The TDL Experimental Domain in Ptolemy

Stefan Resmerita
Patricia Derler
Wolfgang Pree

C. Doppler Lab Embedded Software Systems
cs.uni-salzburg.at

UNIVERSITÄT SALZBURG

Contents

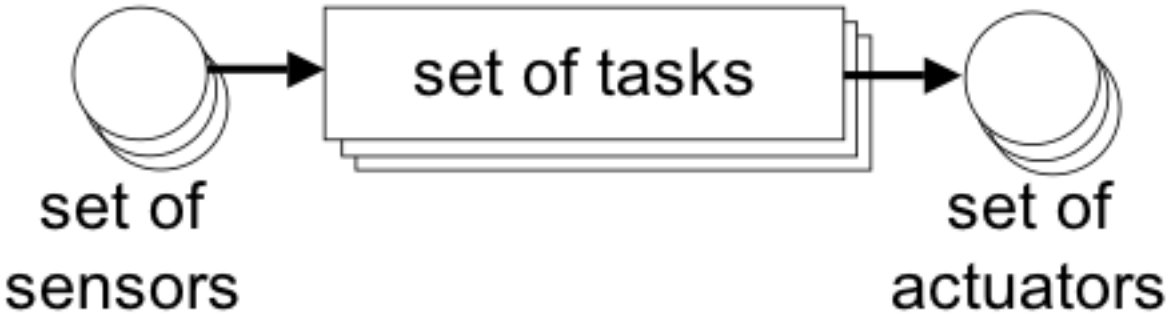
- Timing Definition Language (TDL) in a nutshell
- Implementation of the TDL Ptolemy domain
- Conclusions and further work

UNIVERSITÄT SALZBURG

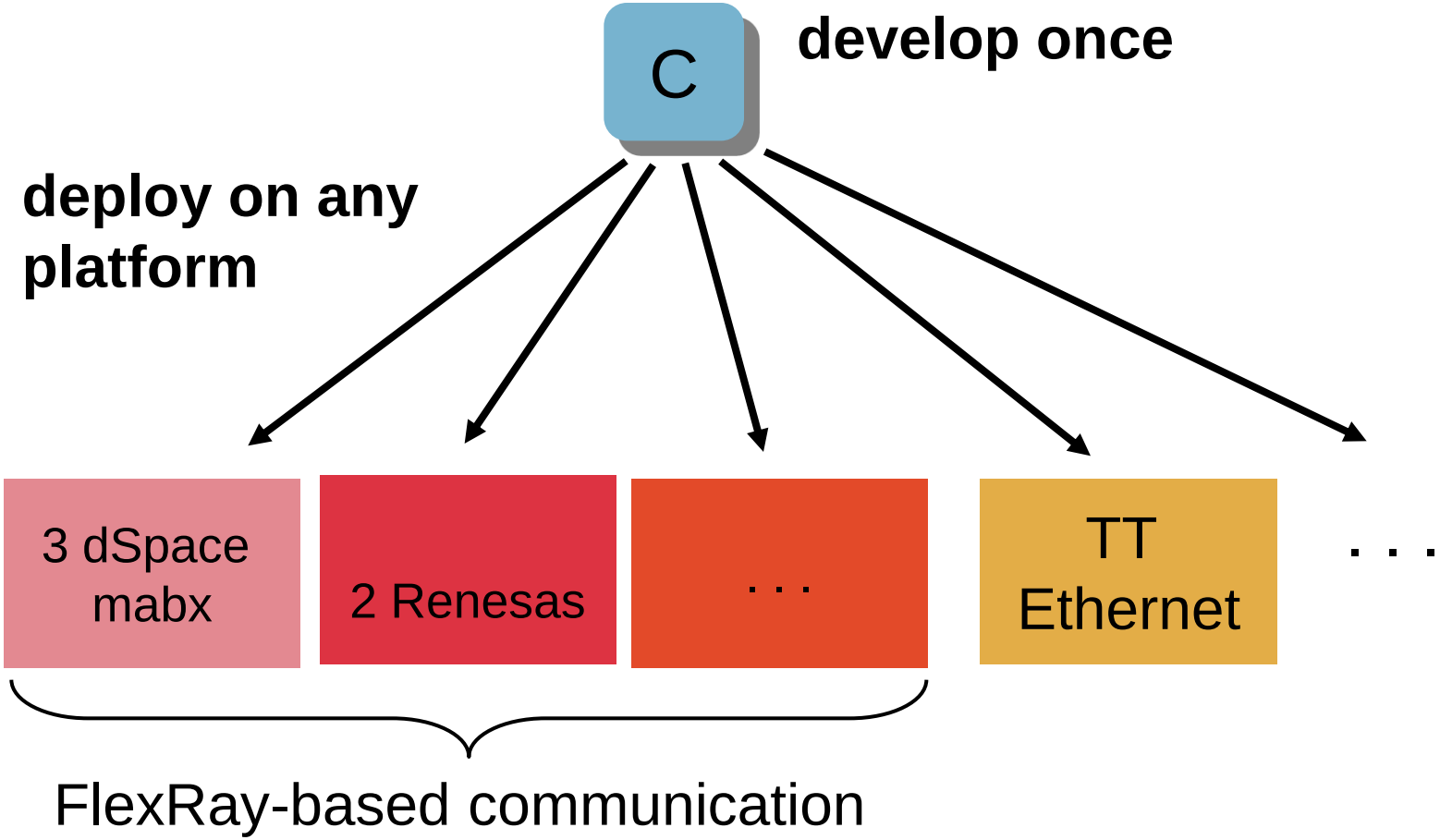
TDL in a nutshell

What is TDL?

- A high-level textual notation for defining the timing behavior of a real-time application.
- Conceptually based on Giotto, in particular its Logical Execution Time (LET) abstraction.
- **TDL = Giotto + syntax + cleanups**
  + component architecture
  + control engineering enhancements.



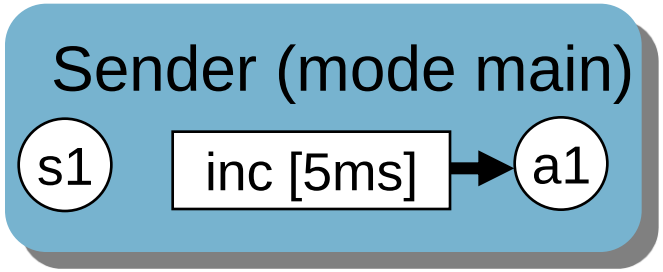
Giotto/TDL vision:

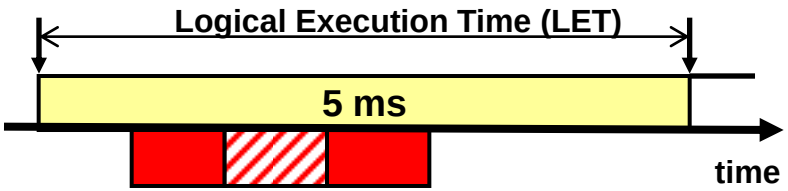


UNIVERSITÄT SALZBURG

TDL syntax by example

```
module Sender {

  sensor boolean s1 uses getS1;
  actuator int a1 uses setA1;

  public task inc {
    output int o := 10;
    uses incImpl(o);
  }

  start mode main [period=5ms] {
    task
      [freq=1] inc(); // LET=5ms/1=5ms
    actuator
      [freq=1] a1 := inc.o;
    mode
      [freq=1] if exitMain(s1) then freeze;
  }

  mode freeze [period=1000ms] {}
}
```

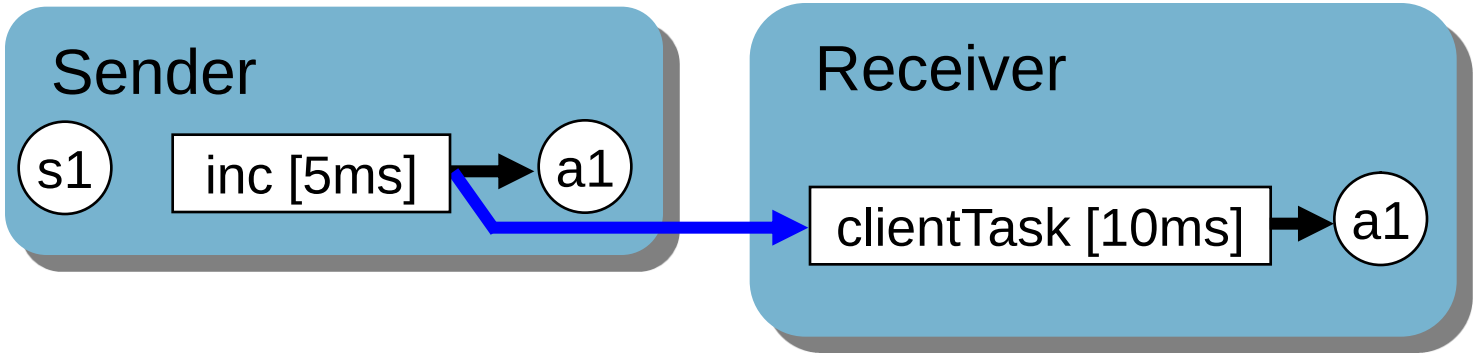




UNIVERSITÄT SALZBURG

Module import

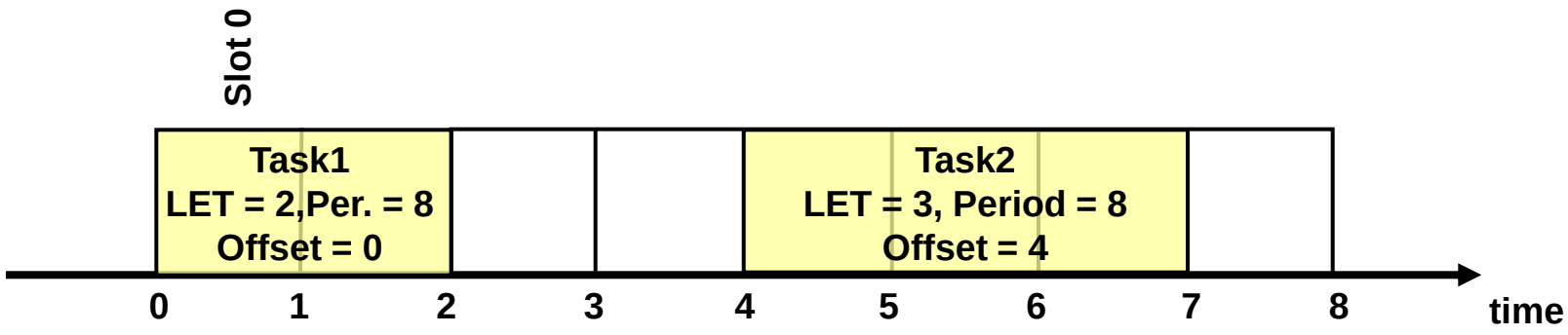
```
module Receiver {

  import Sender;
  …
  task clientTask {
    input int i1;
    …
  }
  mode main [period=10ms] {
    task [freq=1] clientTask(Sender.inc.o); // LET = 10ms / 1 = 10ms
    …
  }
}
```



TDL: Additional Features

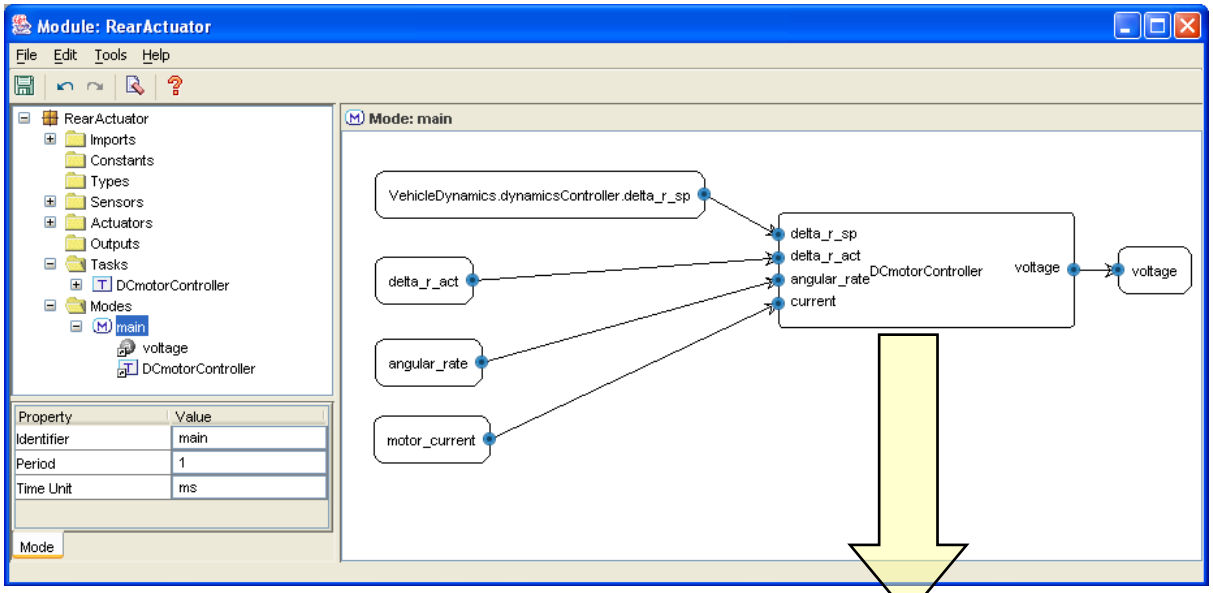
- Fast tasks

- Different LET and period of invocation for a task
  - Flexible placement of the LET within the period
  - Specified by slot selection
  - Example:

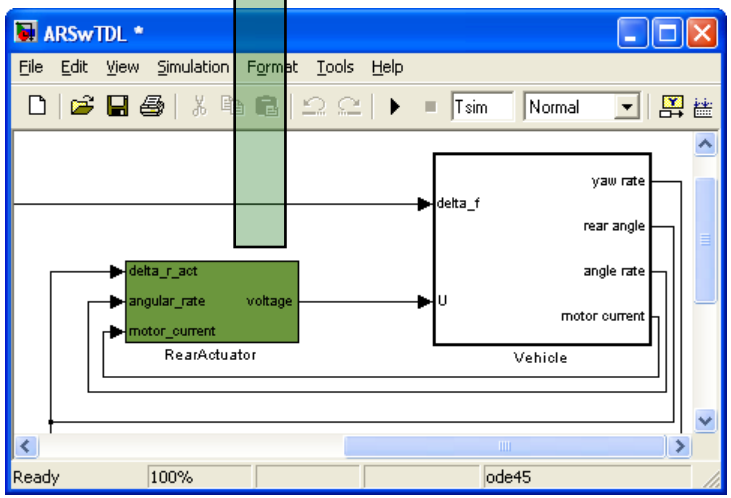


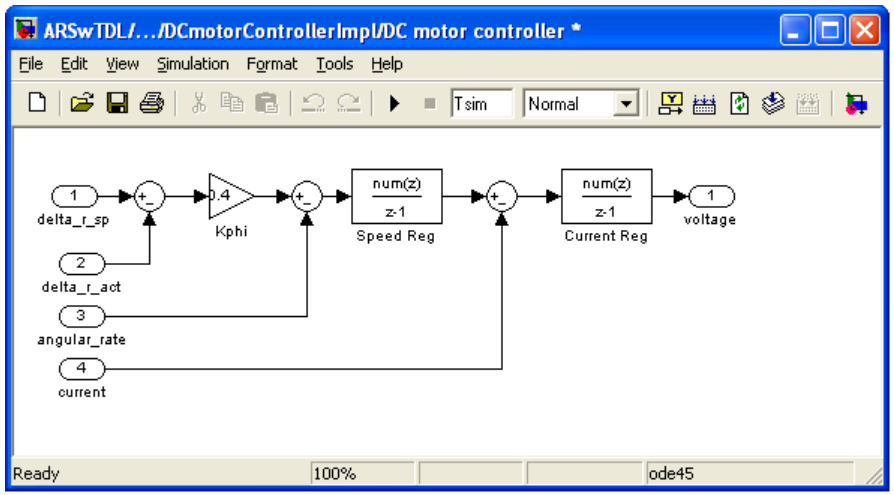
TDL Modeling and Simulation

TDL:VisualCreator



Simulink:





Why Ptolemy?

- Experiment with heterogeneous models involving TDL components
- Rapid testing of new TDL developments
- Existence of computational models closely related to TDL
  - Reuse of functionality
  - Reuse of graphical user interface

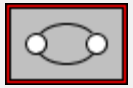
The Experimental TDL Domain in Ptolemy

General

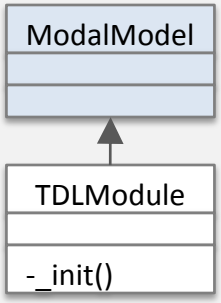
- Based on **modal models** of the **finite state machine domain**
- Reuse of existing concepts
  - Modes with different behaviors
  - Only one active mode
  - Transitions between modes
  - Graphical representation
- Main changes
  - Order of execution
  - Deterministic choice of transitions
  - Mode switches only at certain points during execution
  - Output ports are not updated after every firing
  - Guarded task executions and port updates

TDL Module

TDL Module Actor

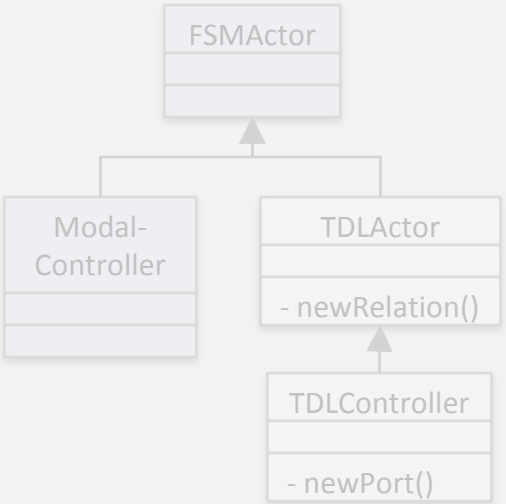


- Contains modes



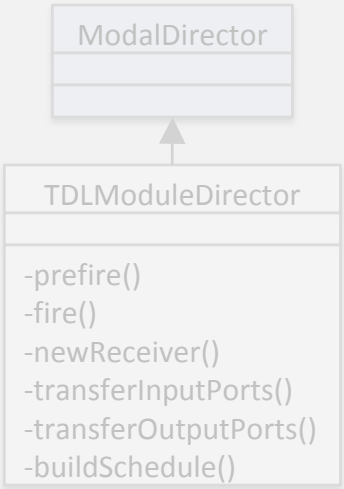
TDL Mode Controller

- Mirrors ports in refinements



TDL Module Director

- Generates and executes schedule

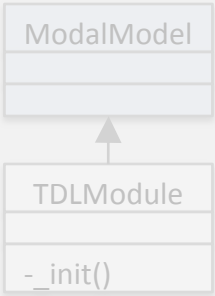


UNIVERSITÄT SALZBURG

TDL Module

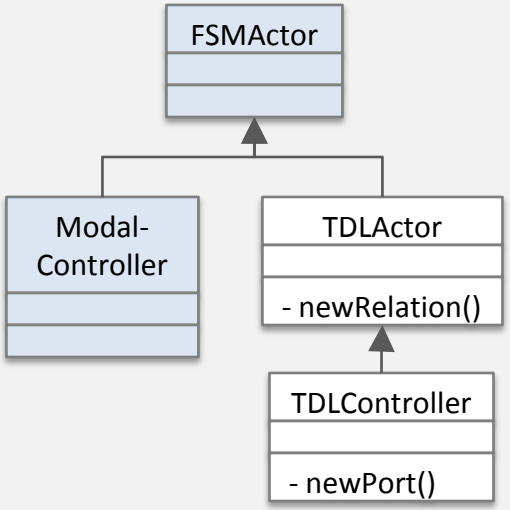
TDL Module Actor

- Contains modes



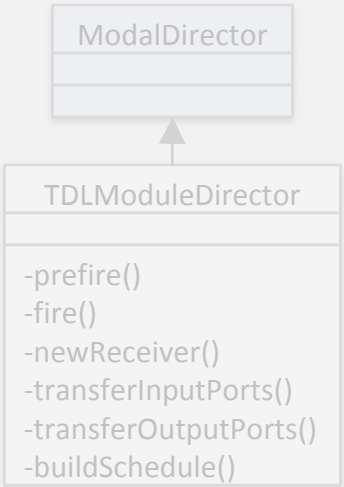
TDL Mode Controller

- Mirrors ports in refinements



TDL Module Director

- Generates and executes schedule

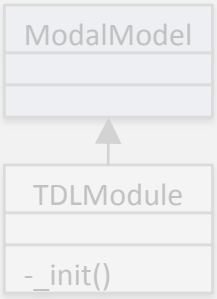


UNIVERSITÄT SALZBURG

TDL Module

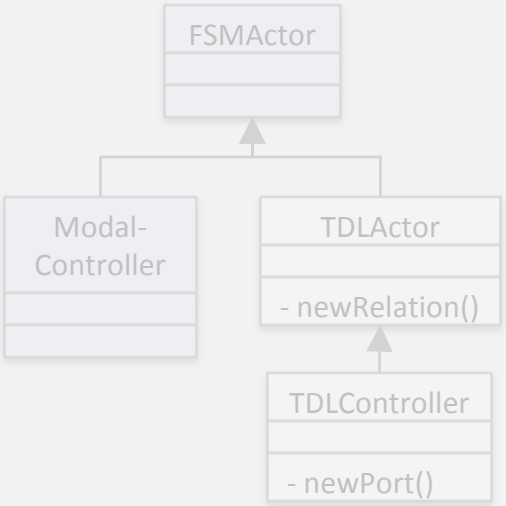
TDL Module Actor

- Contains modes



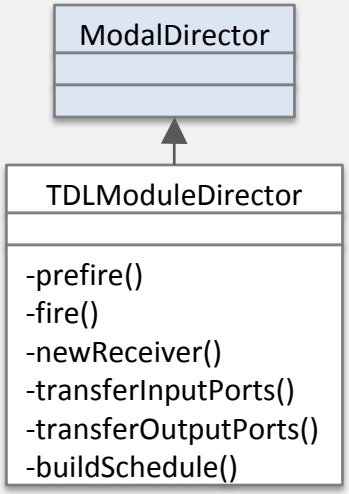
TDL Mode Controller

- Mirrors ports in refinements



TDL Module Director

- Generates and executes schedule



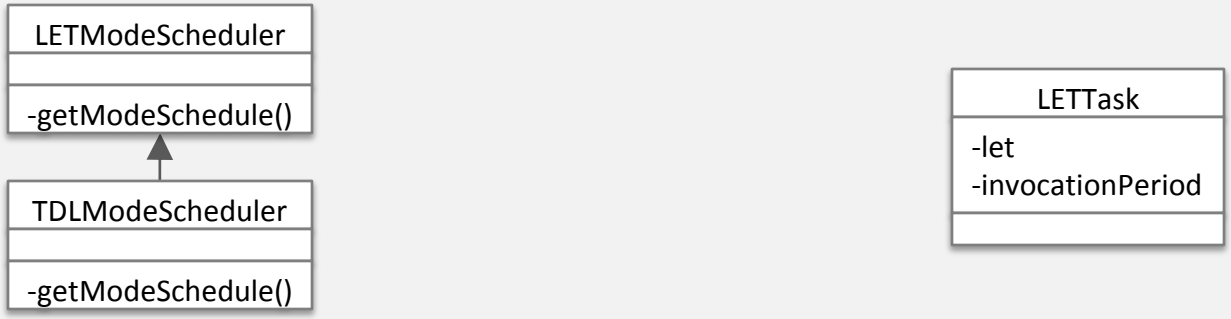
UNIVERSITÄT SALZBURG

TDL Schedule

Mode Schedule Generation

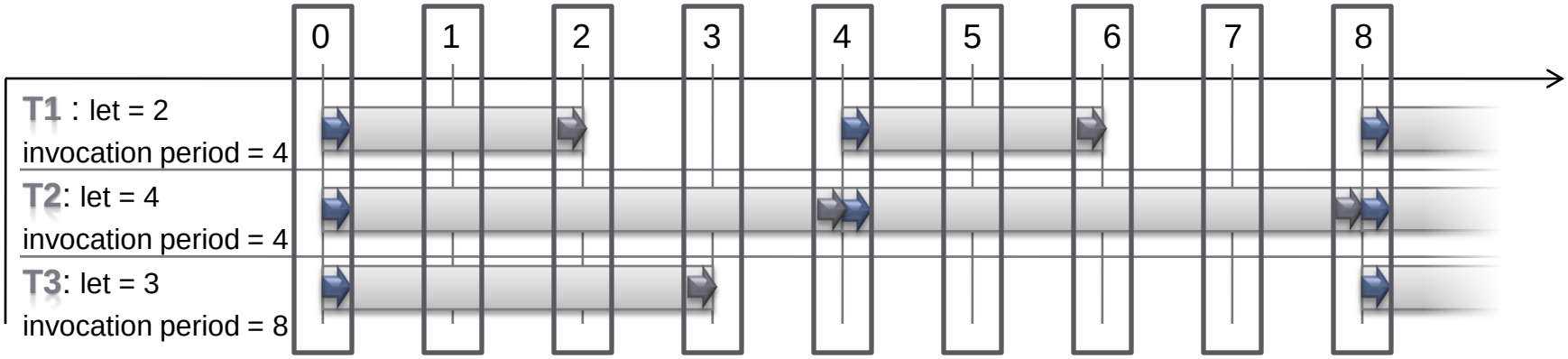
The TDL module director creates a static schedule in 2 steps:

1. LET Schedule: general Schedule for LET based tasks
2. TDL-specific Schedule: actuators, fast tasks, mode switches



UNIVERSITÄT SALZBURG

LET Schedule Example

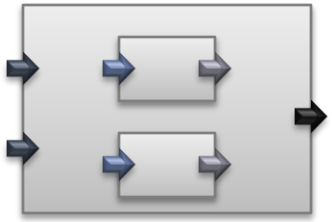


8 is the least common multiple
of all invocation periods

Time	Scheduled actions	Example		
8=8	Update tasks output ports	T1_out	T2_out	T3_out
	Update tasks input ports	T1_in	T2_in	T3_in
	Execute tasks	T1	T2	T3

TDL Schedule

Time	Scheduled actions
0	Update LET tasks output ports
	Update actuators
	Test mode switches
	Fast tasks: - Update input ports - Execute fast tasks - Update output ports - Update connected actuators
	Update LET tasks input ports
	Execute LET tasks
0 + t	Update LET tasks output ports
	…



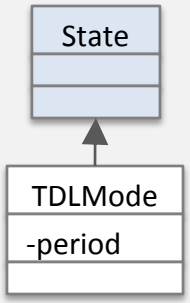
general LET based schedule
TDL specific parts of the schedule

UNIVERSITÄT SALZBURG

TDL Mode

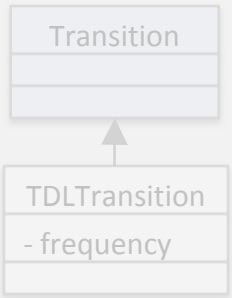
TDL Mode

- Group TDL tasks



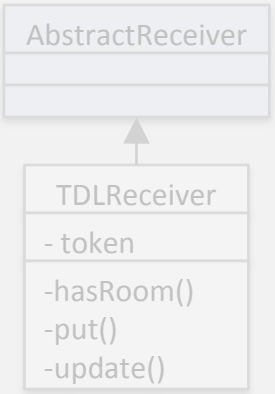
TDL Transition

- Switch between TDL modes



TDL Receiver

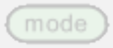
- All receivers inside the TDL module
- Based on Giotto Receiver



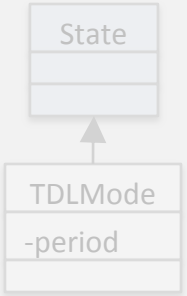
UNIVERSITÄT SALZBURG

TDL Mode

TDL Mode

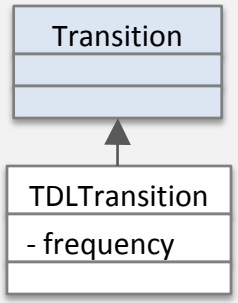


- Group TDL tasks



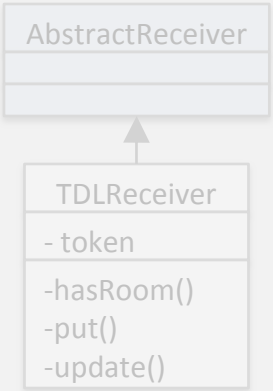
TDL Transition

- Switch between TDL modes



TDL Receiver

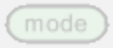
- All receivers inside the TDL module
- Based on Giotto Receiver



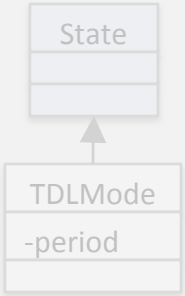
UNIVERSITÄT SALZBURG

TDL Mode

TDL Mode

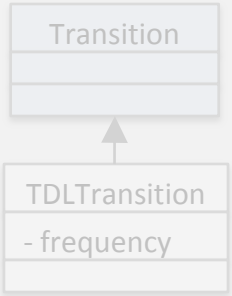


- Group TDL tasks



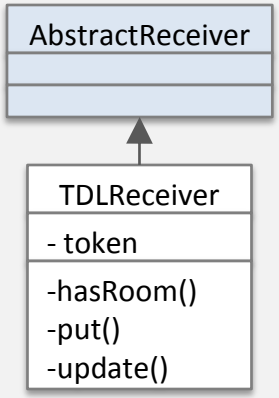
TDL Transition

- Switch between TDL modes



TDL Receiver

- All receivers inside the TDL module
- Based on Giotto Receiver

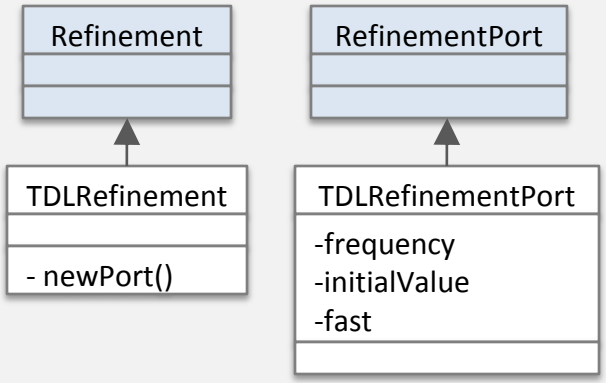


UNIVERSITÄT SALZBURG

TDL Mode

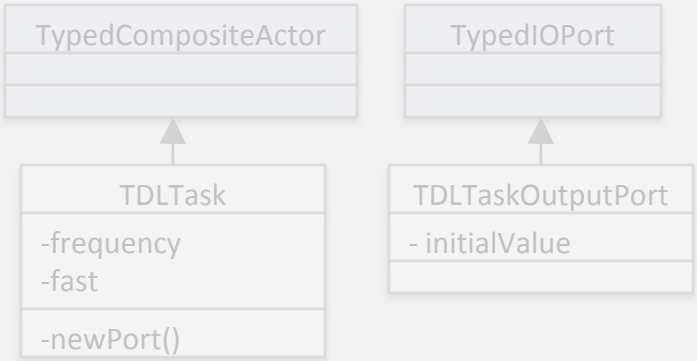
TDL Refinement

- A TDL mode has exactly one TDL refinement



TDL Task Actor

- A TDL refinement contains only TDL tasks
- TDL tasks can only be SDF actors

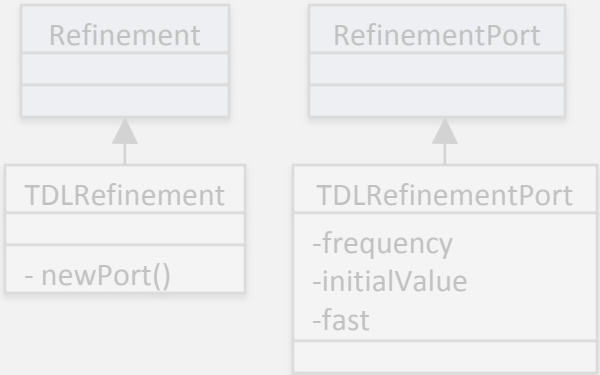


UNIVERSITÄT SALZBURG

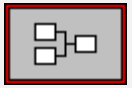
TDL Mode

TDL Refinement

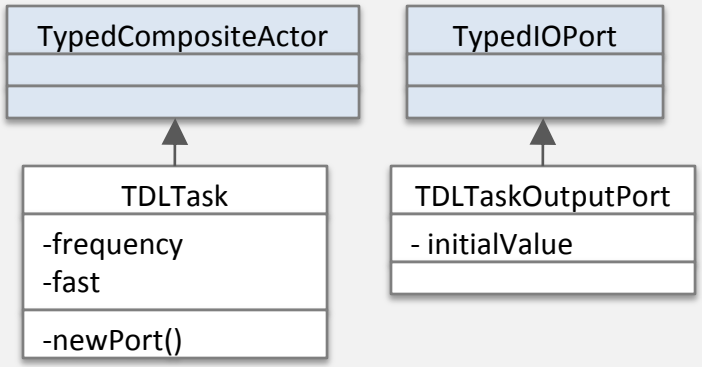
- A TDL mode has exactly one TDL refinement



TDL Task Actor



- A TDL refinement contains only TDL tasks
- TDL tasks can only be SDF actors

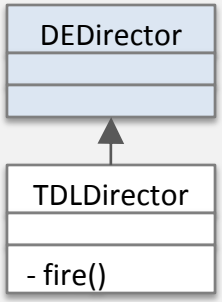


UNIVERSITÄT SALZBURG

TDL Director

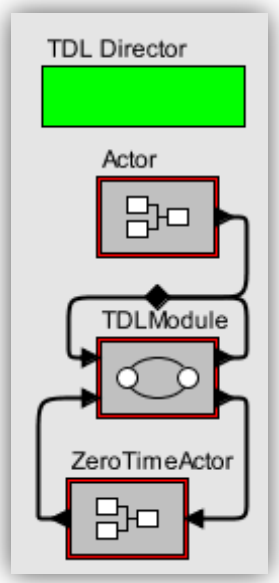
TDL Director

- Must be top-level director for models containing TDL module actors
- Mostly the same as the DE director with special handling of TDL module actors



UNIVERSITÄT SALZBURG

TDL Schedule

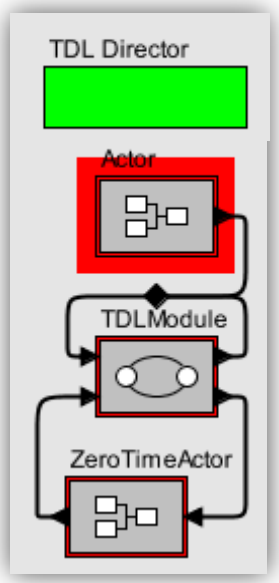


Time	Scheduled actions
0	Update LET tasks output ports
	Update actuators
	Calculate mode switches
	Fast tasks: - Update input ports - Execute fast tasks - Update output ports - Update connected actuators
	Update LET tasks input ports
	Execute LET tasks
0 + t	Update LET tasks output ports
	…

general LET based schedule
TDL specific parts of the schedule

UNIVERSITÄT SALZBURG

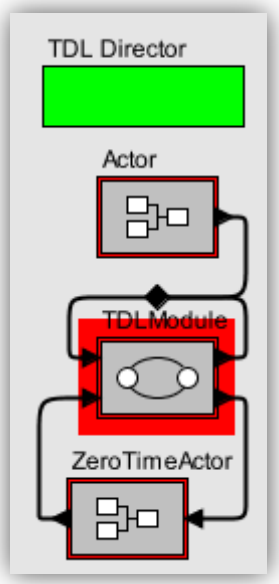
TDL Schedule



Time	Scheduled actions
0	Update LET tasks output ports
	Update actuators
	Calculate mode switches
	Fast tasks: - Update input ports - Execute fast tasks - Update output ports - Update connected actuators
	Update LET tasks input ports
	Execute LET tasks
0 + t	Update LET tasks output ports
	…

UNIVERSITÄT SALZBURG

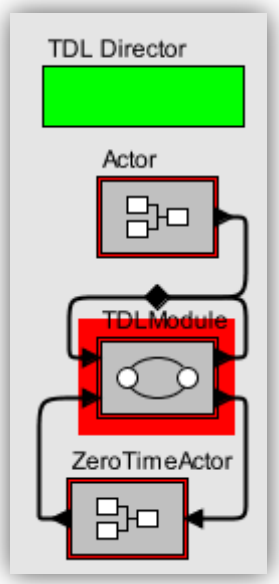
TDL Schedule



Time	Scheduled actions
0	Update LET tasks output ports
	Update actuators
	Calculate mode switches
	Fast tasks: - Update input ports - Execute fast tasks - Update output ports - Update connected actuators
	Update LET tasks input ports
	Execute LET tasks
0 + t	Update LET tasks output ports
	…

UNIVERSITÄT SALZBURG

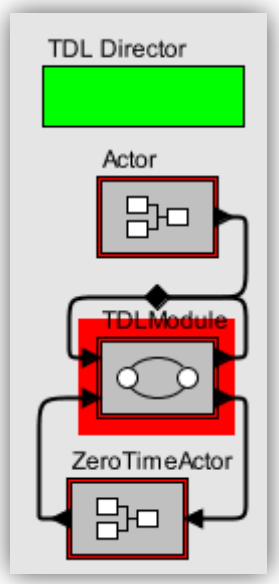
TDL Schedule



Time	Scheduled actions
0	Update LET tasks output ports
	Update actuators
	Calculate mode switches
	Fast tasks: - Update input ports - Execute fast tasks - Update output ports - Update connected actuators
	Update LET tasks input ports
	Execute LET tasks
0 + t	Update LET tasks output ports
	…

UNIVERSITÄT SALZBURG

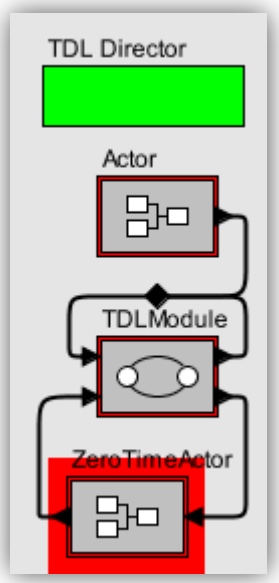
TDL Schedule



Time	Scheduled actions
0	Update LET tasks output ports
	Update actuators
	Calculate mode switches
	Fast tasks: - Update input ports - Execute fast tasks - Update output ports - Update connected actuators
	Update LET tasks input ports
	Execute LET tasks
0 + t	Update LET tasks output ports
	…

UNIVERSITÄT SALZBURG

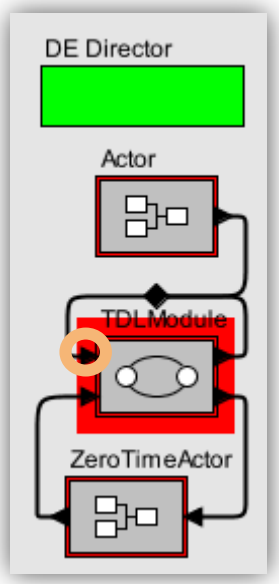
TDL Schedule



Time	Scheduled actions
0	Update LET tasks output ports
	Update actuators
	Calculate mode switches
	Fast tasks: - Update input ports - Execute fast tasks - Update output ports - Update connected actuators
	Update LET tasks input ports
	Execute LET tasks
0 + t	Update LET tasks output ports
	…

UNIVERSITÄT SALZBURG

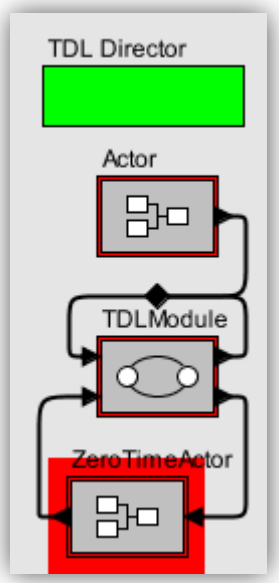
TDL Schedule



Time	Scheduled actions
0	Update LET tasks output ports
	Update actuators
	Calculate mode switches
	Fast tasks: - Update input ports - Execute fast tasks - Update output ports - Update connected actuators
	Update LET tasks input ports
	Execute LET tasks
0 + t	Update LET tasks output ports
	…

UNIVERSITÄT SALZBURG

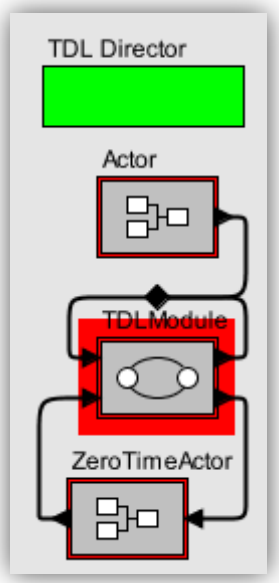
TDL Schedule



Time	Scheduled actions
0	Update LET tasks output ports
	Update actuators
	Calculate mode switches
	Fast tasks: - Update input ports - Execute fast tasks - Update output ports - Update connected actuators
	Update LET tasks input ports
	Execute LET tasks
0 + t	Update LET tasks output ports
	…

UNIVERSITÄT SALZBURG

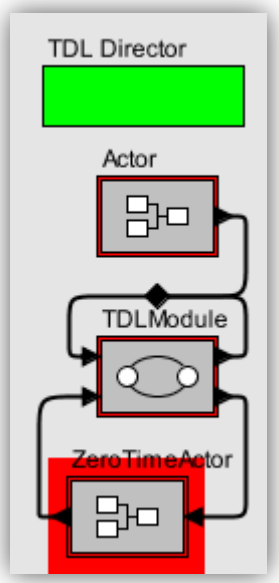
TDL Schedule



Time	Scheduled actions
0	Update LET tasks output ports
	Update actuators
	Calculate mode switches
	Fast tasks: - Update input ports - Execute fast tasks - Update output ports - Update connected actuators
	Update LET tasks input ports
	Execute LET tasks
0 + t	Update LET tasks output ports
	…

UNIVERSITÄT SALZBURG

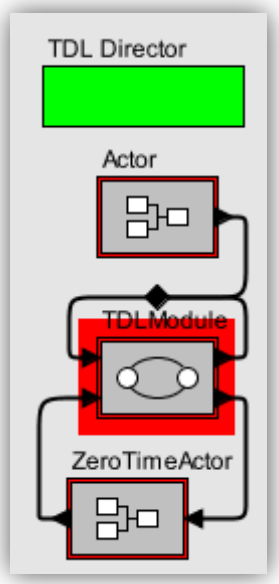
TDL Schedule



Time	Scheduled actions
0	Update LET tasks output ports
	Update actuators
	Calculate mode switches
	Fast tasks: - Update input ports - Execute fast tasks - Update output ports - Update connected actuators
	Update LET tasks input ports
	Execute LET tasks
0 + t	Update LET tasks output ports
	…

UNIVERSITÄT SALZBURG

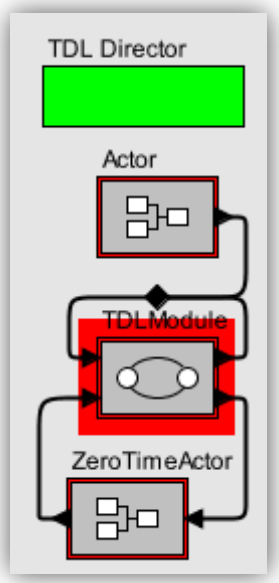
TDL Schedule



Time	Scheduled actions
0	Update LET tasks output ports
	Update actuators
	Calculate mode switches
	Fast tasks: - Update input ports - Execute fast tasks - Update output ports - Update connected actuators
	Update LET tasks input ports
	Execute LET tasks
0 + t	Update LET tasks output ports
	…

UNIVERSITÄT SALZBURG

TDL Schedule



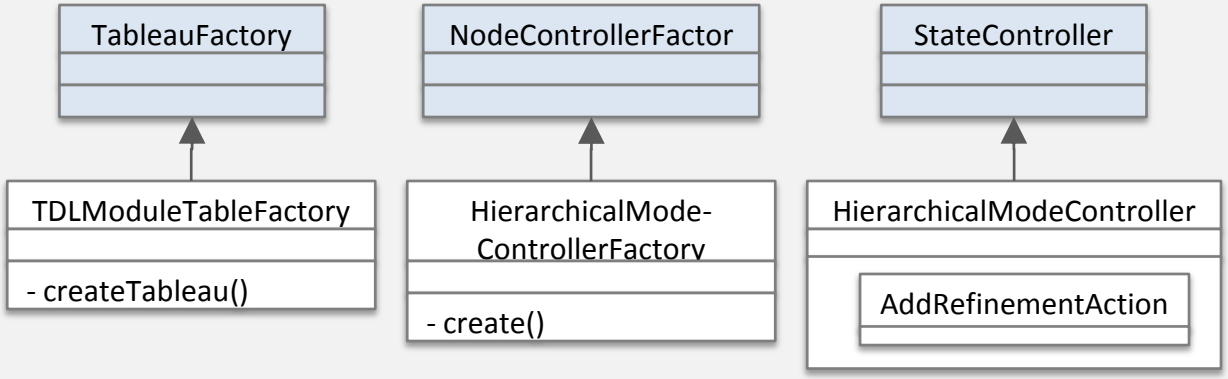
Time	Scheduled actions
0	Update LET tasks output ports
	Update actuators
	Calculate mode switches
	Fast tasks: - Update input ports - Execute fast tasks - Update output ports - Update connected actuators
	Update LET tasks input ports
	Execute LET tasks
0 + t	Update LET tasks output ports
	…

UNIVERSITÄT SALZBURG

User Interface Changes

Vergil

- Graphical representation of TDL modules
- Enable adding TDL refinements to TDL modes



Configuration Files

- Adding TDL actors to Libraries
- Icon for TDL module

UNIVERSITÄT SALZBURG

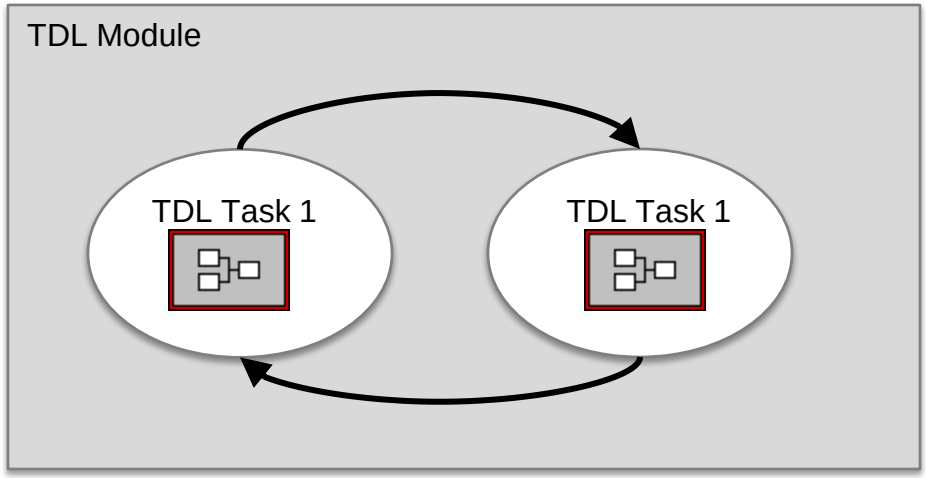
Difficulties Extending Ptolemy

- Many configuration files in different places

- Extending Ptolemy by subclassing
  - Private and protected variables/methods in Ptolemy make deriving difficult
  - Lots of code duplication in case of deriving
    - 1749 LOC for the `TDLDirector` to override the fire of the `DEDirector`
  - Hierarchy
    - E.g. a `TDLTransition` is derived from a `Transition`, however they should be at the same level

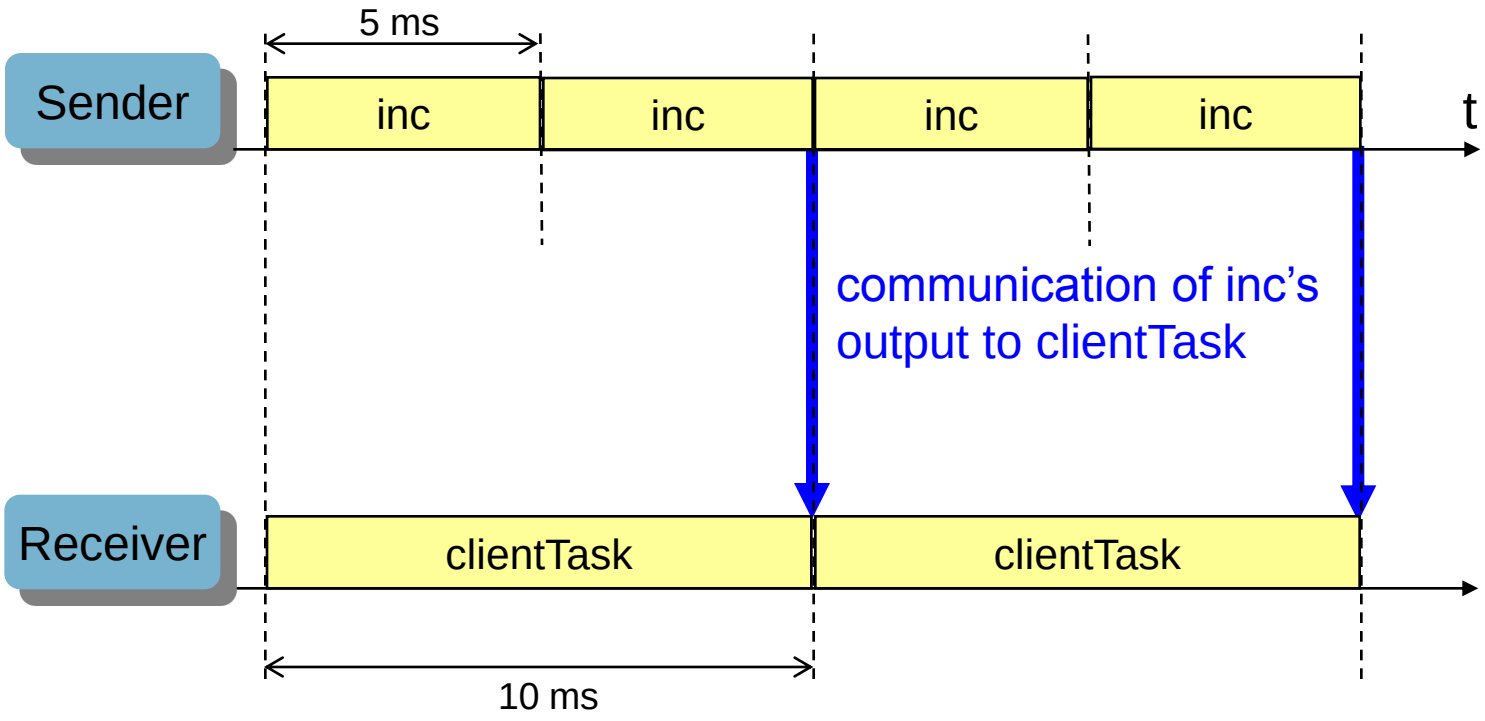
TDL in Ptolemy – Open Issues

- Having the same instance of an actor (task) in different refinements is hard to achieve

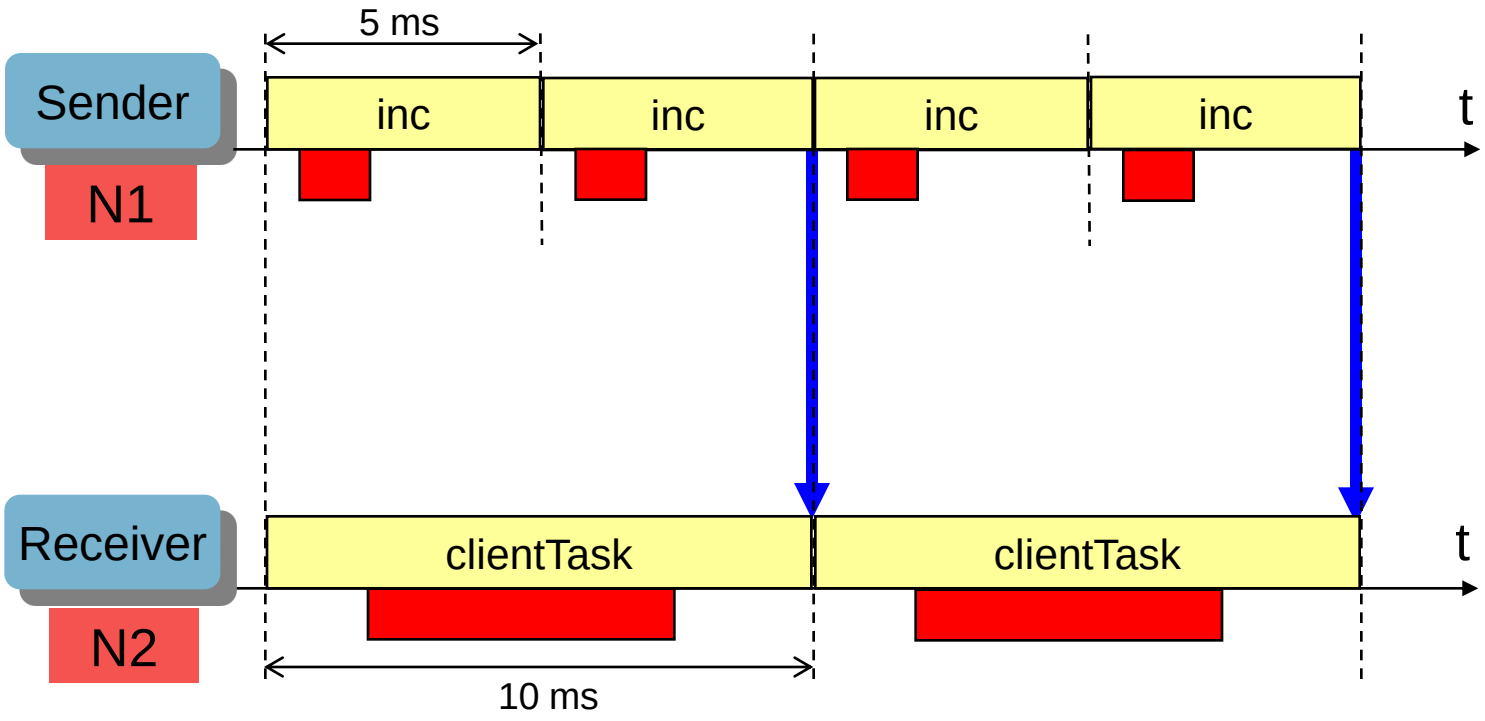


UNIVERSITÄT SALZBURG

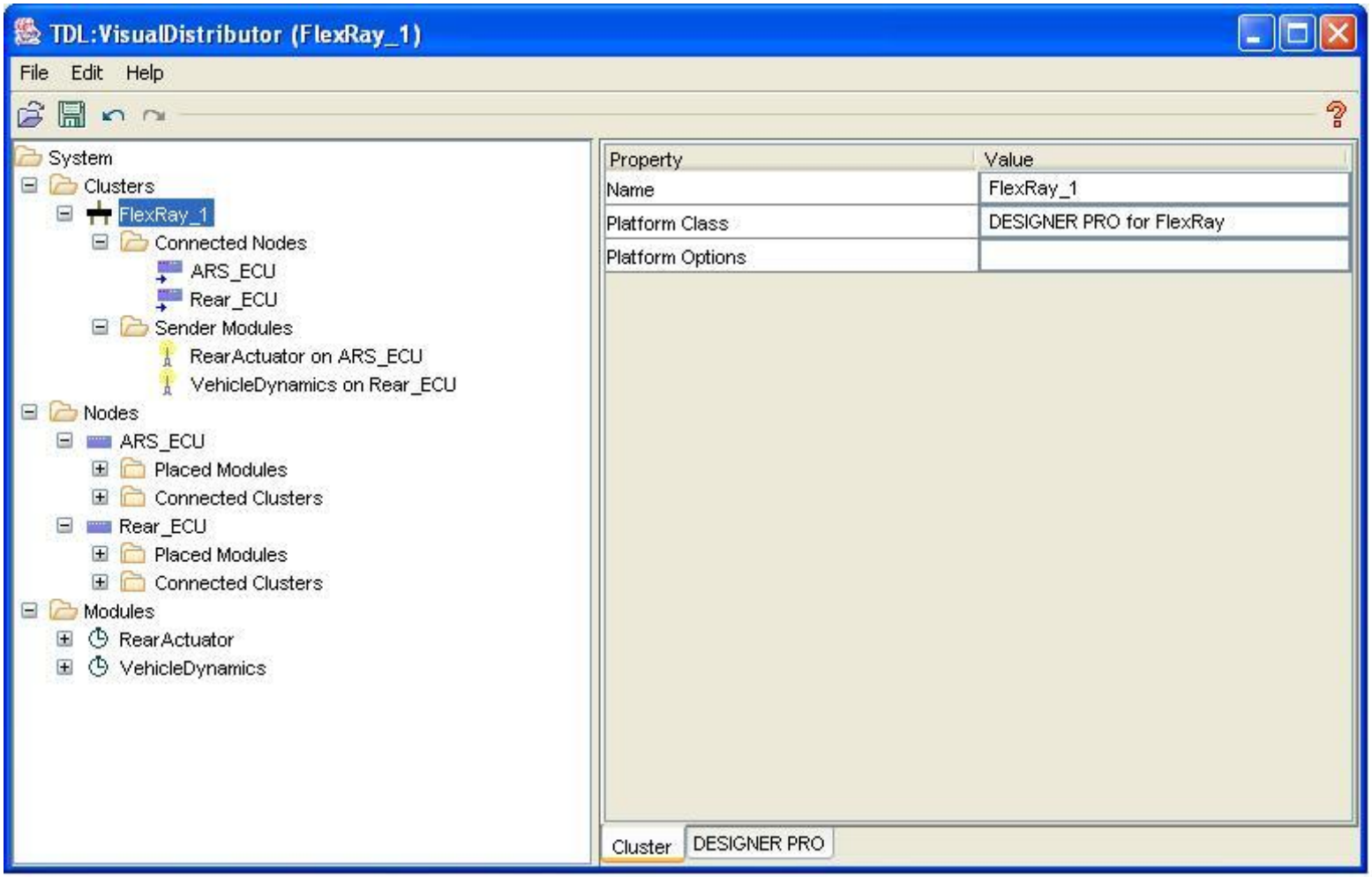
LET behavior



Execution on distributed systems



Deployment of TDL Code



UNIVERSITÄT SALZBURG

Summary

- Experimental TDL domain
  - LET Task
  - TDL Module
  - TDL Domain Controller
- Further work

  Functionality
  - TDL extensions
  - Code generation

  Usability
  - Hide library elements that should not be used
  - Provide all necessary parameters

  Research
  - Interaction with other domains

UNIVERSITÄT SALZBURG