Component-Based Design of Embedded Control Systems

Edward A. Lee & Jie Liu

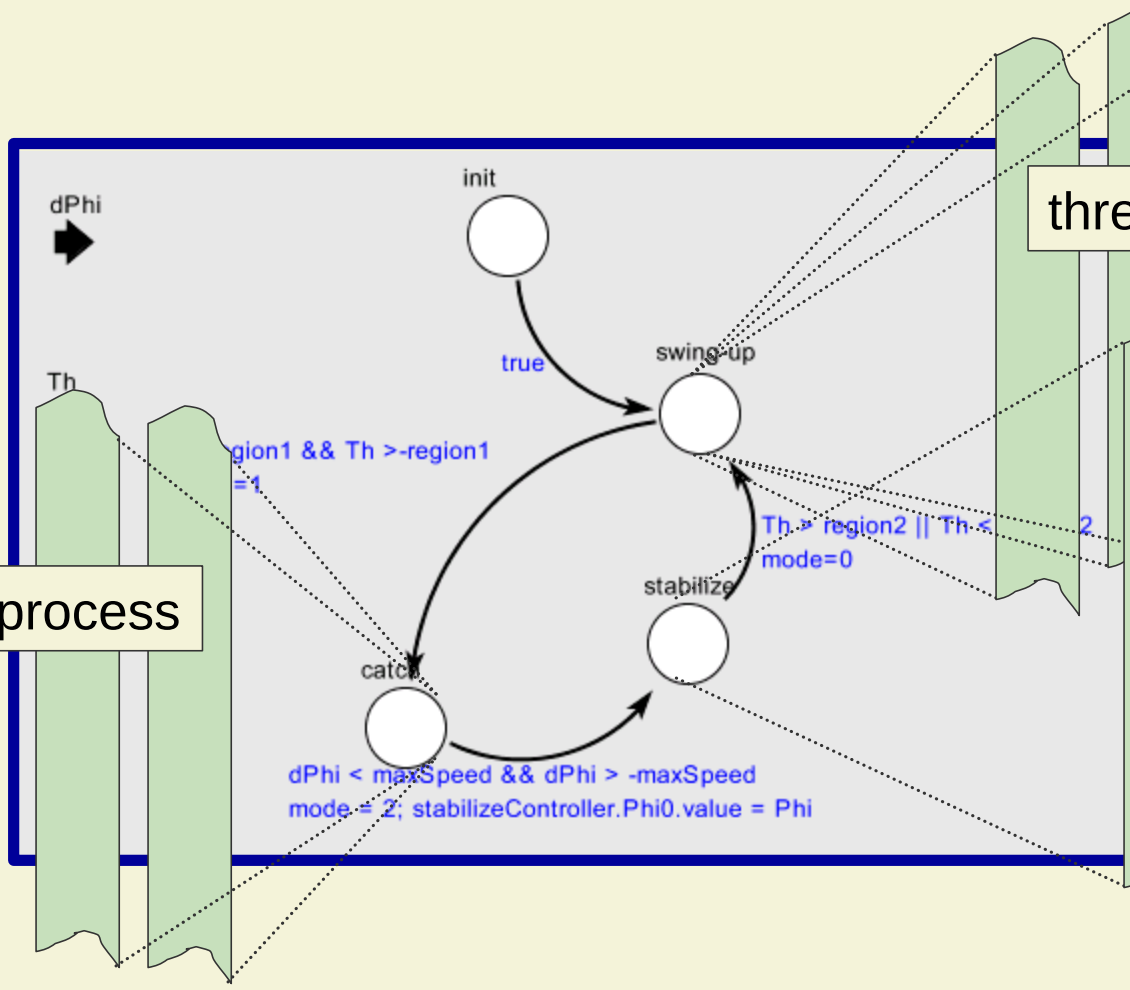
UC Berkeley

with thanks to the entire Berkeley and Boeing SEC teams

SEC PI Meeting

Annapolis, May 8-9, 2001

Precise Mode Change Problem



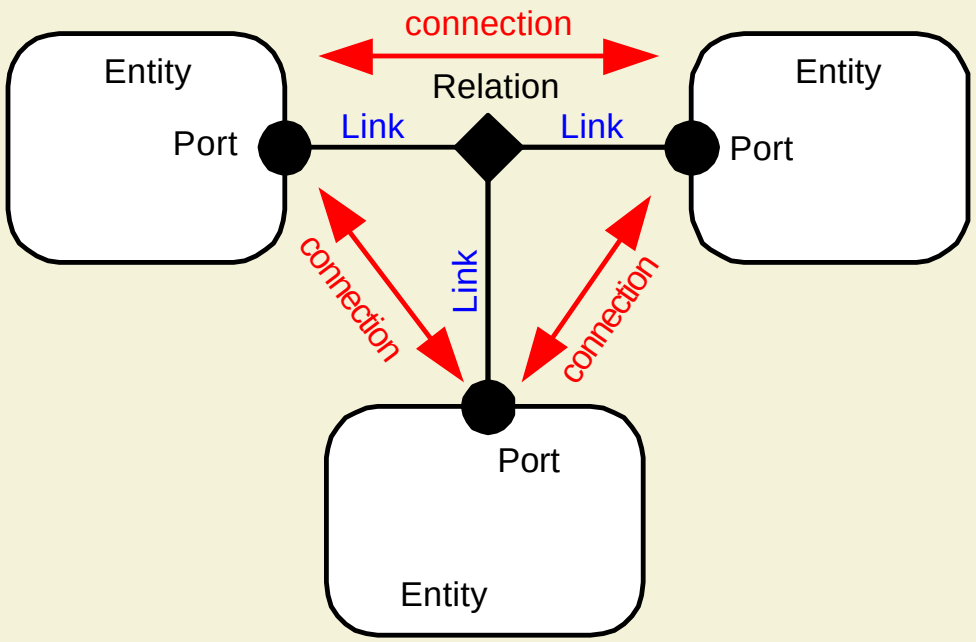
thread or process

thread or process

thread or process

How do you get the processes to a quiescent state to take a mode change?

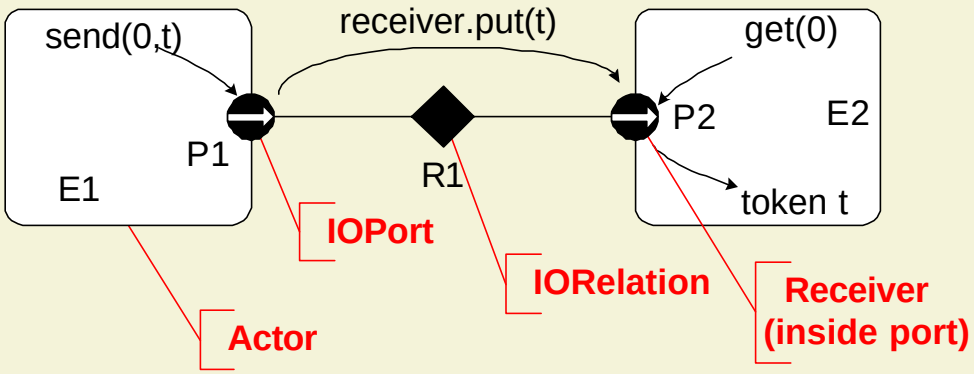
Components and their Relationships



An *abstract syntax* - clustered graphs - is well suited to a wide variety of component-based modeling strategies, ranging from state machines to process networks.

Actor View of Producer/Consumer Components

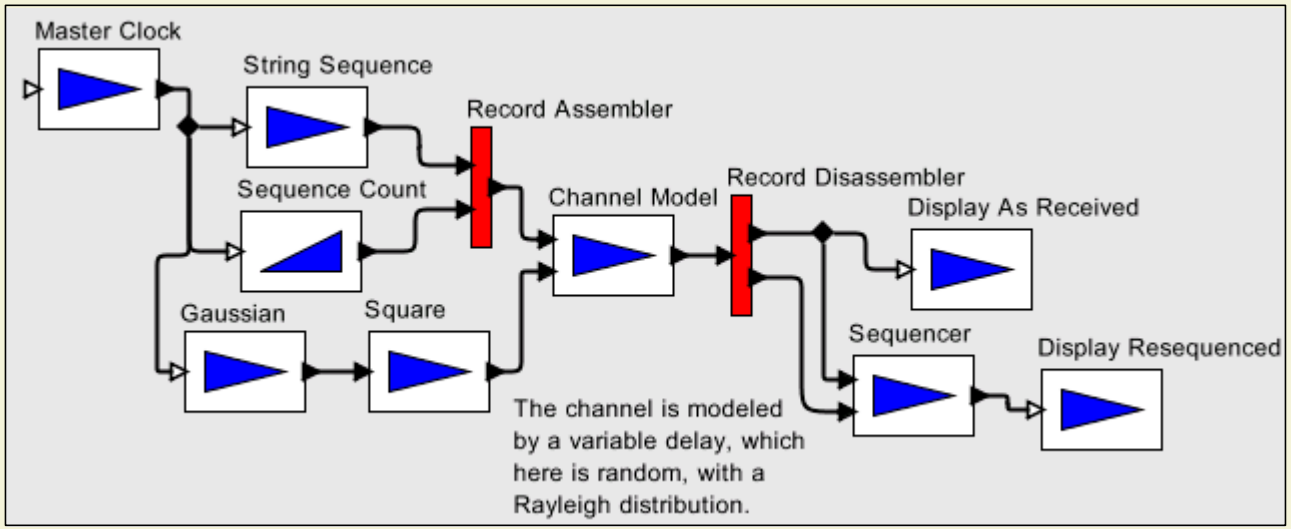
Basic Transport:



Producer/consumer styles:

- continuous-time
- dataflow
- discrete events
- synchronous
- time-driven
- publish/subscribe
- …

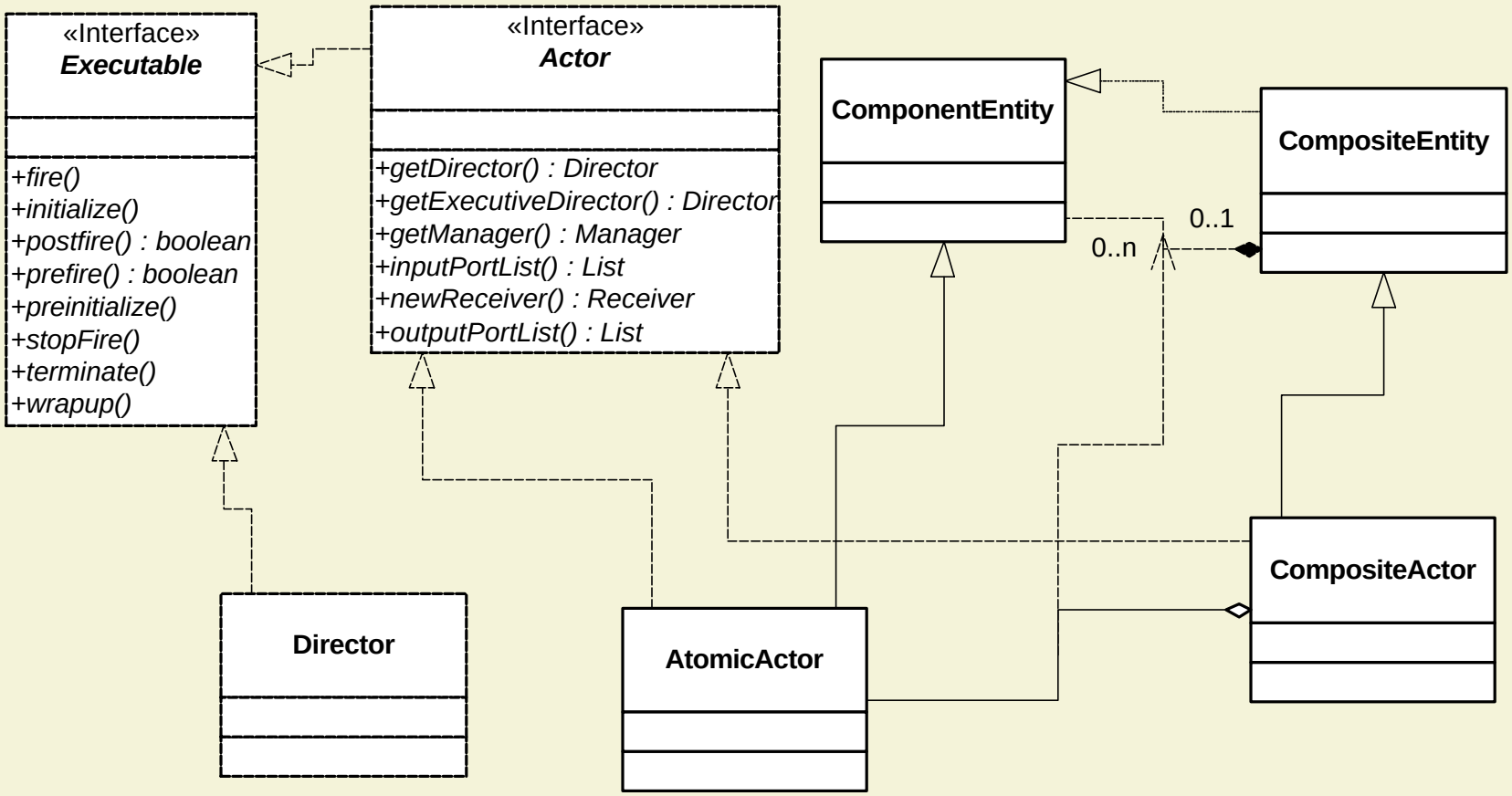
A Laboratory for Exploring Models of Computation



Ptolemy II – Java based, network integrated

- A realization of a model of computation is called a “domain.” Multiple domains can be mixed hierarchically in the same model.

Basic Object Model for Executable Components



Abstract Semantics – How Components Interact

flow of control

- Initialization
- Execution
- Finalization

communication

- Structure of signals
- Send/receive protocols

Abstract Semantics – How Components Interact

flow of control

- Initialization
- Execution
- Finalization

communication

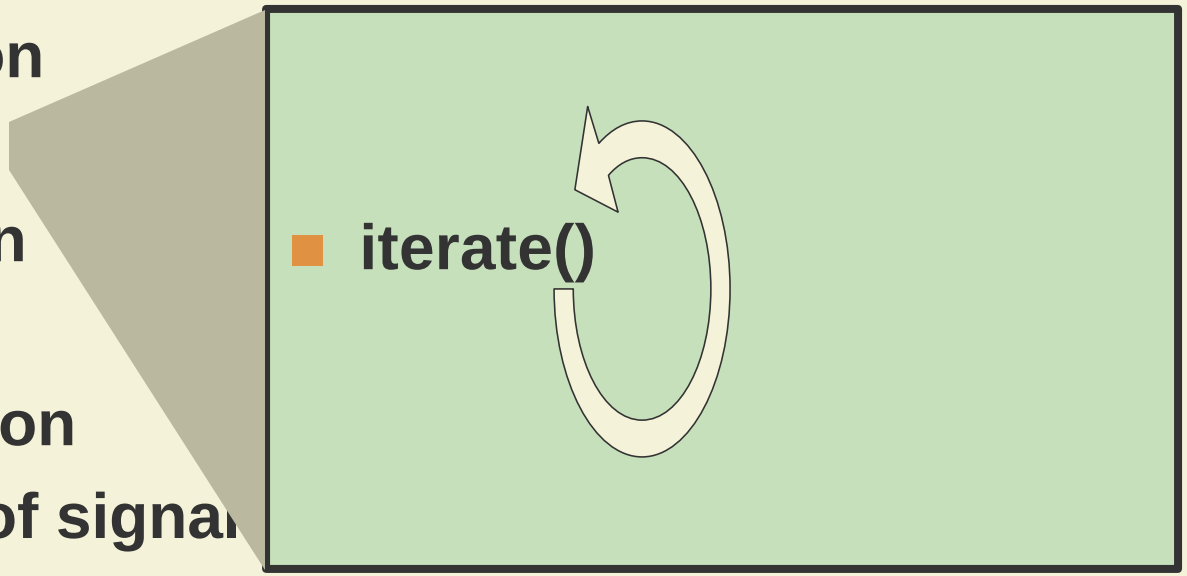
- Structure of signal
- Send/receive protocols

- **preinitialize()**
  - declare static information, like type constraints, scheduling properties, temporal properties, structural elaboration
- **initialize()**
  - initialize variables

Abstract Semantics – How Components Interact

flow of control

- Initialization
- Execution
- Finalization



communication

- Structure of signal
- Send/receive protocols

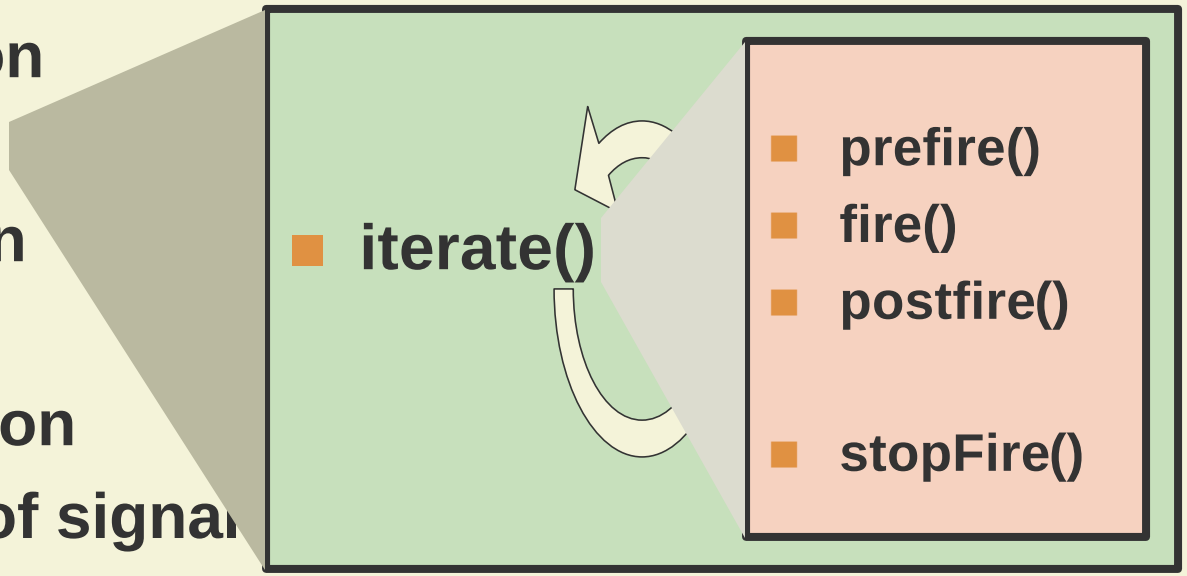
Abstract Semantics – How Components Interact

flow of control

- Initialization
- Execution
- Finalization

communication

- Structure of signal
- Send/receive protocols



The Key Action Methods

- **Prefire()**
  - obtain required resources
  - may read inputs
  - may start computations
  - returns a boolean indicating readiness
- **Fire()**
  - produces results
- **Postfire()**
  - commits state updates (transactional)
- **StopFire()**
  - request premature termination

All of these are atomic (non-preemptible)

This Abstract Semantics has Worked For

- Continuous-time models
- Finite state machines

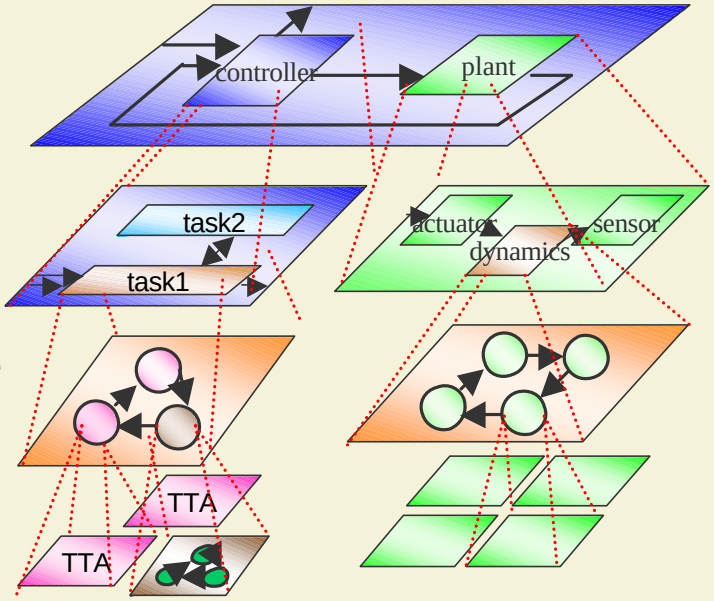
} **Hybrid systems**

- Dataflow
- Discrete-event systems
- Synchronous/reactive systems
- Time-driven models (Giotto)
- …

Can we make it work for priority-driven multitasking (RTOS style)?

Benefits

- **Composable semantics**
  - arbitrarily deep hierarchies
  - heterogeneous hierarchies
- **Precise mode switching**
  - nest FSMs with anything else

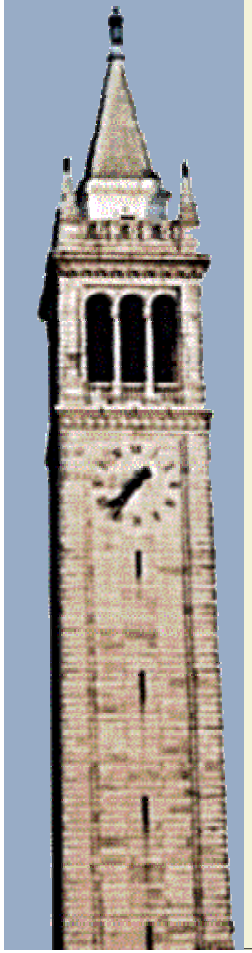


Hierarchical, heterogeneous, system-level model

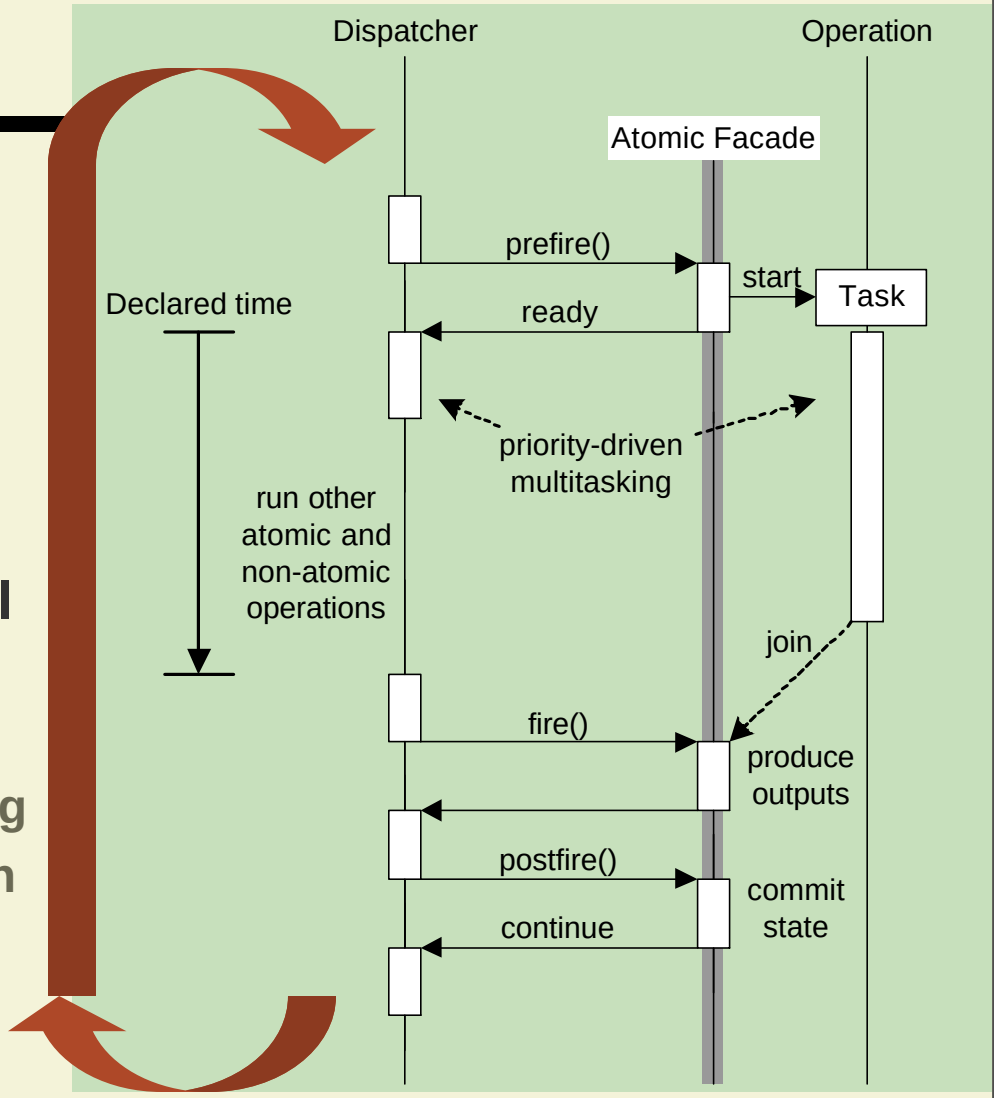
RTOS Domain

- **Objective:**
  - understand and improve OCP semantics
  - support priority-driven preemptive scheduling
  - use atomic execution, to get composability
  - solve the precise mode change problem
- **Solution:**
  - Atomic execution when possible
  - Façade to long-running processes when not

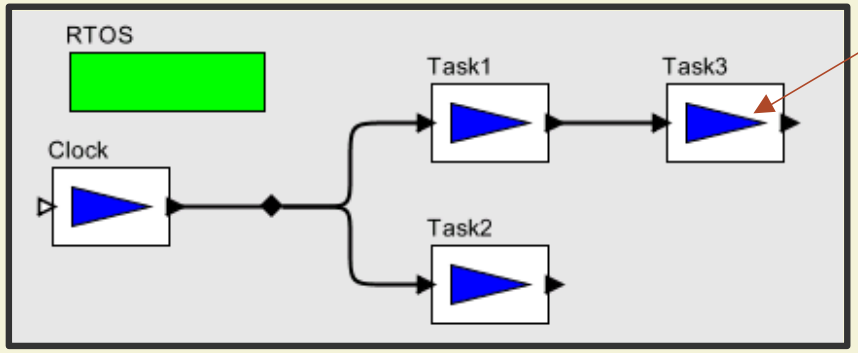
Atomic Façade to Long-Running Computations



- Each component defines the interaction between the atomic façade and the long-running process.
- There are several useful patterns:
  - allow task to complete
  - enforce declared timing
  - “anytime” computation
  - transactional



RTOS Domain Implementation

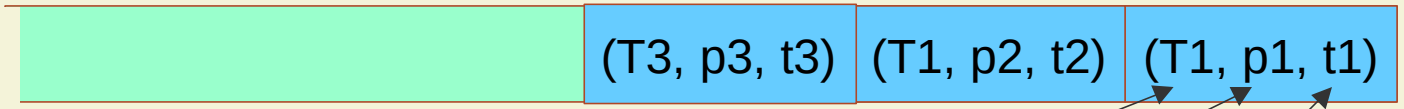


- priority
- executionTime

RT-Q

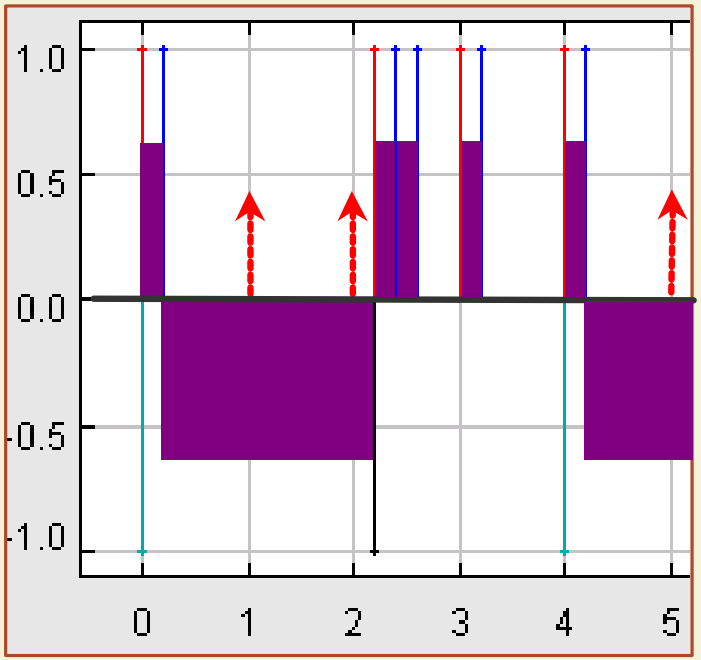
(actor, output time)

OS-Q

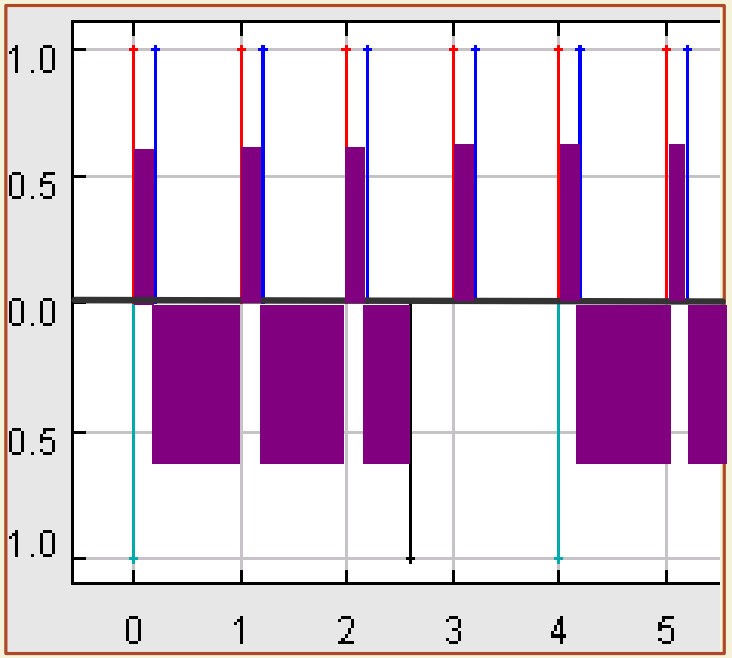


(task, priority, remaining processing time)

Example: two simple tasks

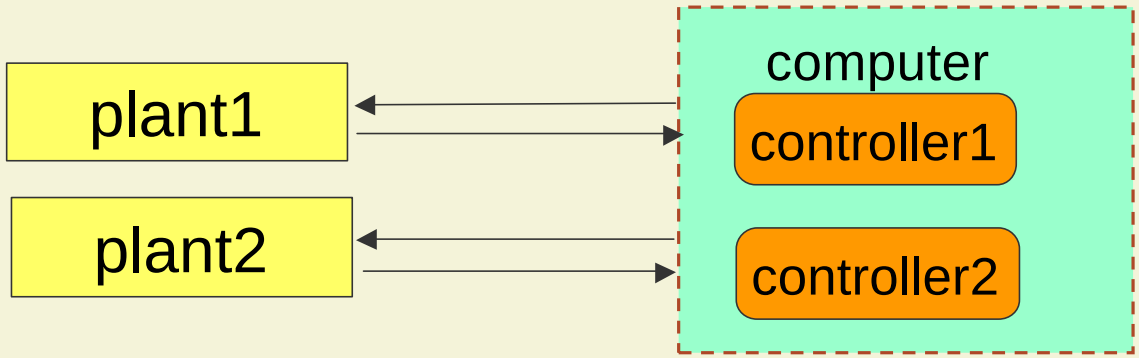


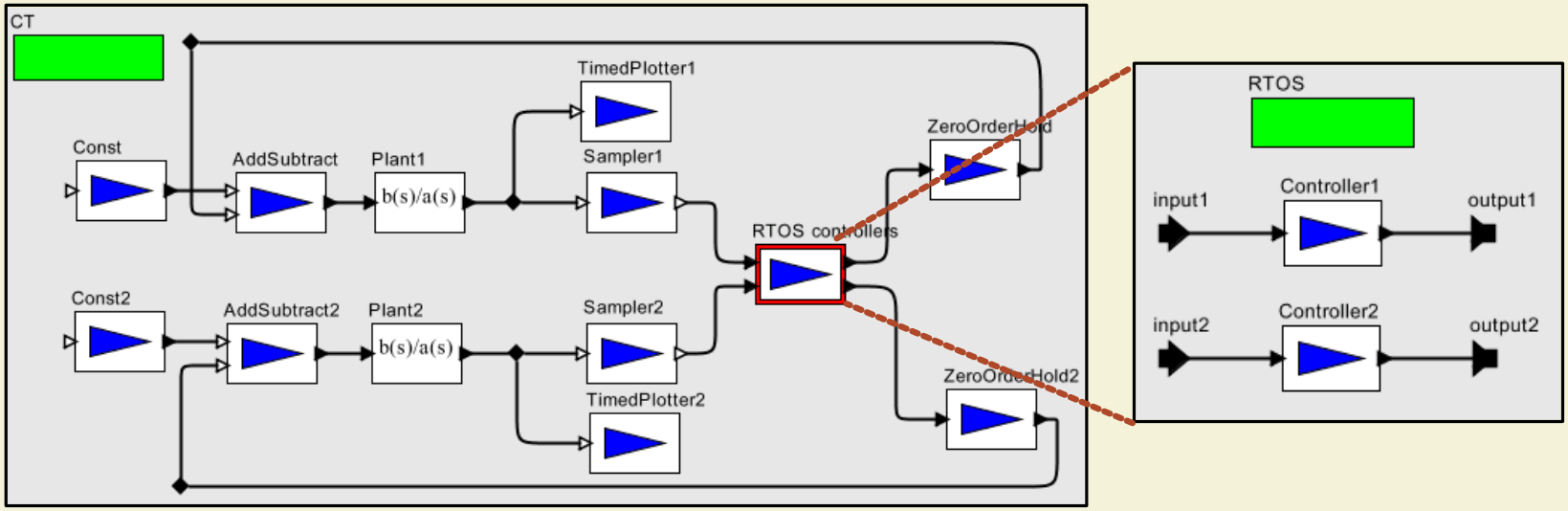
nonpreemptive



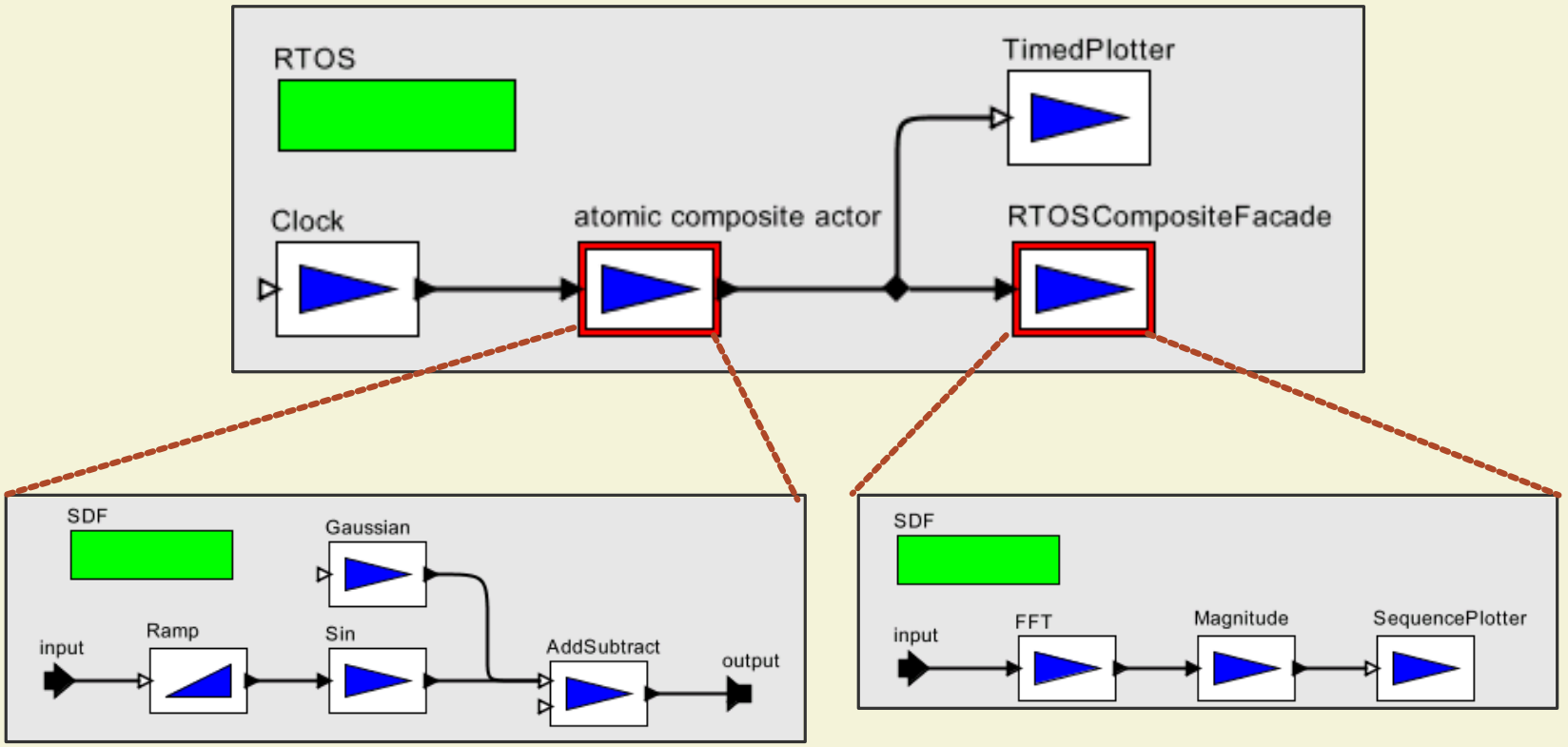
preemptive

Inter-domain example: shared-resource controllers





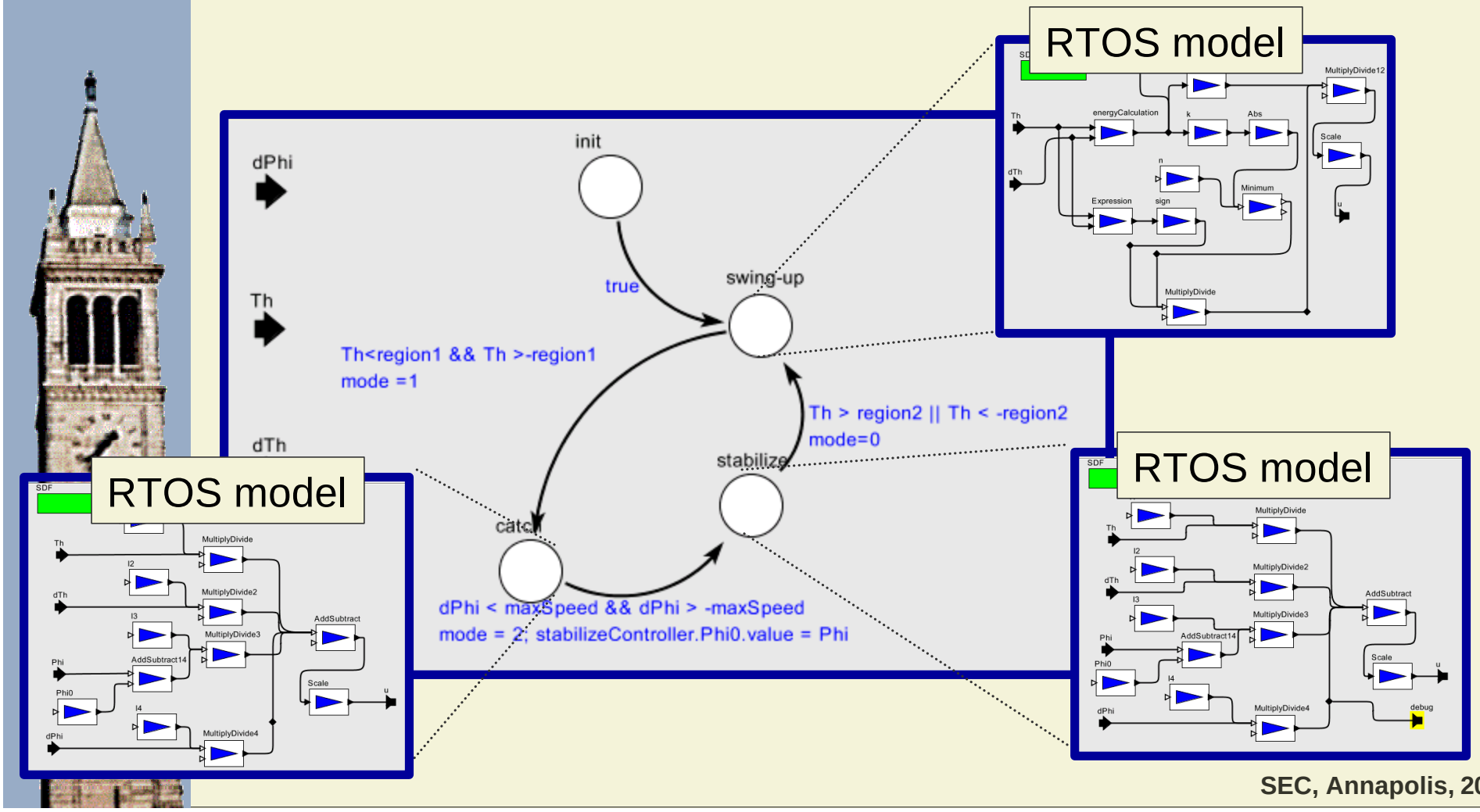
Background process example: Data acquisition and processing



atomic

background processes

What a Modal Control System Might Look Like



Conclusion

***Systematic*, *principled*, *real-time*, *heterogeneous*, hierarchical composition of:**

- **Processes and/or threads**
- **Finite automata (mode controllers)**
- **Other models of computation**
  - **Continuous-time models**
  - **Dataflow models**
  - **…**

The key is the abstract semantics of Ptolemy II, which defines hierarchical heterogeneous composition of models of computation.