Software Praktikum
- Projects overview-

Stefan Resmerita
stefan.resmerita@sbg.ac.at

Winter 2019

http://www.softwareresearch.net/ -> teaching -> software-praktikum

Workflow

- Teamwork
- Regular meetings (personal and virtual)
- Regular code reviews
- Version control (github repos)
- Evaluation
  - Midterm presentation : 12.12.2019
  - Final project submitted by 15.02.2020

Software development

- Model-based design
  - Build from models
- Actor-oriented design
  - More than OO
- Reusability (Frameworks)
  - Open source
- Testing
- Build automation
- Continuous integration

Tools

- Eclipse (Java)
  - OSGi platform, Apache Commons plugins
- Matlab/Simulink (Java, C)
  - Modeling->simulation->code generation->testing
- Validator (C)
  - Platform-aware testing
- Ptolemy II (Java)
  - Actor-oriented design
- Maven
  - Build management
- Jenkins
  - Continuous integration

Overview of projects

- **R1**: Jenkins plugin for the Validator (Java, 2)
- **R2**: Maven for Eclipse Plugin Development (Java, 2-3)
- **R3**: Co-simulation Matlab/Simulink-Validator (C, 2)
- **R4**: Eclipse GUI plugin for consistency testing (Java, 2)
- **R5**: Eclipse GUI extended (R4 companion, 2)
- **R6**: Code generation from Finite State Machines (FSM) (Java,XML,C, 2-3)
- **R7**: Eclipse GUI for FSM editing (R6 companion, 2)
- **R8**: Modeling and simulation of an intersection controller in Ptolemy II (Java, 2)
- **R9**: Modeling and simulation of a parking garage for autonomous vehicles in Ptolemy II (Java, 2)

R1: Jenkins plugin

- https://jenkins.io/ - Install and configure system on local machine
- Create a custom plugin to configure and run simulations with the Validator
  - Command with launch arguments...
- Interpret and display simulation results in Jenkins
  - What to display
  - Which format
  - Advanced: interpret trends in test results

R2: Maven for Eclipse Plugin Development

- Managing of large Eclipse plugin structures
- Dependencies between various plugins
- Dependencies to 3rd party libraries (e.g., Apache Commons, …) from plugins
- https://maven.apache.org/
- https://www.eclipse.org/tycho/ (still active?)
- Create dummy Eclipse plugins according to a given setup
- Create Maven scripts to manage builds and 3rd party libs.
- Advanced: Integrate Code Obfuscation as final step of build (ProGuard)
  - https://www.guardsquare.com/en/products/proguard

R3: Co-simulation

- Validator gets dedicated port for receiving commands and configuration
- Send configuration from Simulink to Validator
  - Format
  - Model update?
- Send commands from Simulink to the Validator
  - Init, Start, Stop, Quit command
  - Pause command?
  - Update command?
- Status updates from Validator to Simulink

R4: Eclipse GUI plugin for consistency testing

- Goal: Graphical support for the user to configure consistency sets and timing properties of software parts
  - Consistency sets contain variables
  - Timing properties contain minimum and worst case execution times
- Data is stored in an XML file
- GUI should be based on EMF
- Various options to develop a GUI in Eclipse
  - Sirius (https://www.eclipse.org/sirius)
  - GMF/GEF (https://www.eclipse.org/gmf-tooling)
  - Graphiti (https://www.eclipse.org/graphiti)

R5: Extended Eclipse GUI plugin for consistency testing

- Companion project to R4 (required)
- Run consistency test simulations from within configuration GUI
- Result analysis
- Result visualization
- Automatic reconfiguration

R6: Code generation from Finite State Machines represented in XML

- FSM specified in
  - State Chart XML (Apache Commons SCXML) or
  - Ptolemy II MOML
- Target language: C
- Target framework: Validator
  - Custom interfaces and glue code
- Methods:
  - Parse the xml (e.g., from Java), or
  - Use XSLT stylesheet and transformer (e.g., libxslt), or
  - Customize the scxmlcc

R7: Eclipse GUI for FSM editing

- Companion project to R6
  - SCXML or DOT
- Methods:
  - Use Eclipse GEF DOT - Graphviz authoring environment
  - Integrate/extend existing application, e.g., scxmlgui

R8: Intersection controller in PtolemyII

- Control of traffic lights at an intersection (+) based on sensing incoming and outgoing vehicles
- Each incoming lane has two sensors (e.g., induction coils)
  - Sensor for vehicles approaching the intersection
  - Sensor for vehicles about to enter the intersection
- Each outgoing lane has a sensor for detecting vehicles leaving the intersection
- Vehicles can be autonomous or human-driven
- Autonomous cars can communicate among them

Requirements

- When no human-driven vehicle approaches the intersection, traffic lights are always green
- Design a controller for traffic lights for the case where human-driven vehicles are also involved
- Note: automous vehicles do obey traffic lights!
- Use DE domain

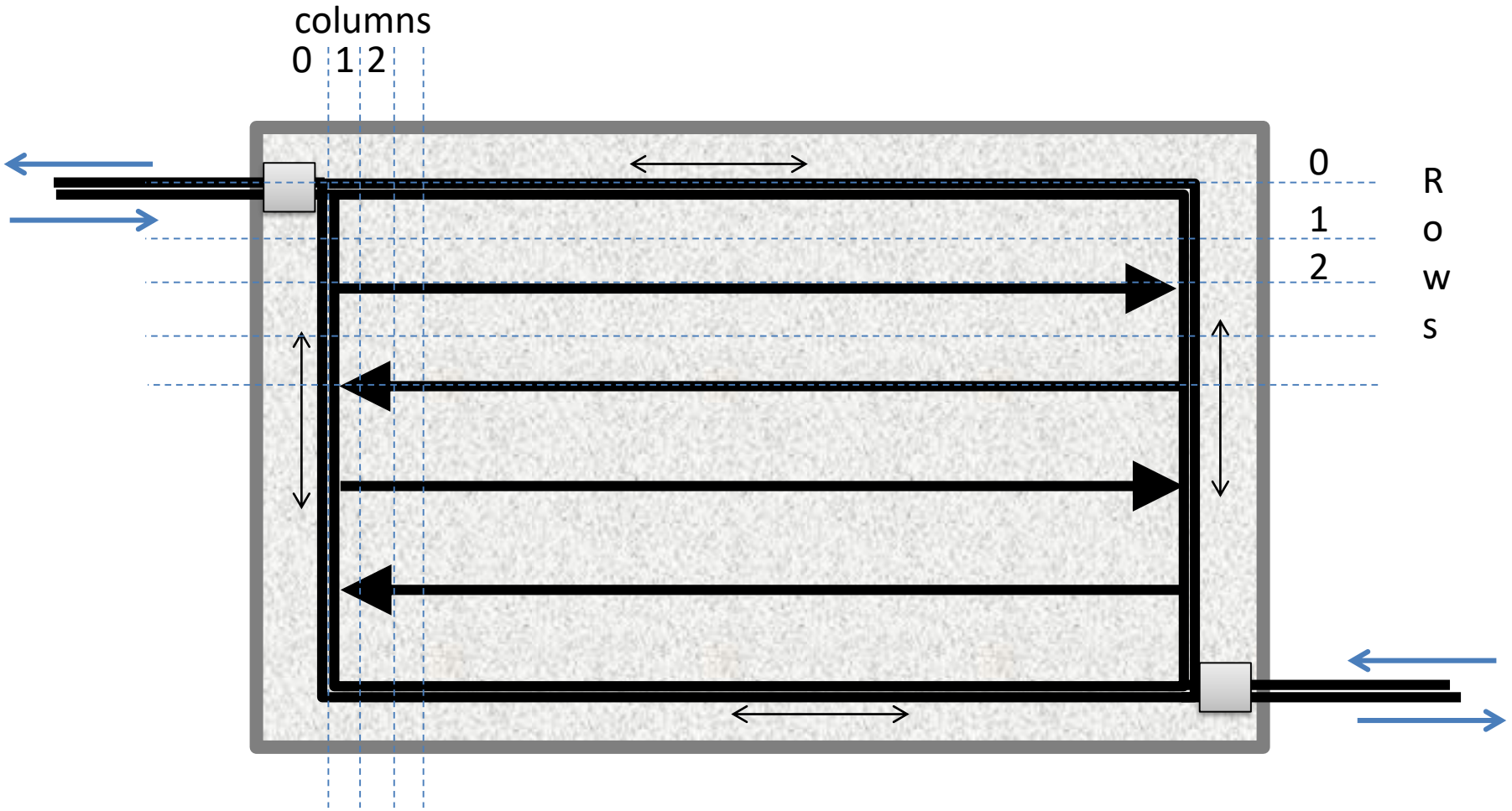
R9: Parking garage control in PtolemyII

- Autonomous cars
- Driver leaves car at the entry (gate),
- The car goes to its parking place according to instruction from the garage controller
- Driver sends time of pickup (e.g., via SMS)
- Car leaves its place, exits the garage and waits for the driver after the exit

Model behavior

- Parking garage for autonomous cars
- Driver leaves car at the entry (gate), then the car goes to its parking place according to instruction from the garage controller
- Driver sends time of pickup (e.g., via SMS)
- Car leaves its place, exits the garage and waits for the driver after the exit
- Each car has a unique id

Parking garage layout



Car behavior

- Upon entry/exit, a car receives the trajectory to follow, in the form of a sequence of waypoints and associated times:

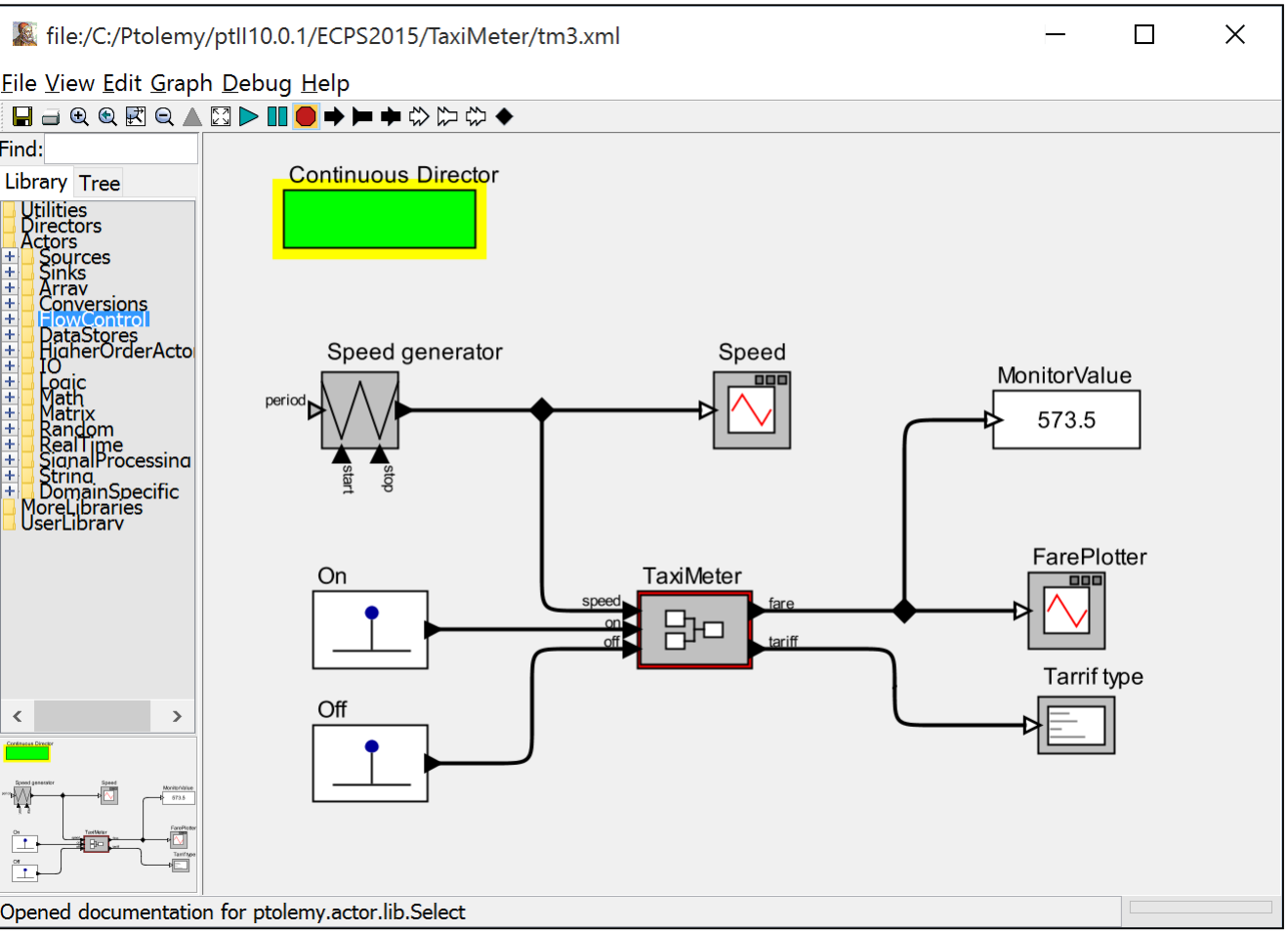
  $(p_1,t_1) \rightarrow (p_2,t_2) \rightarrow \ldots \rightarrow (p_n,t_n)$

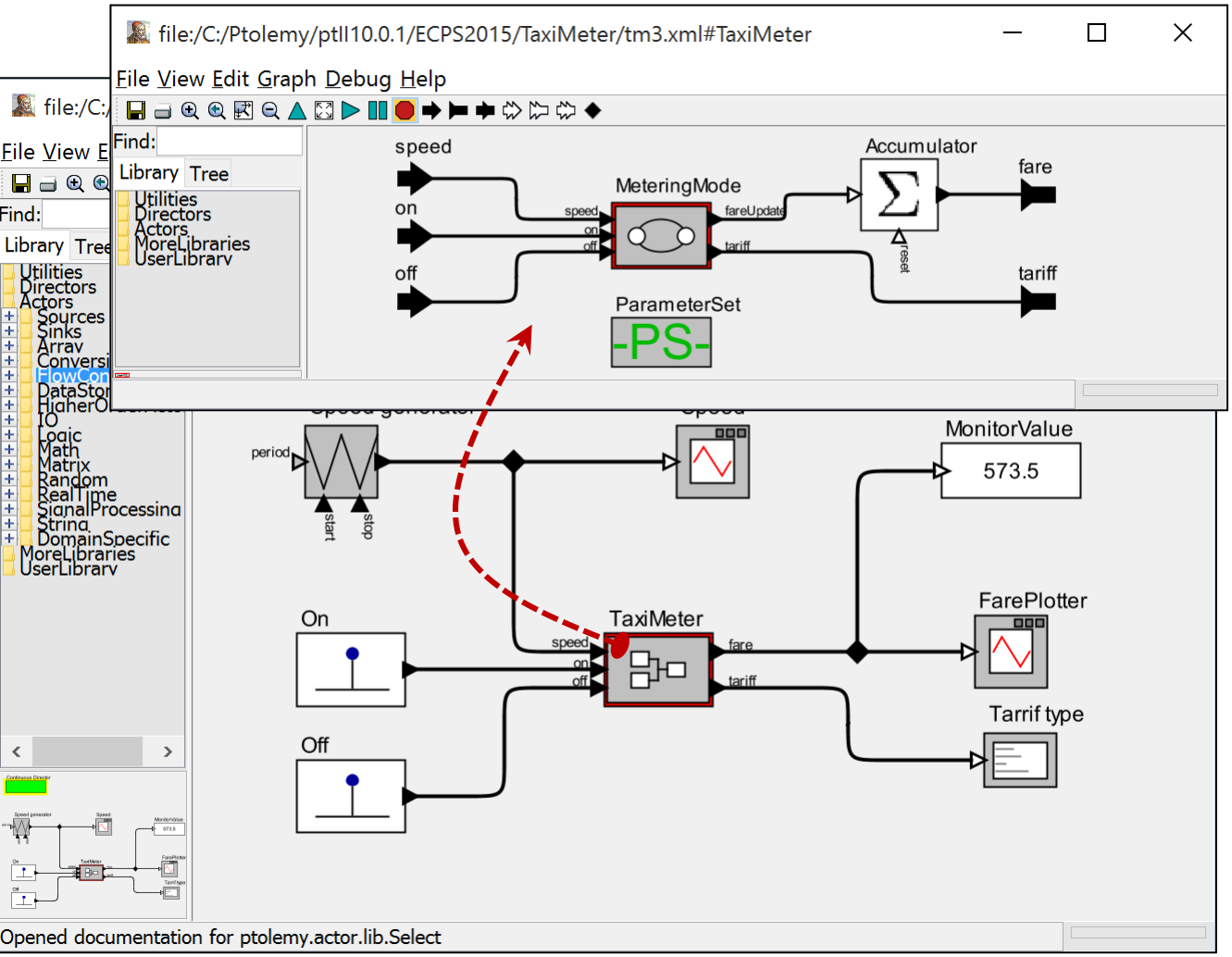
  - The car moves with constant speed in the segment $[p_i,p_{i+1})$
  - Assume instantaneous speed change possible
- The car also receives a priority number together with the trajectory
- Each car has a proximity sensor with a certain radius. If two moving cars detect a possible collision, one of them stops and informs the garaje controller. Then the controller sends back an updated trajectory and priority number.

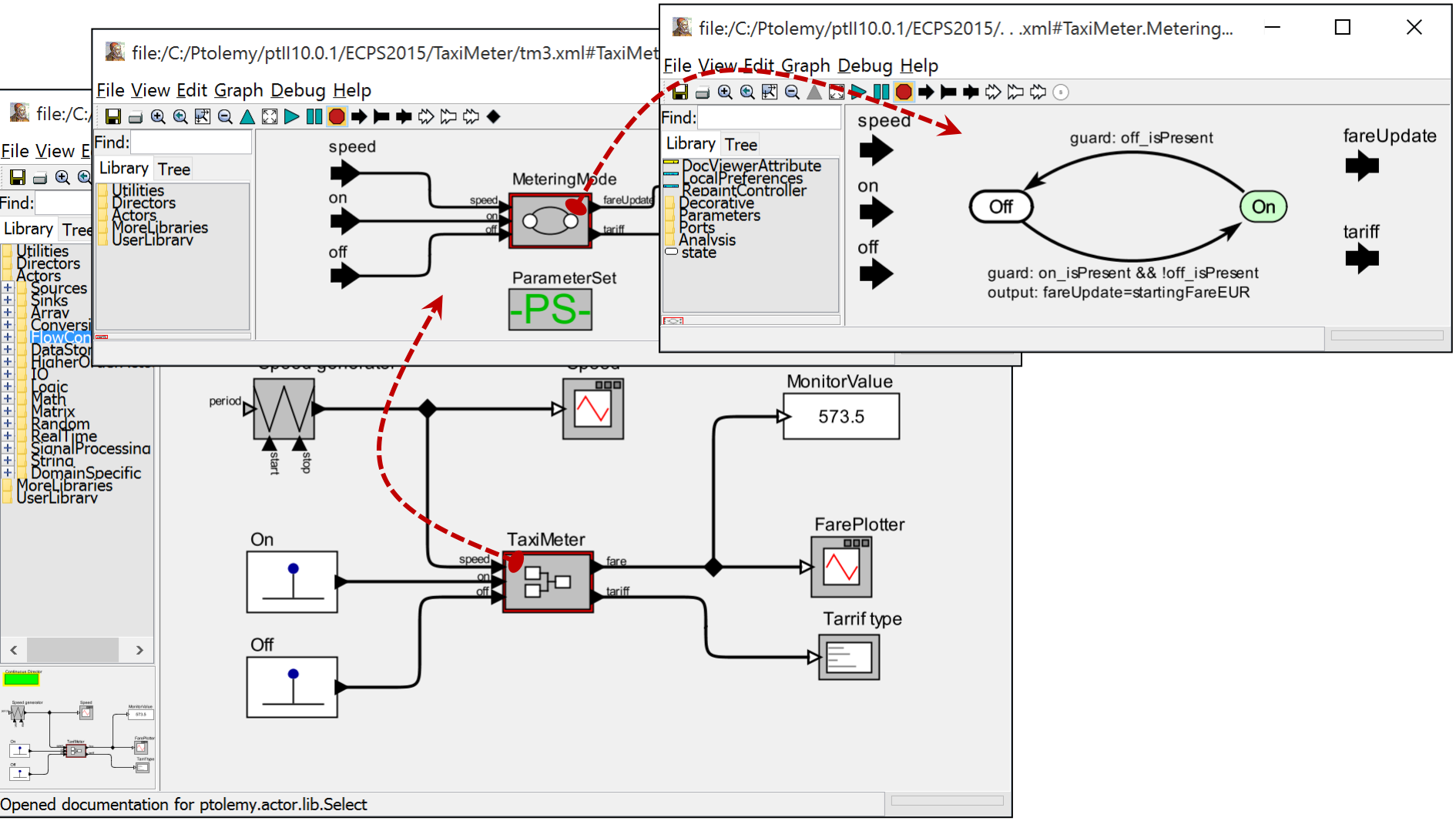
Requirements

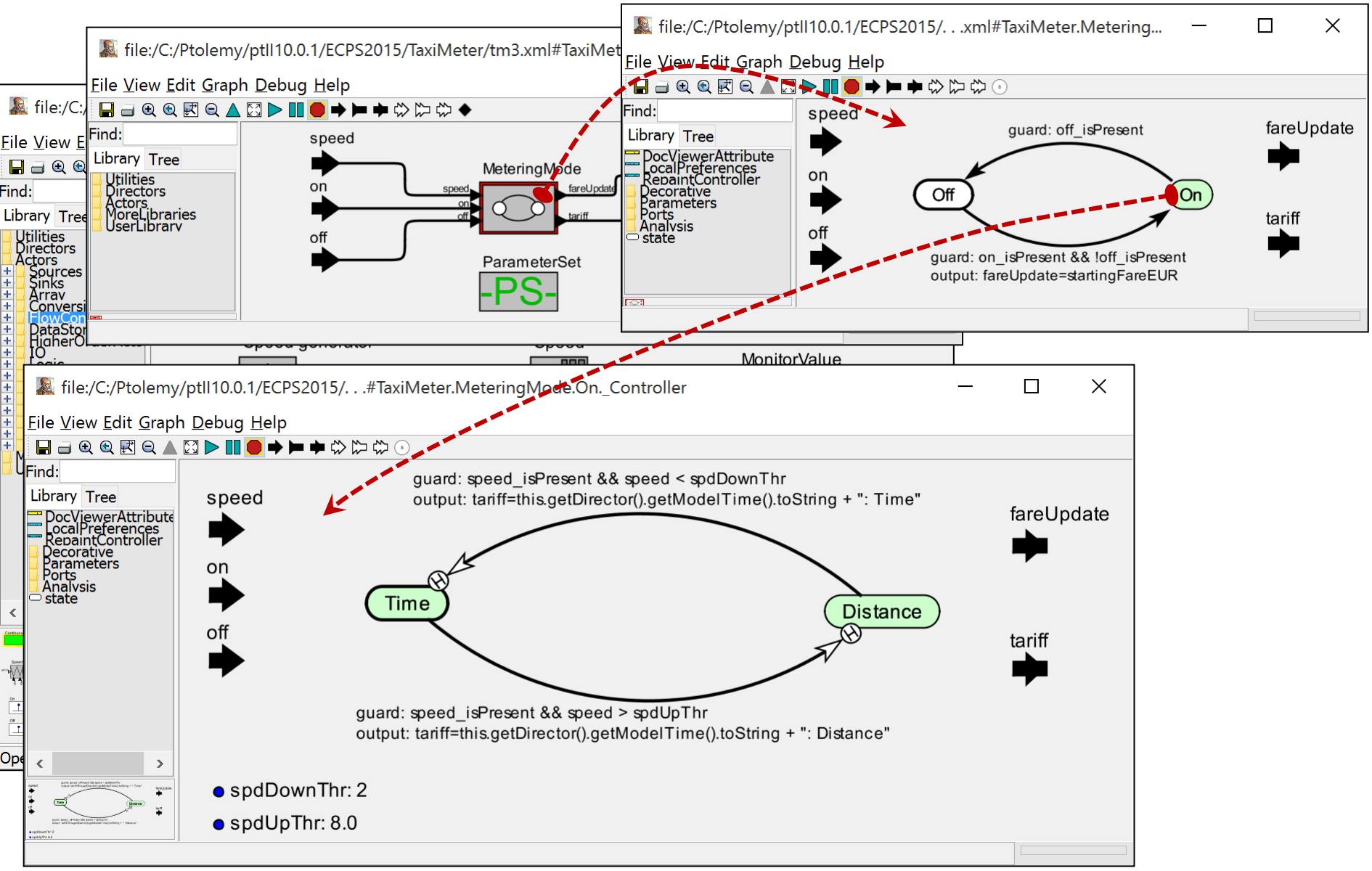
- A parking place in the garage may be assigned to at most one car at a time
- A car is denied entry in the garage only if the garage is full. In this case, the garage provides an estimate of the next time when a place may become free, if this information is available.
- No car may stay indefinitely often on driveways (due to collision conflicts with other cars)

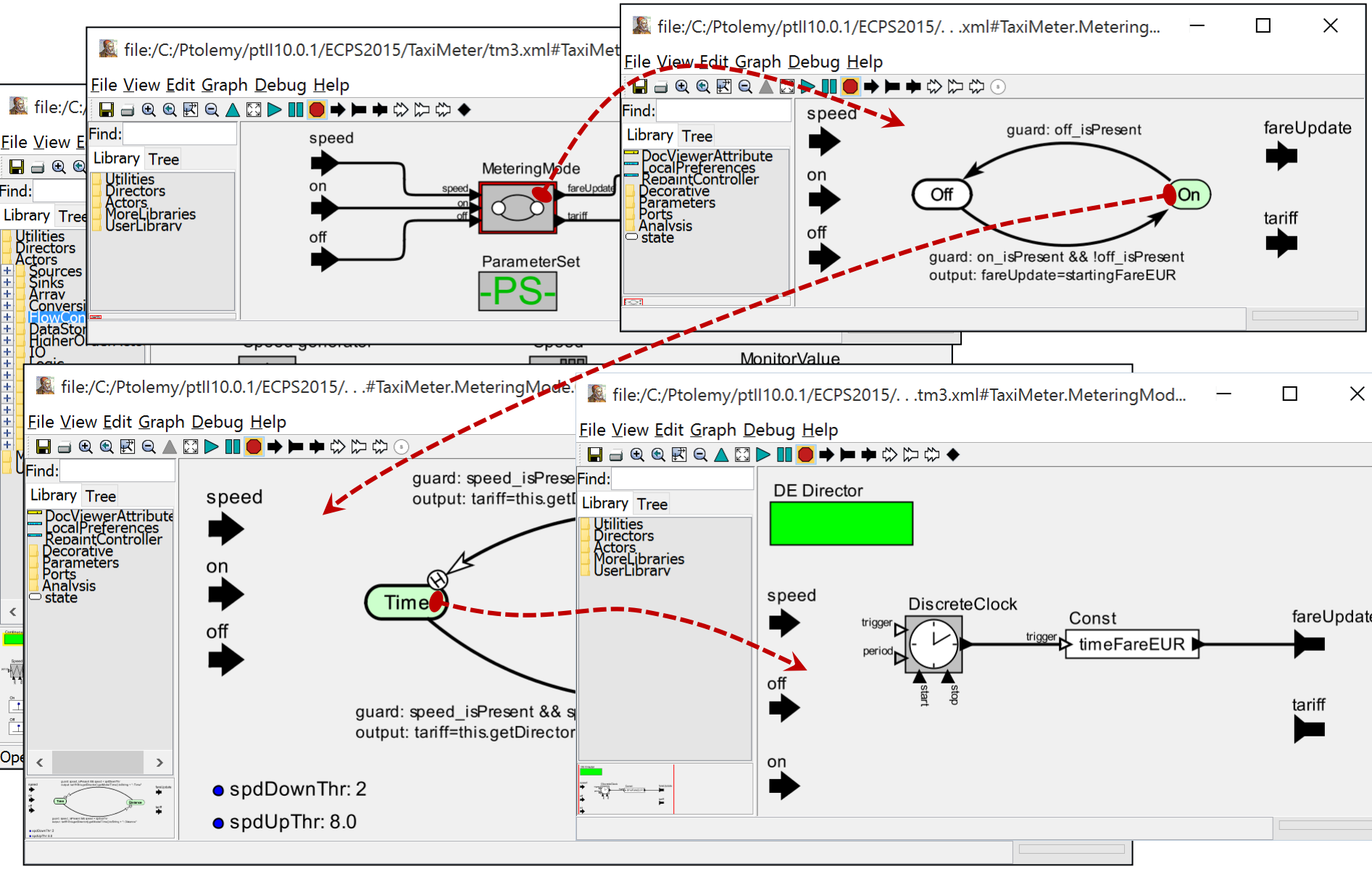
Ptolemy model example (I)




file:/C:/Ptolemy/ptII10.0.1/ECPS2015/TaxiMeter/tm3.xml#TaxiMeter
File View Edit Graph Debug Help
Find:
Library Tree
Utilities
Directors
Actors
MoreLibraries
UserLibrary
speed
on
off
MeteringMode
speed
on
off
fareUpdate
tariff
Accumulator
fare
reset
tariff
ParameterSet
-PS-
period
start
stop
MonitorValue
573.5
On
TaxiMeter
speed
on
off
fare
tariff
FarePlotter
Off
Tarrif type
Opened documentation for ptolemy.actor.lib.Select


file:/C:/Ptolemy/ptII10.0.1/ECPS2015/. . .xml#TaxiMeter.Metering...
File View Edit Graph Debug Help
Find:
Library Tree
DocViewerAttribute
LocalPreferences
RepaintController
Decorative
Parameters
Ports
Analysis
state
speed
on
off
guard: off_isPresent
Off
On
guard: on_isPresent && !off_isPresent
output: fareUpdate=startingFareEUR
fareUpdate
tariff
file:/C:/Ptolemy/ptII10.0.1/ECPS2015/TaxiMeter/tm3.xml#TaxiMet
File View Edit Graph Debug Help
Find:
Library Tree
Utilities
Directors
Actors
MoreLibraries
UserLibrary
speed
on
off
MeteringMode
speed
on
off
fareUpdate
tariff
ParameterSet
-PS-
period
start
stop
MonitorValue
573.5
On
TaxiMeter
speed
on
off
fare
tariff
FarePlotter
Off
Tarrif type
Opened documentation for ptolemy.actor.lib.Select


file:/C:/Ptolemy/ptII10.0.1/ECPS2015/TaxiMeter/tm3.xml#TaxiMet
File View Edit Graph Debug Help
Find:
Library Tree
Utilities
Directors
Actors
MoreLibraries
UserLibrary
speed
on
off
MeteringMode
speed
on
off
fareUpdate
tariff
ParameterSet
-PS-
file:/C:/Ptolemy/ptII10.0.1/ECPS2015/. . .xml#TaxiMeter.Metering...
File View Edit Graph Debug Help
Find:
Library Tree
DocViewerAttribute
LocalPreferences
RepaintController
Decorative
Parameters
Ports
Analysis
state
speed
on
off
guard: off_isPresent
Off
On
guard: on_isPresent && !off_isPresent
output: fareUpdate=startingFareEUR
fareUpdate
tariff
MonitorValue
file:/C:/Ptolemy/ptII10.0.1/ECPS2015/. . .#TaxiMeter.MeteringMode.On._Controller
File View Edit Graph Debug Help
Find:
Library Tree
DocViewerAttribute
LocalPreferences
RepaintController
Decorative
Parameters
Ports
Analysis
state
speed
on
off
guard: speed_isPresent && speed < spdDownThr
output: tariff=this.getDirector().getModelTime().toString + ": Time"
Time
Distance
guard: speed_isPresent && speed > spdUpThr
output: tariff=this.getDirector().getModelTime().toString + ": Distance"
fareUpdate
tariff
spdDownThr: 2
spdUpThr: 8.0


file:/C:/Ptolemy/ptII10.0.1/ECPS2015/TaxiMeter/tm3.xml#TaxiMet
File View Edit Graph Debug Help
Find:
Library Tree
Utilities
Directors
Actors
MoreLibraries
UserLibrary
speed
on
off
MeteringMode
speed
on
off
fareUpdate
tariff
ParameterSet
-PS-
file:/C:/Ptolemy/ptII10.0.1/ECPS2015/. . .xml#TaxiMeter.Metering...
File View Edit Graph Debug Help
Find:
Library Tree
DocViewerAttribute
LocalPreferences
RepaintController
Decorative
Parameters
Ports
Analysis
state
speed
on
off
guard: off_isPresent
Off
On
guard: on_isPresent && !off_isPresent
output: fareUpdate=startingFareEUR
fareUpdate
tariff
MonitorValue
file:/C:/Ptolemy/ptII10.0.1/ECPS2015/. . .#TaxiMeter.MeteringMode.
File View Edit Graph Debug Help
Find:
Library Tree
DocViewerAttribute
LocalPreferences
RepaintController
Decorative
Parameters
Ports
Analysis
state
speed
on
off
guard: speed_isPrese
output: tariff=this.getD
Time
guard: speed_isPresent && s
output: tariff=this.getDirector
spdDownThr: 2
spdUpThr: 8.0
file:/C:/Ptolemy/ptII10.0.1/ECPS2015/. . .tm3.xml#TaxiMeter.MeteringMod...
File View Edit Graph Debug Help
Find:
Library Tree
Utilities
Directors
Actors
MoreLibraries
UserLibrary
DE Director
speed
off
on
DiscreteClock
trigger
period
start
stop
Const
trigger
timeFareEUR
fareUpdate
tariff

Steps for Ptolemy-based projects

1. Download and install Ptolemy II
2. Read about actor-based design and DE domain from the Ptolemy book.
3. Run DE domain demos
4. Design your system model
5. Design custom Ptolemy actors as needed
6. Simulate!

Next steps

- Download the slides and look through the projects/links
- Pair up (teams)
- Send email message with team and 3 projects in order of preference (1 is most preferred)
- Receive confirmation
- Next week: introductions for the chosen projects