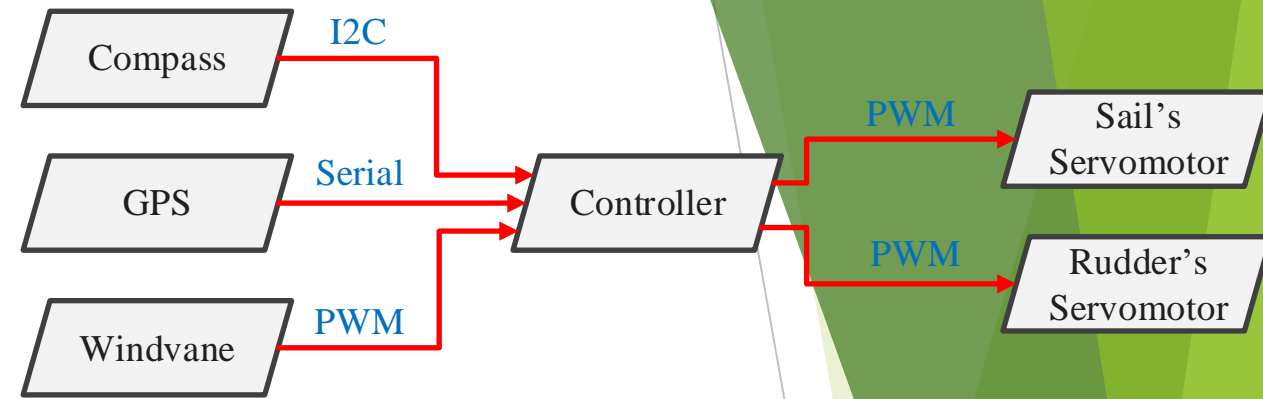


Wi-Fi

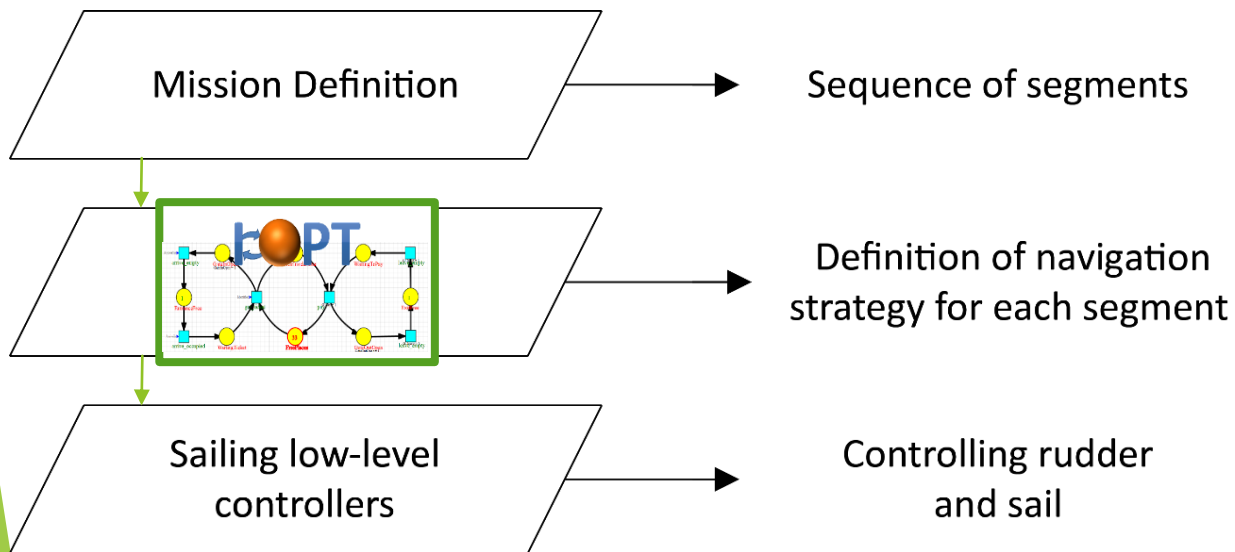
# On-line control with remote monitoring



# Other application example



## ► Autonomous navigation of a sailboat





# Conclusions

- ▶ In the years ahead it is expected the appearance of millions of new Internet aware embedded devices, both for existing applications and for applications yet to be discovered.
- ▶ This way, the development tools for embedded and cyber-physical systems will need to offer rapid prototyping as well as the support for remote operation, monitoring, debug, troubleshoot and diagnose problems on malfunction devices.
- ▶ Model-based development and Petri nets have an important role to play.
- ▶ IOPT-Tools have been successful used to developed embedded controllers, targeting both software and hardware platforms.