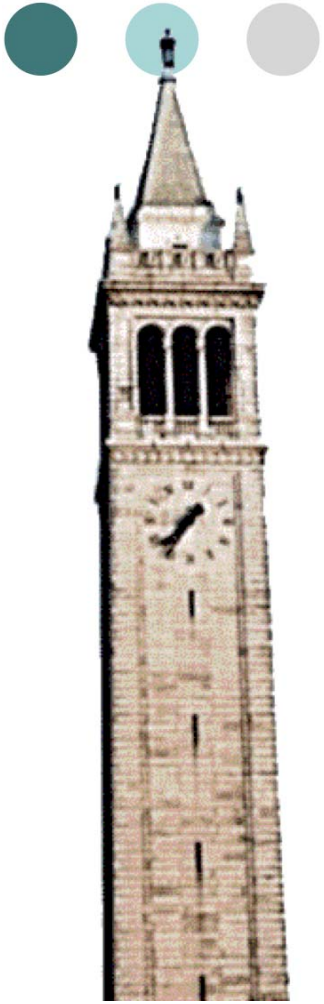


Model-Based Design for Signal Processing Systems

Edward A. Lee

Robert S. Pepper Distinguished Professor
UC Berkeley

Invited Keynote Talk
IEEE Workshop on Signal Processing Systems (SiPS)
Tampere, Finland

October 7-9, 2009

Context of my work: Chess: Center for Hybrid and Embedded Software Systems




This center, founded in 2002, blends systems theorists and application domain experts with software technologists and computer scientists.

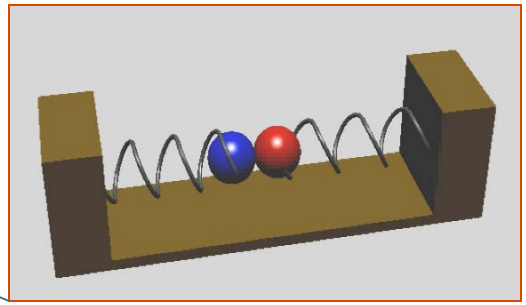
Board of Directors
- Edward A. Lee
- Alberto Sangiovanni-Vincentelli
- Shankar Sastry
- Claire Tomlin

Executive Director
- Christopher Brooks

Other key faculty at Berkeley
- Dave Auslander
- Ruzena Bajcsy
- Raz Bodik
- Karl Hedrick
- Kurt Keutzer
- George Necula
- Masayoshi Tomizuka
- Pravin Varaiya

Some Research Projects
- Precision-timed (PRET) machines
- Distributed real-time computing
- Systems of systems
- Theoretical foundations of CPS
- Hybrid systems
- Design technologies
- Verification
- Intelligent control
- Modeling and simulation

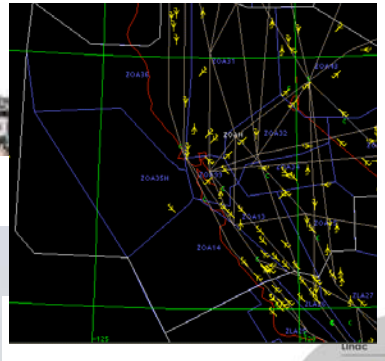
Applications
- Building systems
- Automotive
- Synthetic biology
- Medical systems
- Instrumentation
- Factory automation
- Avionics



Cyber-Physical Systems (CPS):

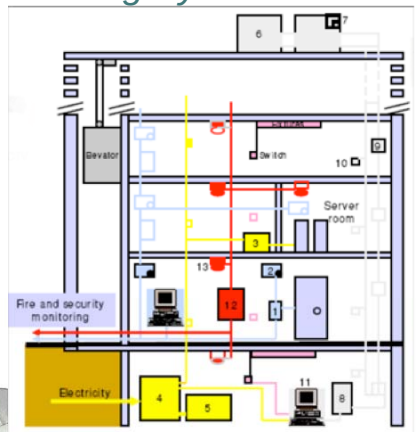
Orchestrating networked computational resources with physical systems

Avionics



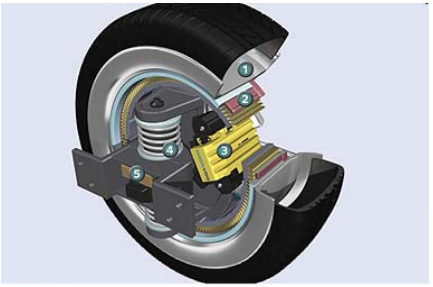
Transportation (Air traffic control at SFO)

Building Systems

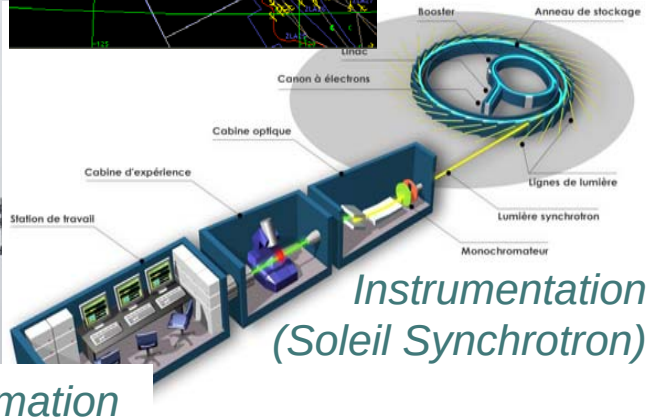


Telecommunications

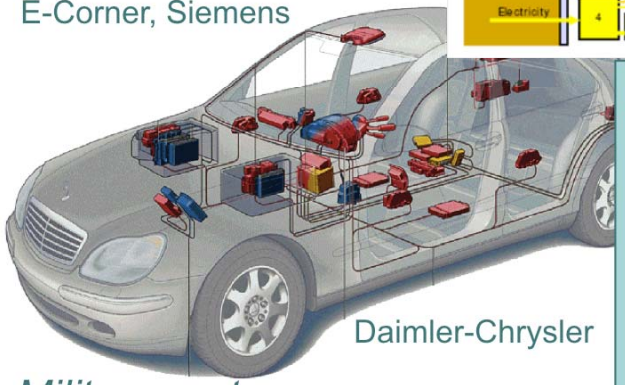
Automotive



E-Corner, Siemens

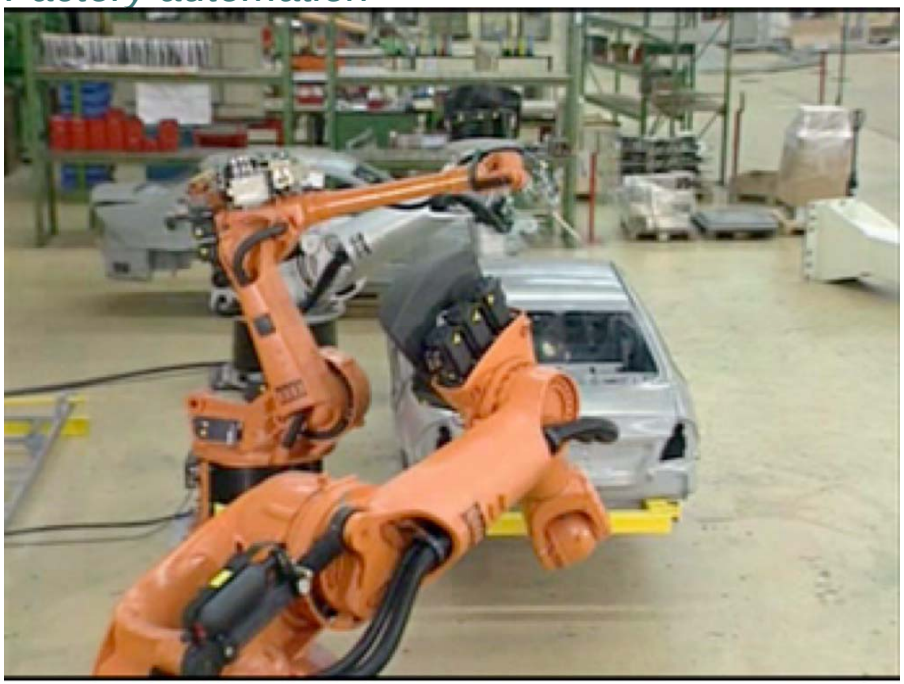


Instrumentation (Soleil Synchrotron)



Daimler-Chrysler

Factory automation

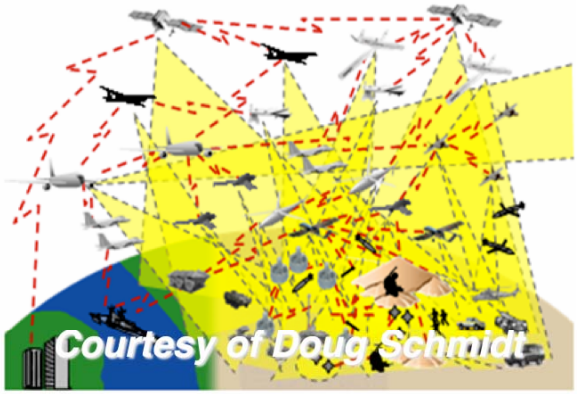


Courtesy of Kuka Robotics Corp.

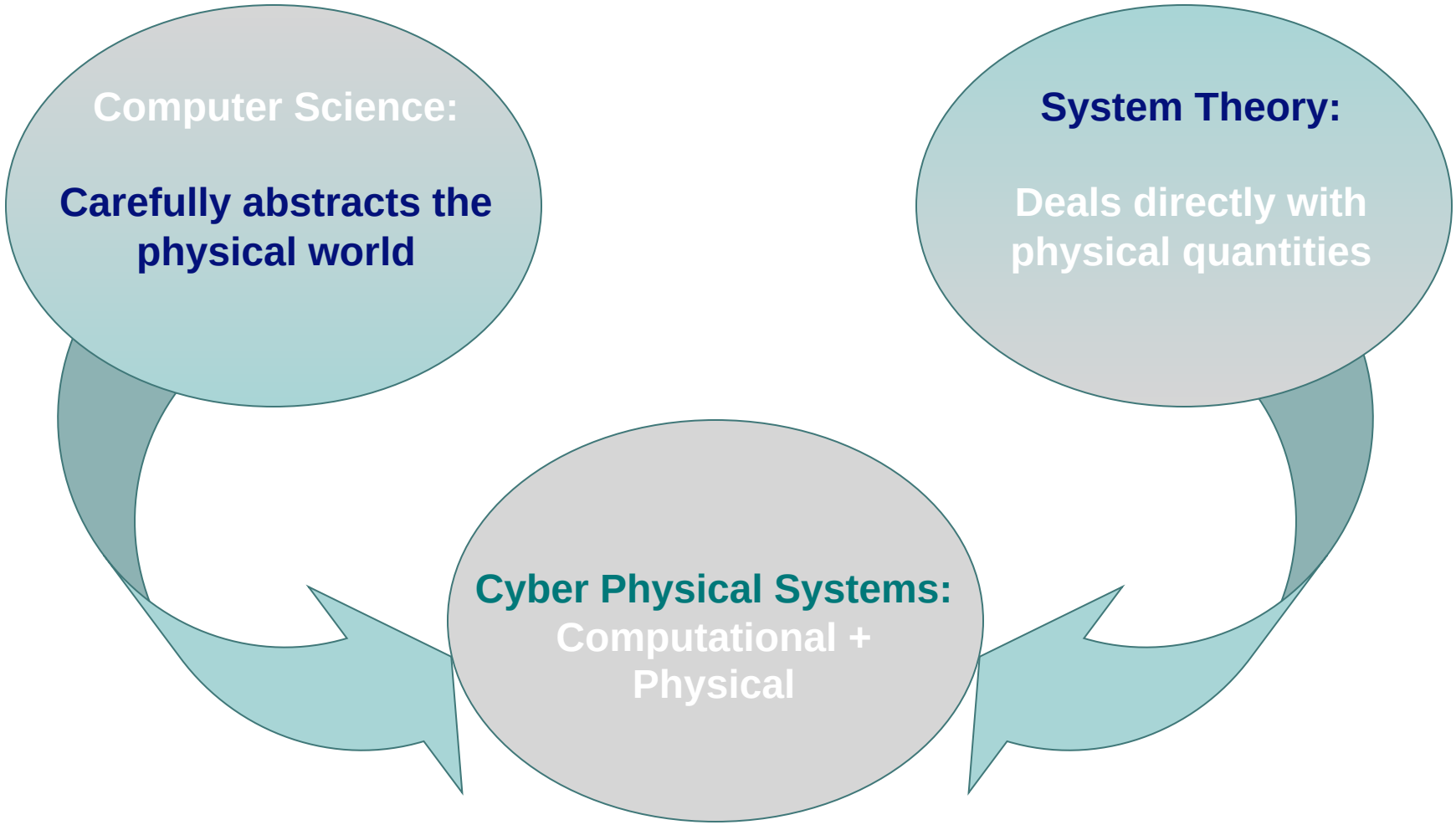
Power generation and distribution

Courtesy of General Electric

Military systems:



CPS is Multidisciplinary

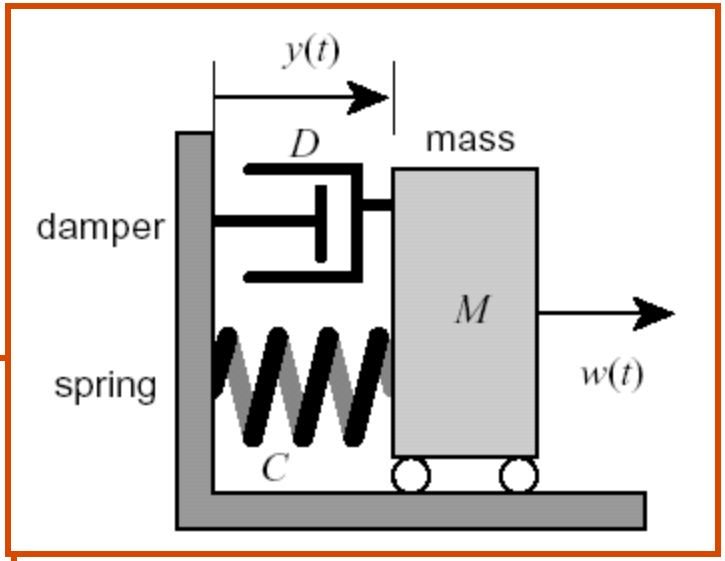


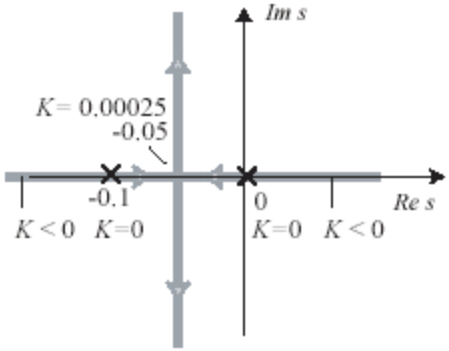
Theme of This Talk

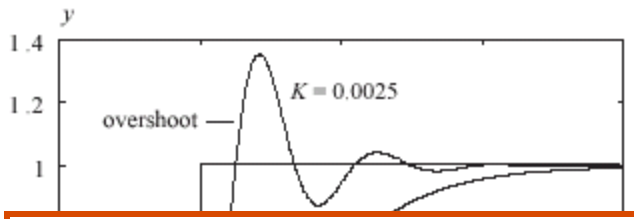
Opportunities on the systems side...

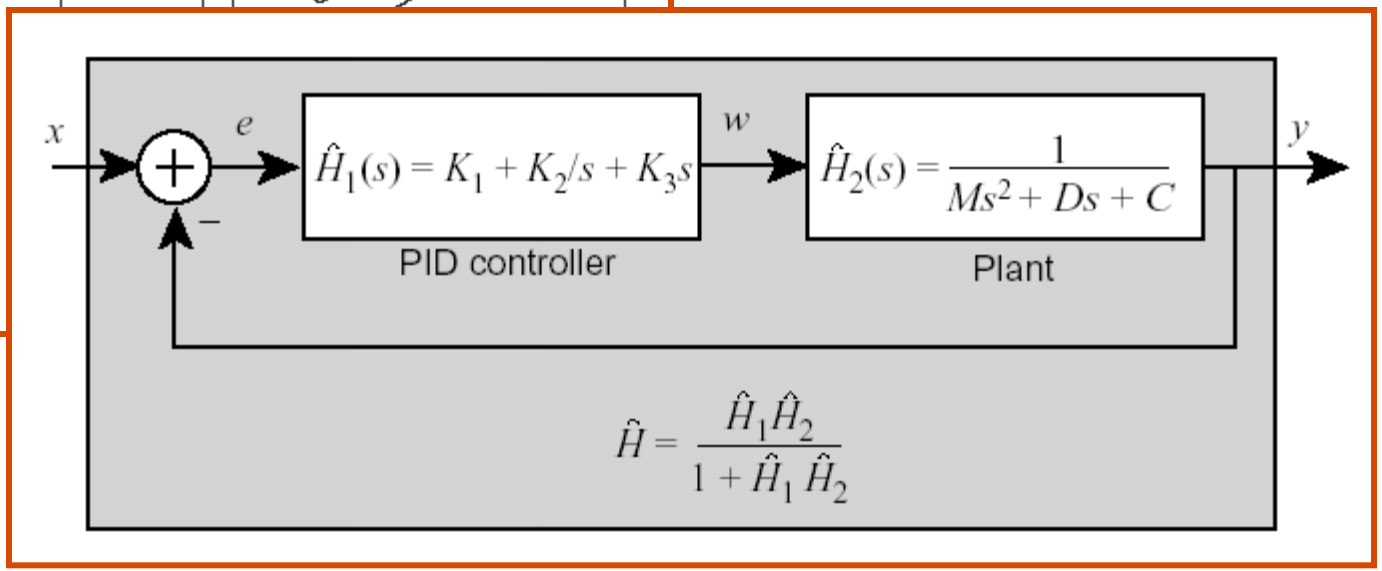
Models on the Systems Side

- Models of continuous-time dynamics
- Sophisticated stability analysis
- *But not accurate for software controllers*



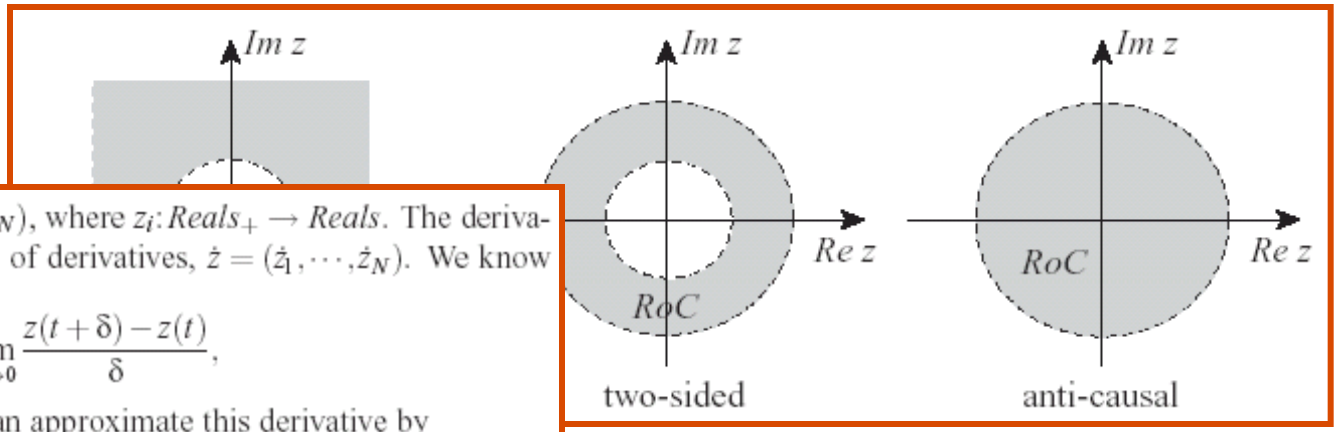






Discretized Model – A Step Towards Software

- Numerical integration techniques provided sophisticated ways to get from the continuous idealizations to computable algorithms.
- Discrete-time signal processing techniques offer the same sophisticated stability analysis as continuous-time methods.
- *But it's still not accurate for software controllers*



In general, z is an N-tuple, $z = (z_1, \cdots, z_N)$, where z_i: $Reals_+ \rightarrow Reals$. The derivative of an N-tuple is simply the N-tuple of derivatives, $\dot{z} = (\dot{z}_1, \cdots, \dot{z}_N)$. We know from calculus that

$$\dot{z}(t) = \frac{dz}{dt} = \lim_{\delta \to 0} \frac{z(t+\delta) - z(t)}{\delta},$$

and so, if $\delta > 0$ is a small number, we can approximate this derivative by

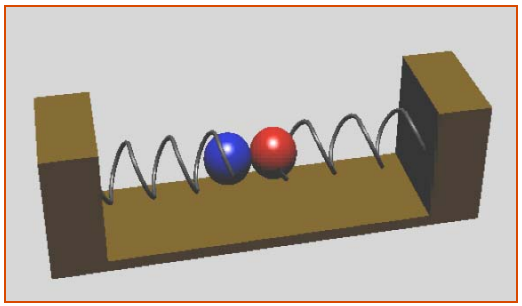
$$\dot{z}(t) \approx \frac{z(t+\delta) - z(t)}{\delta}.$$

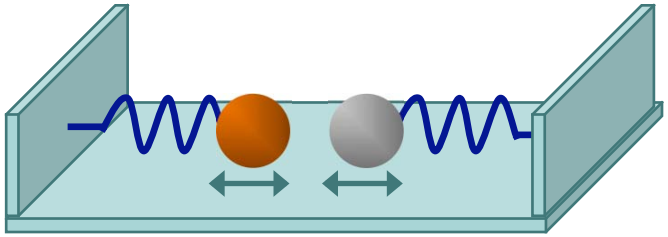
Using this for the derivative in the left-hand side of (5.50) we get

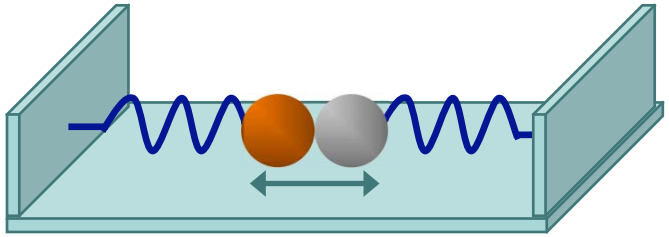
$$z(t+\delta) - z(t) = \delta g(z(t), v(t)). \qquad (5.51)$$

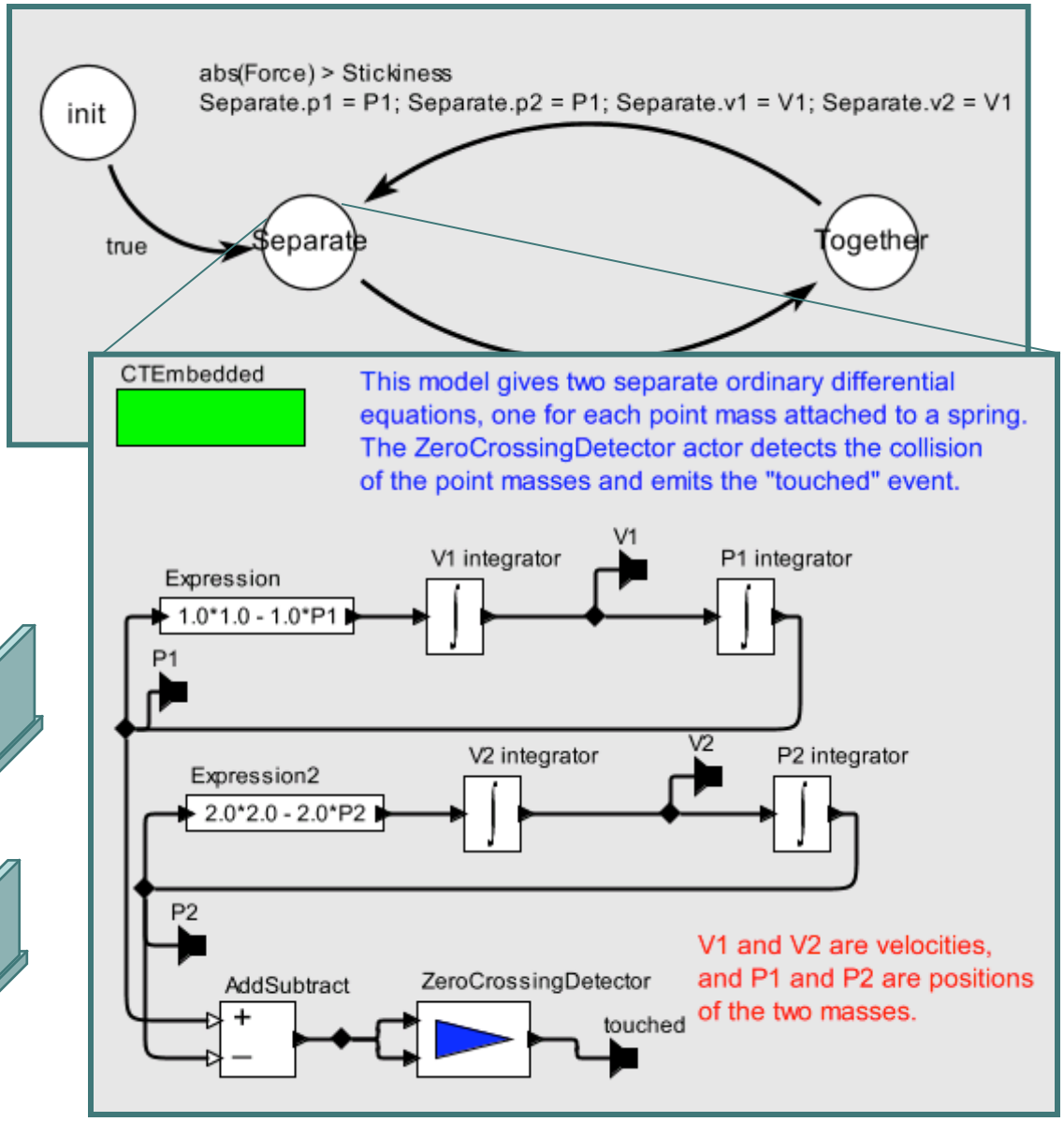
Hybrid Systems – Reconciling Continuous & Discrete, Version 1.0

But it's still not accurate for software controllers





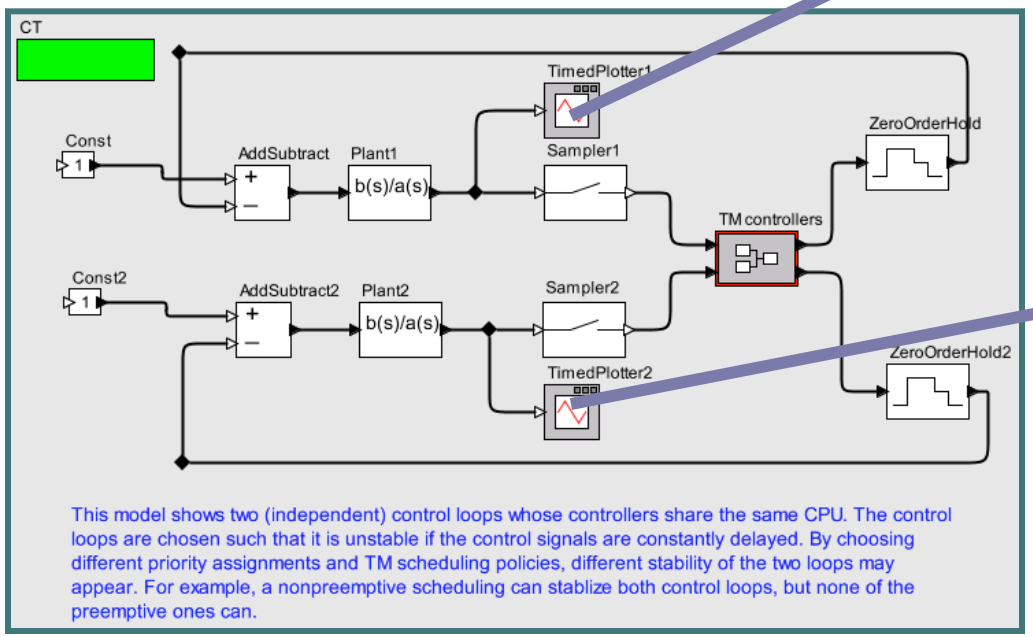


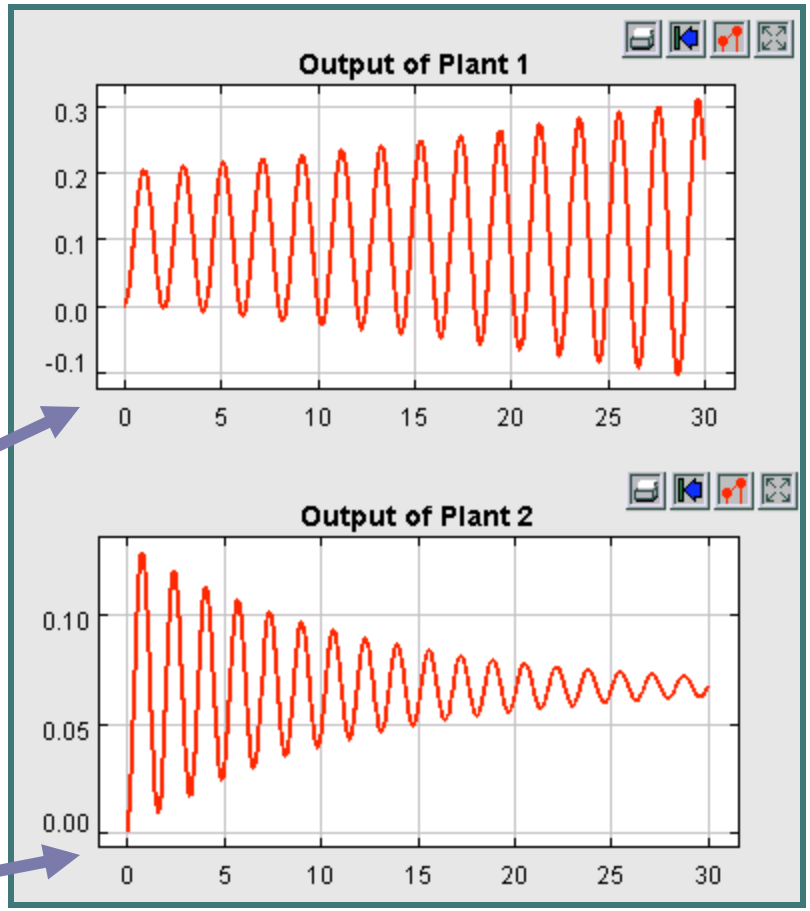


Timing in Software is More Complex Than What the Theory Deals With

An example, due to Jie Liu, models two controllers sharing a CPU under an RTOS. Under preemptive multitasking, only one can be made stable (depending on the relative priorities). Under non-preemptive multitasking, both can be made stable.

Where is the theory for this?





Another Example: Sensor Fusion

Consider a real-time program on an embedded computer that is connected to two sensors *A* and *B*, each providing a stream of data at a normalized rate of one sample per time unit. The data from the two sensors is deposited by an interrupt service routine into a memory location.

Assume a program that looks like this:

```
while(true) {
    wait for new data from A;
    wait a fixed amount of time T;
    observe registered data from B;
    average data from A and B;
}
```

Example is due to Stephen Neuendorffer.

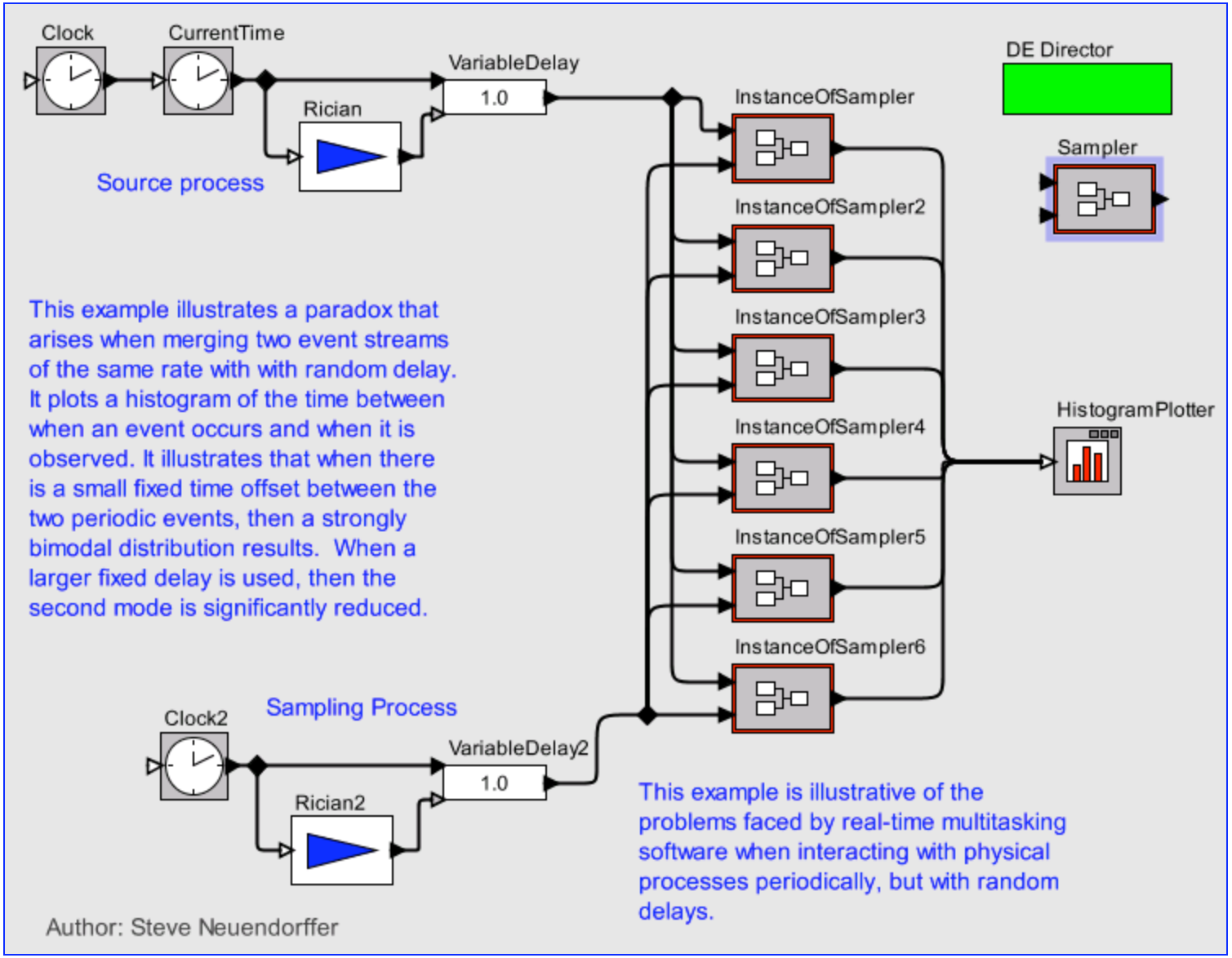
The Design Question

Assume that there are random delays in the software (due to multitasking, interrupt handling, cache management, etc.) for both the above program and the interrupt service routines.

What is the best choice for the value for T?

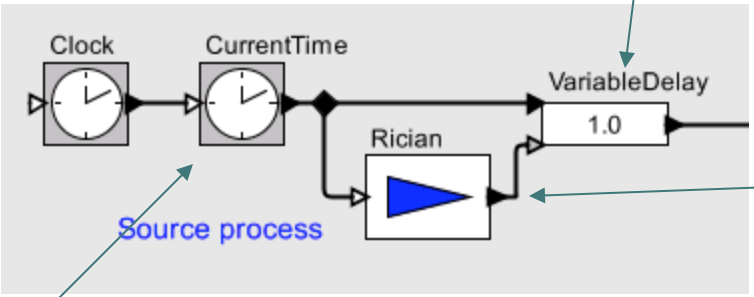
One way to frame the question: How old is the data from *B* that will be averaged with the data from *A*?

A Model that Measures for Various Values of T



Modeling Random Delay in Sensor Data Using a Discrete-Event Modeling Framework

Given an event with time stamp t on the upper input, the VariableDelay actor produces an output with the same value but time stamp $t + t'$, where t' is the value of the most recently seen event on the lower input.

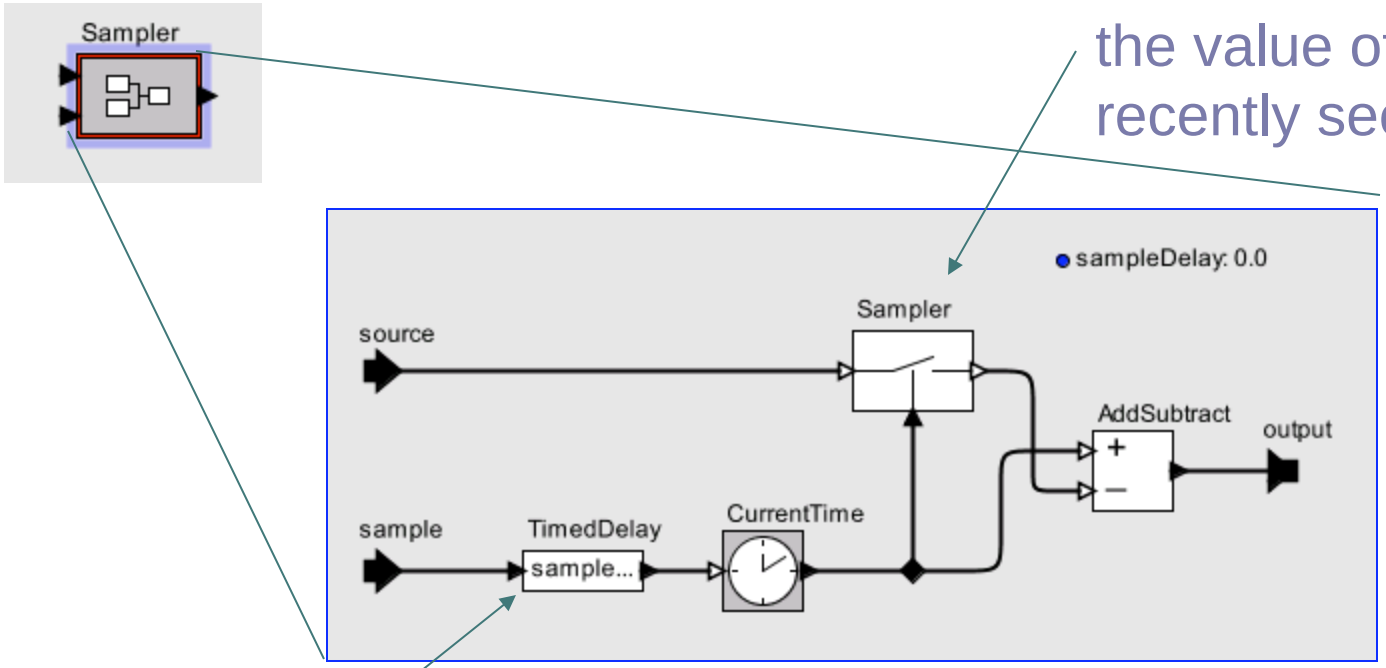


The Rician actor, when triggered, produces an output event with a non-negative random value and with time stamp equal to that of the trigger event.

Given an input event at time t with any value, the CurrentTime actor outputs the double t with time stamp t.

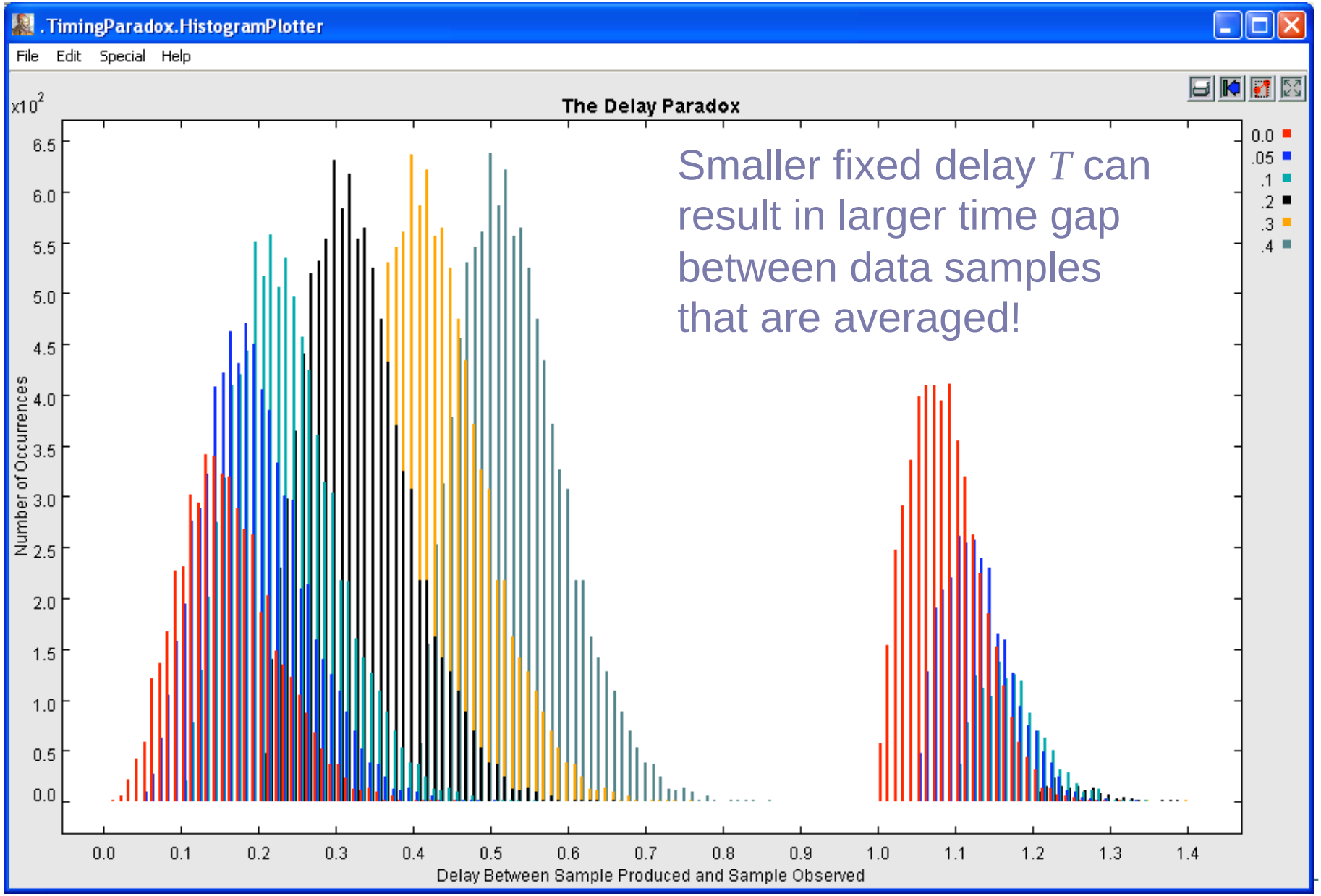
Actor-Oriented Sampler Class

Given a trigger event with time stamp t the Sampler actor produces an output event with value equal to the value of the most recently seen input event.



The TimedDelay actor transfers every input event to the output with a fixed increment in the time stamp. Here, the value is sampleDelay, a parameter of the composite actor.

Time gap between samples for various values of T



A Real-World Application: Automotive Engine Controller

Embedded software uses timers, interrupts, threads, shared memory, priorities, and mutual exclusion. How to relate the systems theory with the software behavior?

The Standard Model in Signal Processing

Continuous time signal: $x : \mathbb{R} \rightarrow \mathbb{R}$

Discrete-time signal: $x : \mathbb{N} \rightarrow \mathbb{R}$

Problems with this model:

- Simultaneity: $x(t_1)$ and $y(t_2)$ may be in different parts of the system. What does it mean for $t_1 = t_2$?
- Causality: Suppose x crosses a threshold at t and causes y to be discontinuous at t. What is $y(t)$? Are $y(t)$ and $x(t)$ simultaneous? How is their causal connection represented?
- Discreteness: Suppose a signal x cannot be meaningfully defined for some interval of time?
- Synchrony: Suppose $x(n)$ and $y(n)$ have a non-trivial phase relationship in their samples. How to represent?

A Richer Model of Signals

A signal is a partial function $x : T \rightarrow A$, where A is a set of possible event values (a data type and maybe an element indicating "absent"), and T is a totally or partially ordered set of *tags* that represent *time stamps* and ordering of events at the same time stamp.

The standard model is a special case of this model, but there may be better alternatives.

First Attempt at a Specific Richer Model for Signals

Let $\mathbb{R}_+$ be the non-negative real numbers. Let V be an arbitrary family of values (a data type, or alphabet). Let

$$V_\varepsilon = V \cup \{\varepsilon\}$$

be the set of values plus "absent." Let s be a signal, given as a partial function:

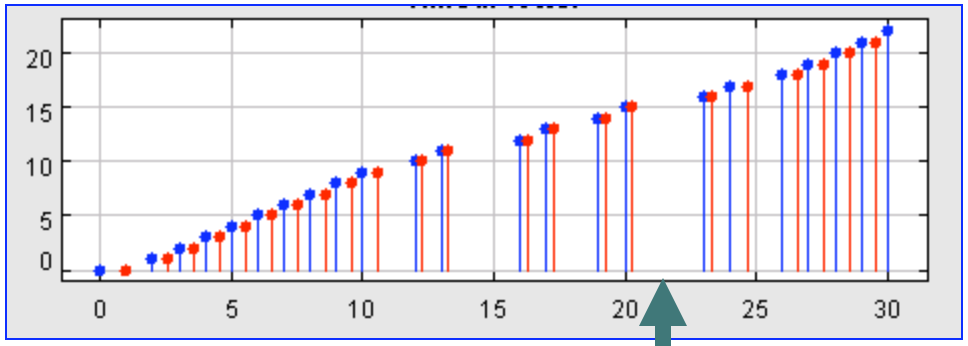
$$s : \mathbb{R}_+ \rightharpoonup V_\varepsilon$$

defined on an initial segment of $\mathbb{R}_+$

This First Attempt can Model Nontrivial Timing

$$s : \mathbb{R}_+ \rightharpoonup V_\varepsilon$$

Values V

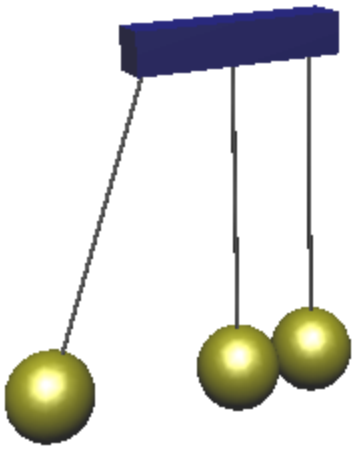


Initial segment $I \subseteq \mathbb{R}_+$ where the signal is defined.

Absent: $s(\tau) = \varepsilon$ for almost all $\tau \in I$.

This model is still not rich enough because it does not allow a signal to have multiple events at the same time.

Example Motivating the Need for Simultaneous Events Within a Signal



Newton's Cradle:

- Steel balls on strings
- Collisions are discrete events
- Position, speed, acceleration are all signals
- So is momentum

Momentum of the middle ball has three values at the time of collision.

Other examples:

- Batch arrivals at a queue.
- Software sequences abstracted as instantaneous.
- Transient states of a system.

A Better Model for Signals: *Superdense Time*

Let $T = \mathbb{R}_+ \times \mathbb{N}$ be a set of "tags" where $\mathbb{N}$ is the natural numbers, and give a signal s as a partial function:

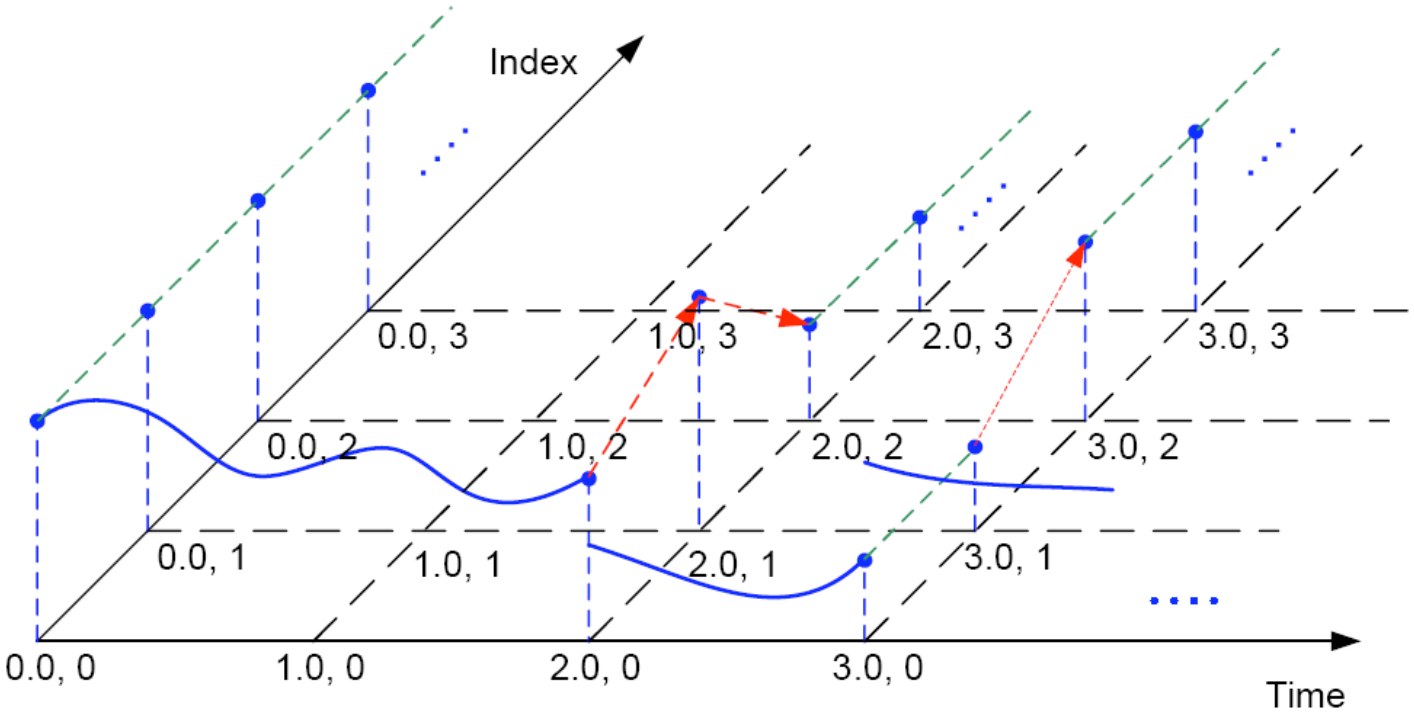
$$s : T \rightharpoonup V_\varepsilon$$

defined on an initial segment of T, assuming a lexical ordering on T:

$$(t_1, n_1) \leq (t_2, n_2) \iff t_1 < t_2, \text{ or } t_1 = t_2 \text{ and } n_1 \leq n_2 .$$

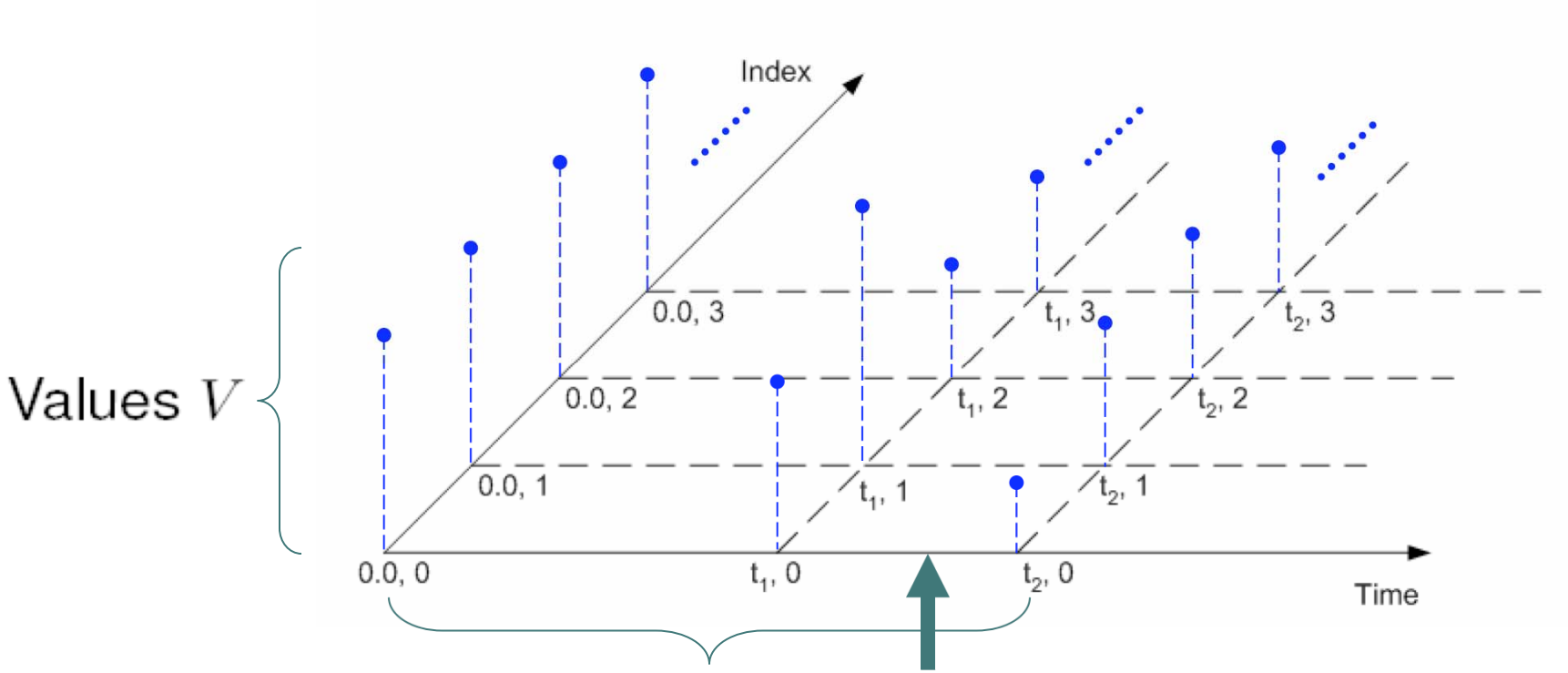
This allows signals to have a sequence of values at any real time t.

Superdense Time



At each tag, the signal has exactly one value. At each time point, the signal has an infinite number of values. The red arrows indicate value changes between tags, which correspond to discontinuities.

Superdense Time Supports Discrete Signals (using an "absent" value)



Initial segment $I \subseteq \mathbb{R}_+ \times \mathbb{N}$ where the signal is defined

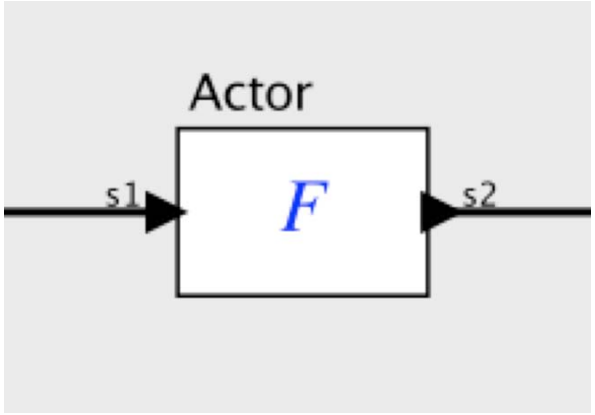
Absent: $s(\tau) = \varepsilon$ for almost all $\tau \in I$.

Using Superdense Time to Build Models of Discrete-Event (DE) Signals

$$s : T \rightharpoonup V_\varepsilon$$

- A *tag* is a time-index pair, $\tau = (t, n) \in T = \mathbb{R}_+ \times \mathbb{N}$.
- An *event* is a tag-value pair, $e = (\tau, v) \in T \times V$.
- $s(\tau)$ is an event if $s(\tau) \neq \varepsilon$.

Actors

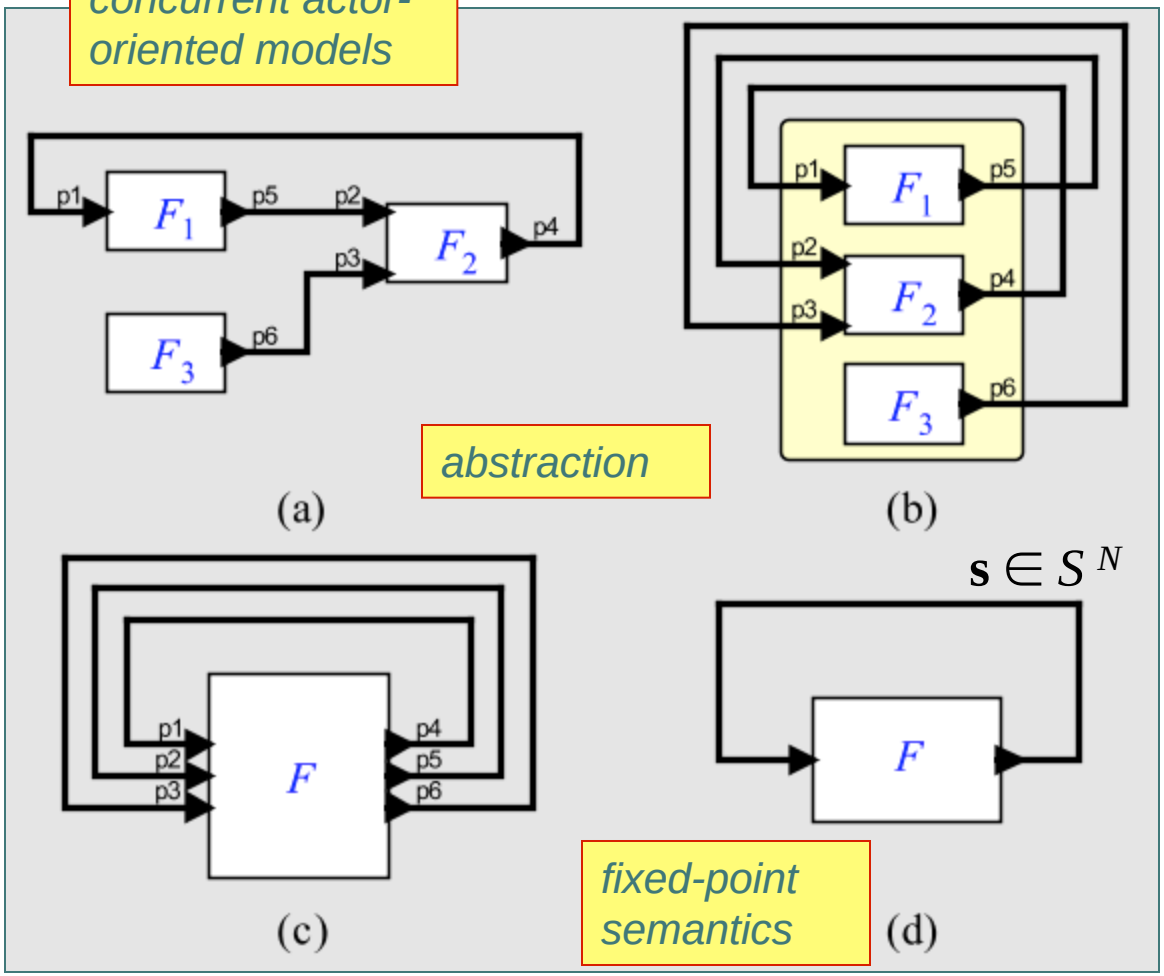


An actor is a relation on signals. It is useful to impose certain constraints:

- It is a function mapping inputs to outputs.
- It is prefix monotonic
- It is (Scott) continuous
- It is causal
- …

Model-Based Design with Superdense Time

concurrent actor-oriented models



super-dense time

- Signal: $s : \mathbb{R}_+ \times \mathbb{N} \rightharpoonup V_\varepsilon$
- Set of signals: S
- Tuples of signals: $\mathbf{s} \in S^N$
- Actor: $F : S^N \rightarrow S^M$

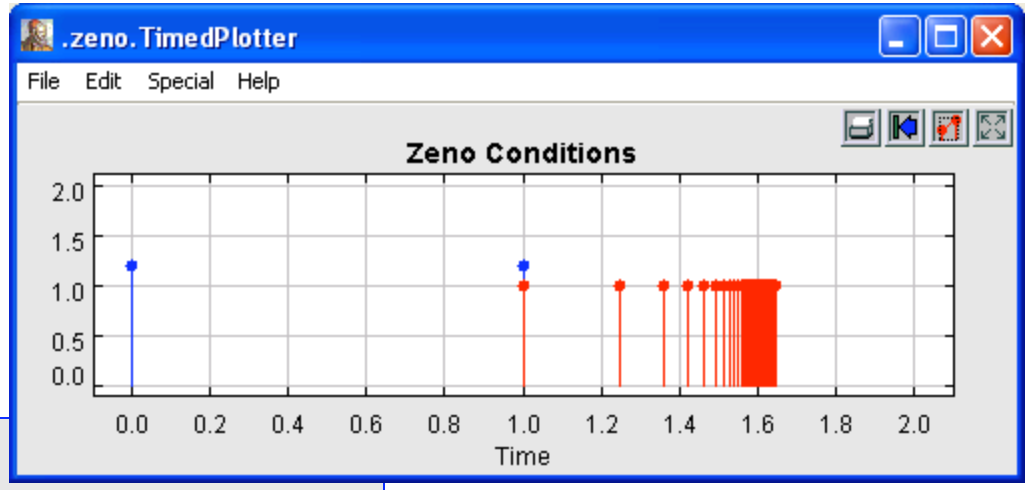
A unique least fixed point, $\mathbf{s} \in S^N$ such that $F(\mathbf{s}) = \mathbf{s}$, exists and be constructively found if S^N is a CPO and F is (Scott) continuous.

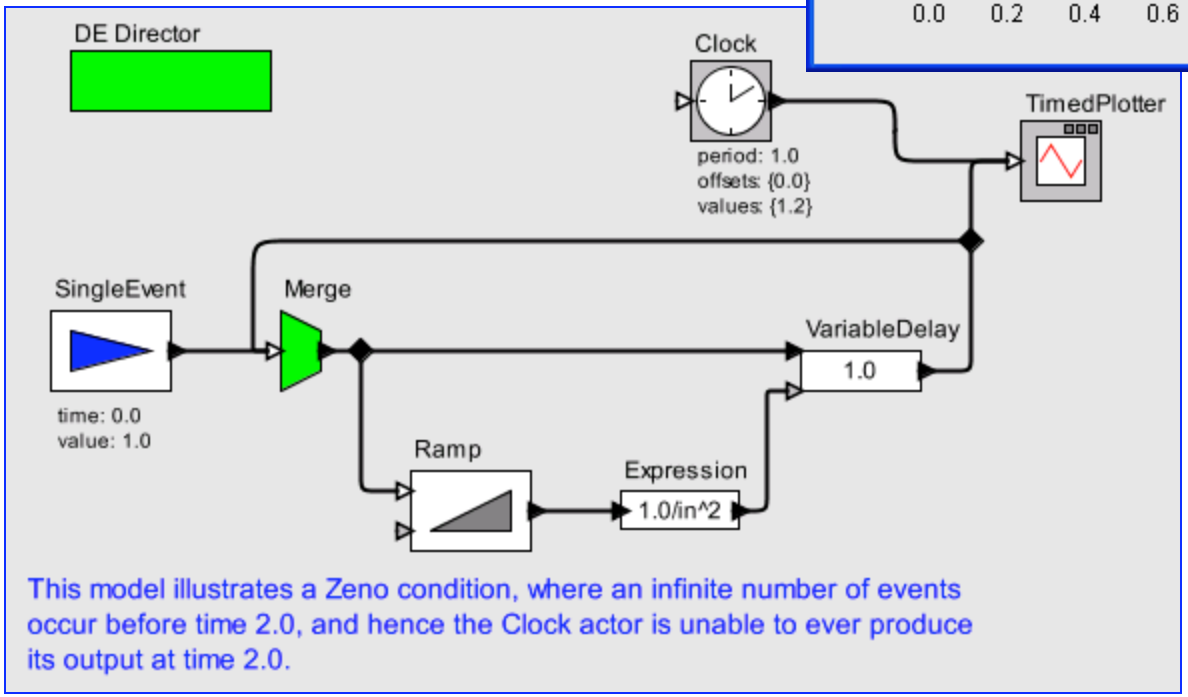
Causal systems operating on signals are usually naturally (Scott) continuous.

Discrete-Event (DE) Signals

A signal s is *discrete* if there is an *order embedding* from its tag set $\pi(s)$ (the tags for which it is defined and not abent) to the integers (under their usual order).

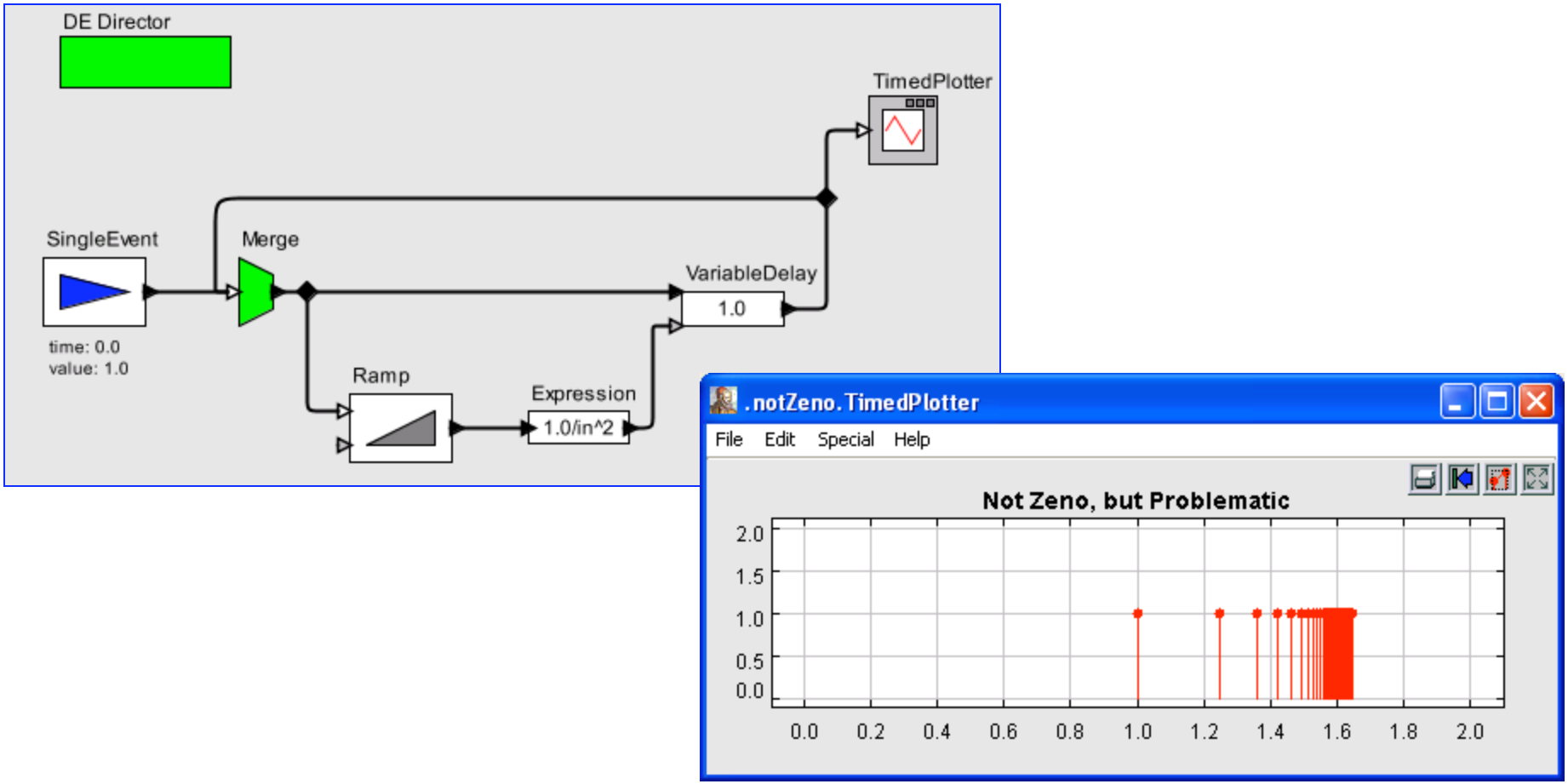
A Zeno system is not discrete.





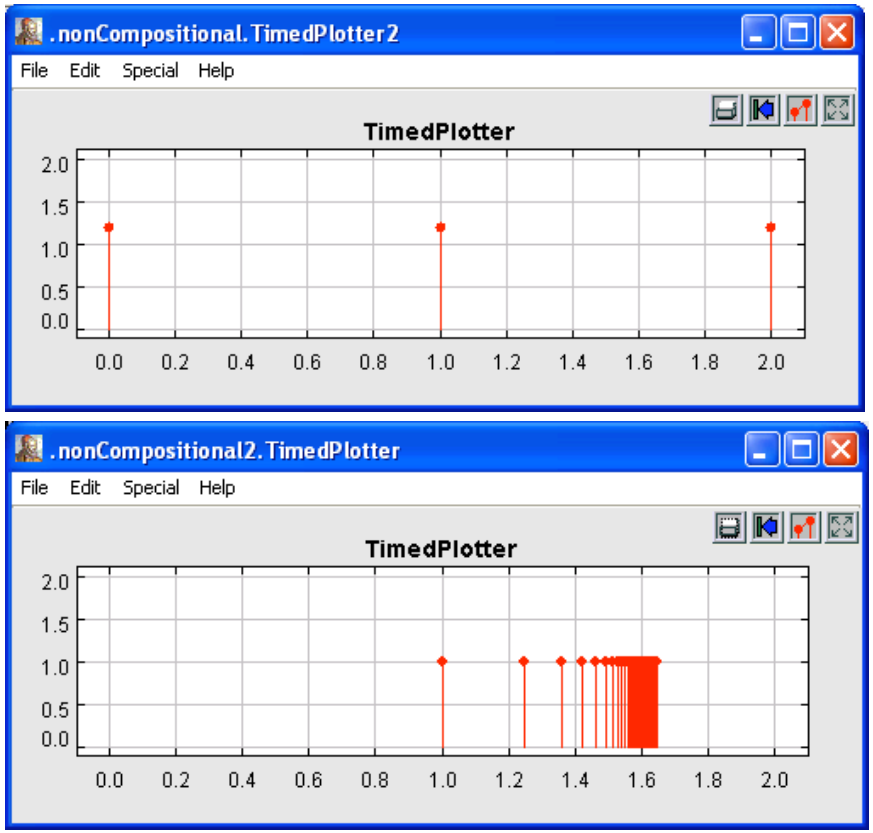
The tag set here includes { 0, 1, 2, …} and { 1, 1.25, 1.36, 1.42, …} .

Is the following system discrete?



Discreteness is Not a Compositional Property

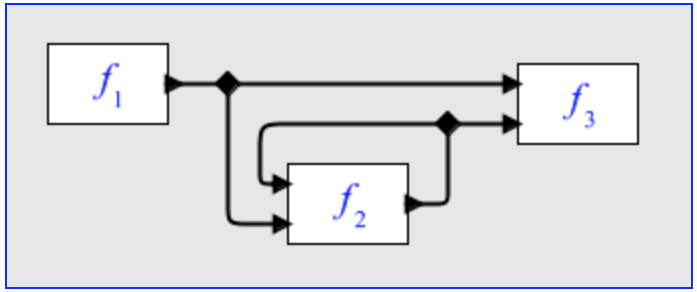
Given two discrete signals s, s' it is not necessarily true that $S = \{ s, s' \}$ is a discrete system.



Putting these two signals in the same model creates a Zeno condition.

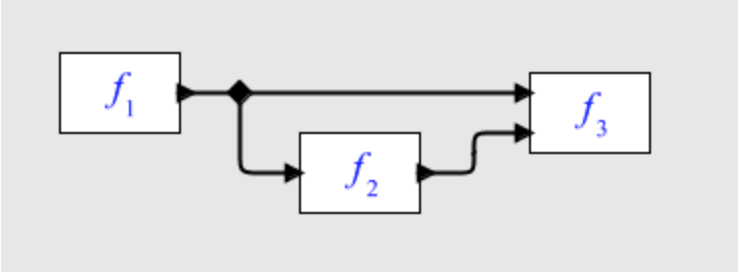
Model-Based Design
Question 1:

Can we find necessary and/or sufficient conditions to avoid Zeno systems? To preserve discreteness under composition?



Model-Based Design
Question 2:

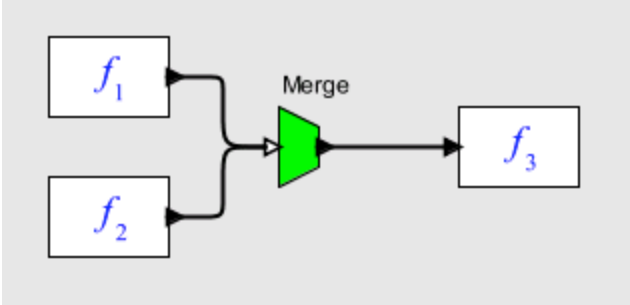
In the following model, if f_2 has no delay, should f_3 see two simultaneous input events with the same tag? Should it react to them at once, or separately?



In Verilog, it is nondeterministic. In VHDL, it sees a sequence of two distinct events separated by “delta time” and reacts twice, once to each input. In Simulink, Labview and the Ptolemy II DE domain, it sees the events together and reacts once.

Model-Based Design
Question 3:

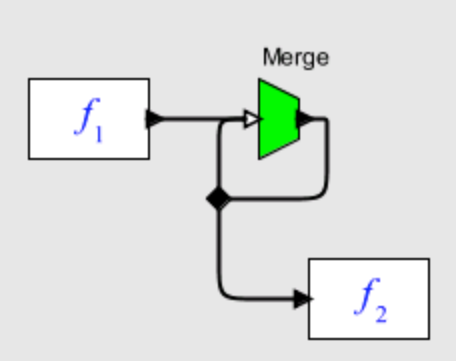
What if the two sources in the following model deliver an event with the same tag? Can the output signal have distinct events with the same tag?



Recall that we require that a signal be a partial function $s : T \rightarrow V$, where V is a set of possible event values (a data type), and T is a totally ordered set of *tags*.

Model-Based Design
Question 4:

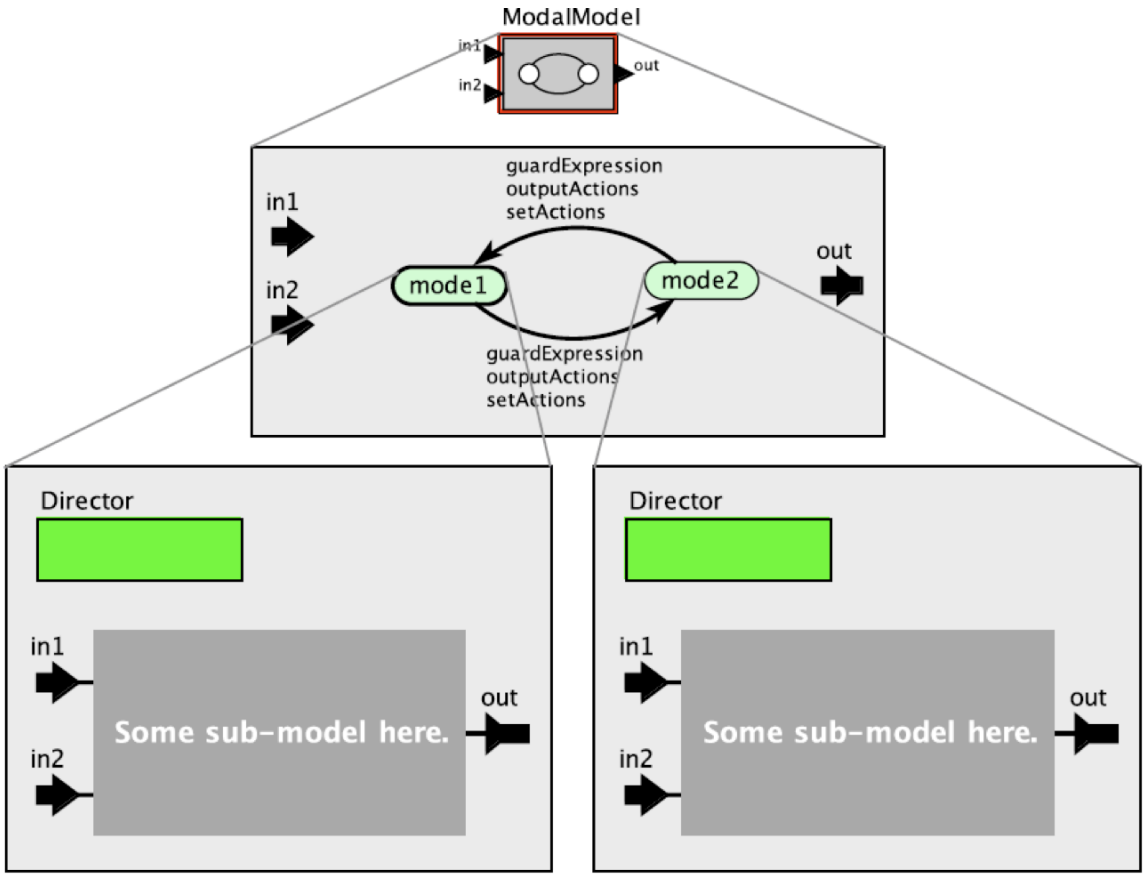
What does this mean?



The Merge presumably does not introduce delay, so what is the meaning of this model?

Model-Based Design
Question 5:

What is the meaning of modal behavior?

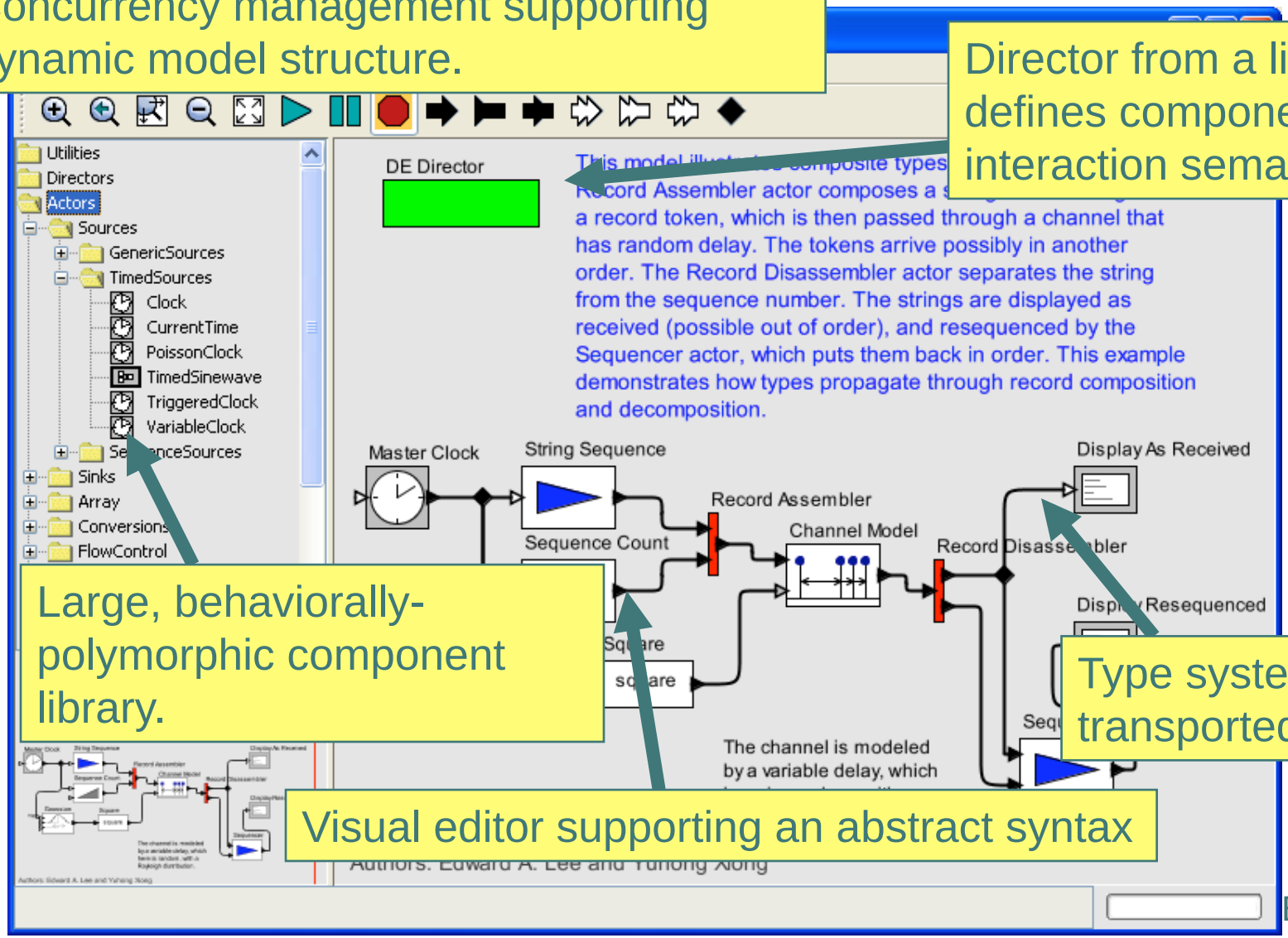


- Do transitions take time?
- Can states be transient (where you spend zero time in them)?
- Can submodels share state?

Ptolemy II: Our Laboratory for Experiments with Techniques for Model-Based Design

Concurrency management supporting dynamic model structure.

Director from a library defines component interaction semantics

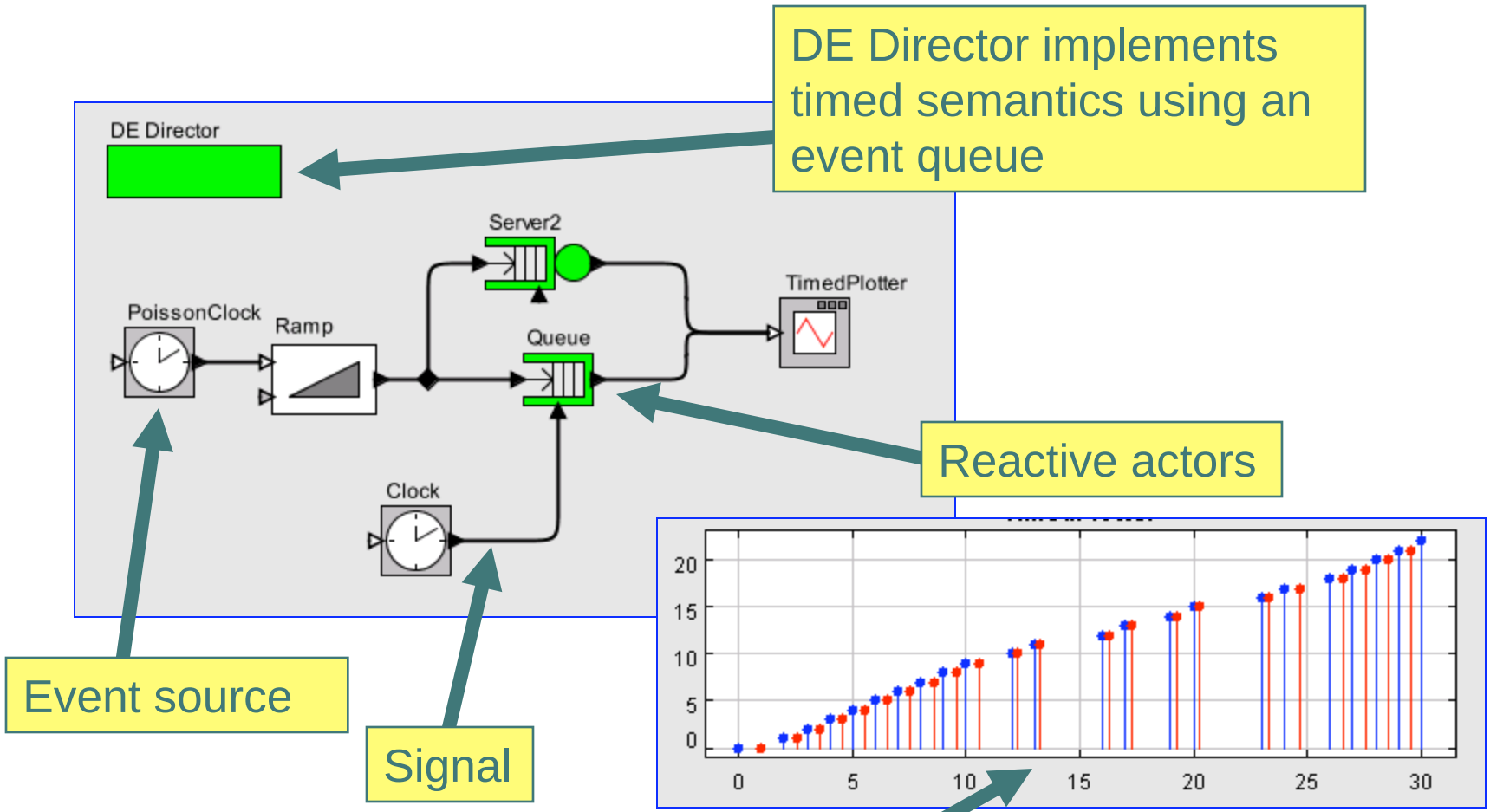


Large, behaviorally-polymorphic component library.

Type system for transported data

Visual editor supporting an abstract syntax

Discrete Event Models in Ptolemy II



Components send time-stamped events to other components, and components react in chronological order.

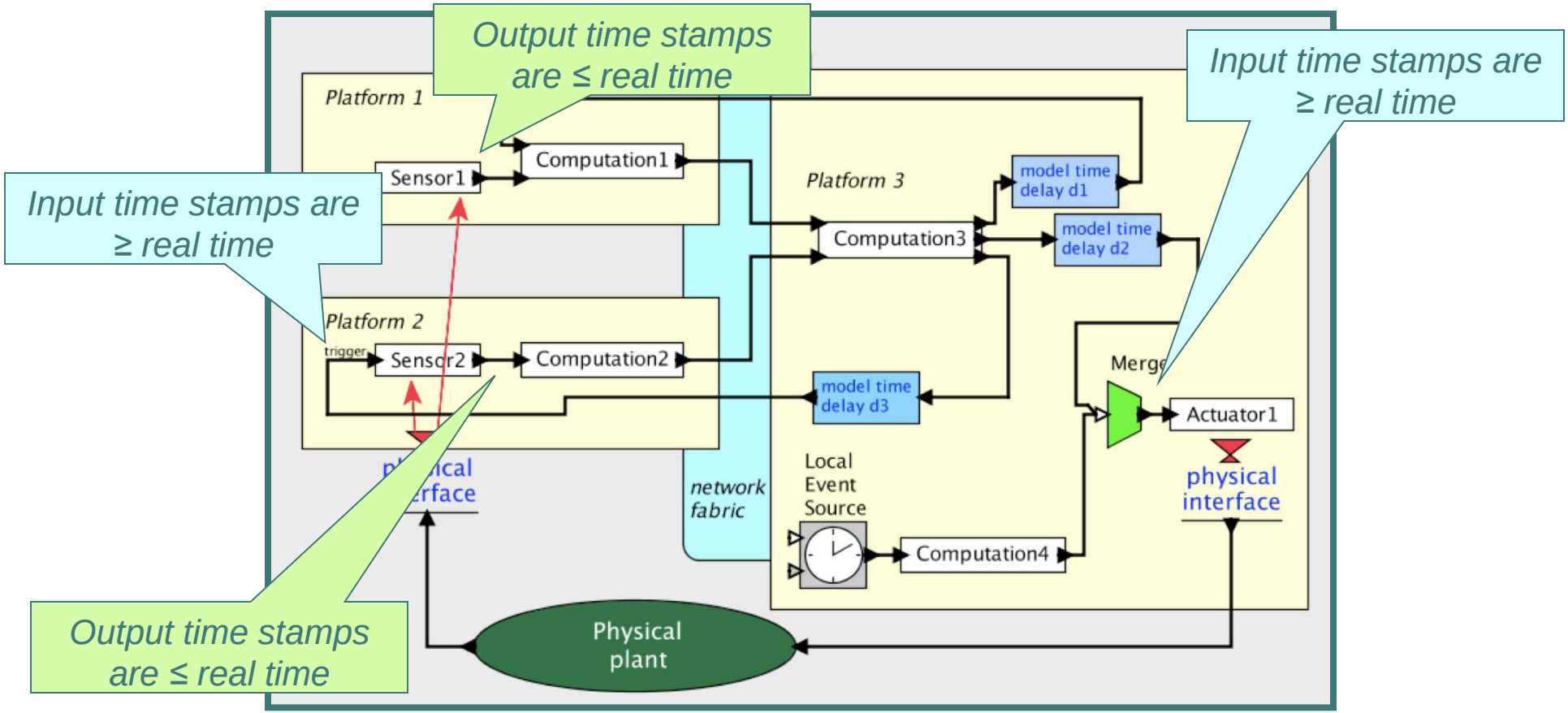
An Application of All This Theory: Using DE Models to Design Distributed Real-Time Systems

- DE is usually a simulation technology.
- Distributing DE is done for acceleration.
- Hardware design languages (e.g. VHDL) use DE where time stamps are literally interpreted as real time, or abstractly as ticks of a physical clock.

- We are using DE for distributed real-time software, binding time stamps to real time only where necessary.
- *PTIDES*: Programming Temporally Integrated Distributed Embedded Systems

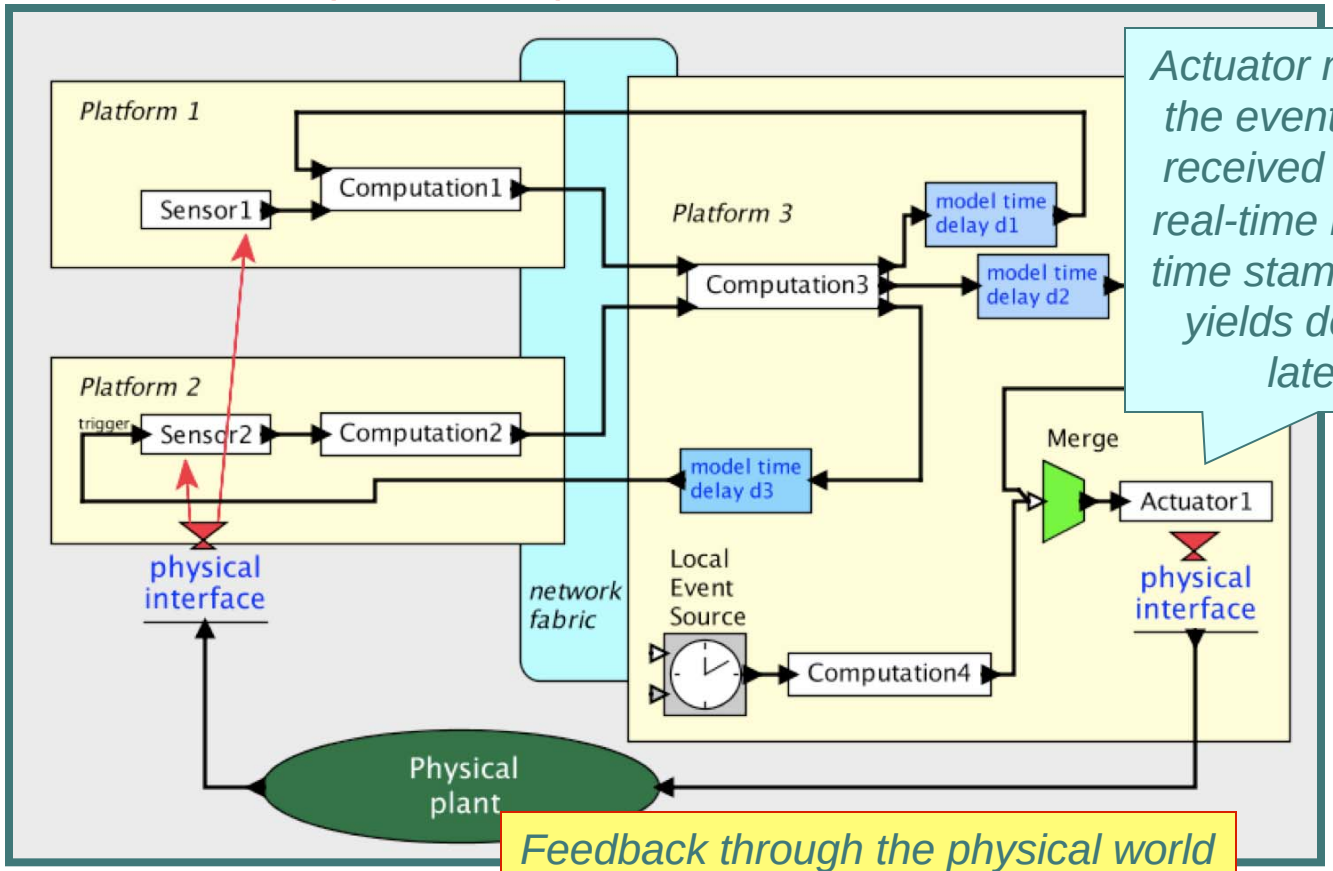
PTIDES: Programming Temporally Integrated Distributed Embedded Systems

Distributed execution under discrete-event semantics, with "model time" and "real time" bound at sensors and actuators.

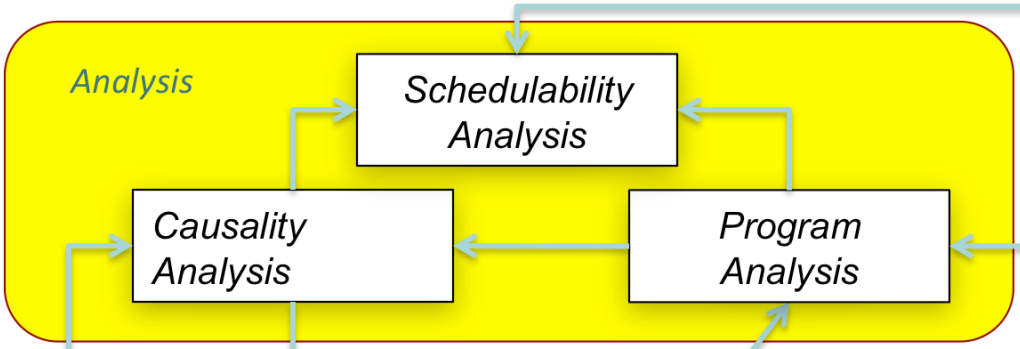


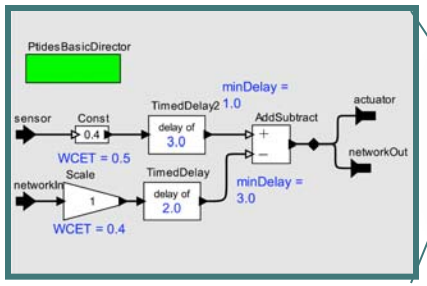
PTIDES: Programming Temporally Integrated Distributed Embedded Systems

… and being explicit about time delays means that we can analyze control system dynamics…



Experimental Setup





Ptolemy II Ptides domain

Ptides Model

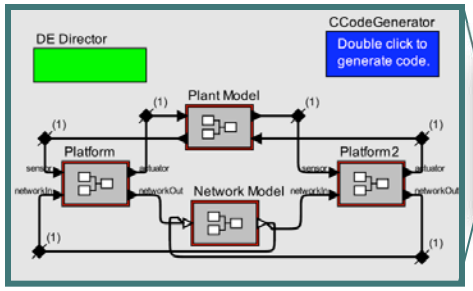
Code Generator

Code

PtidyOS

HW Platform

Software Component Library



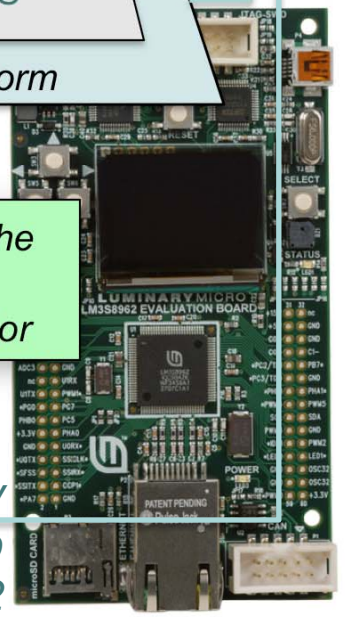
Ptolemy II Discrete-event, Continuous, and Wireless domains

Mixed Simulator

Plant Model

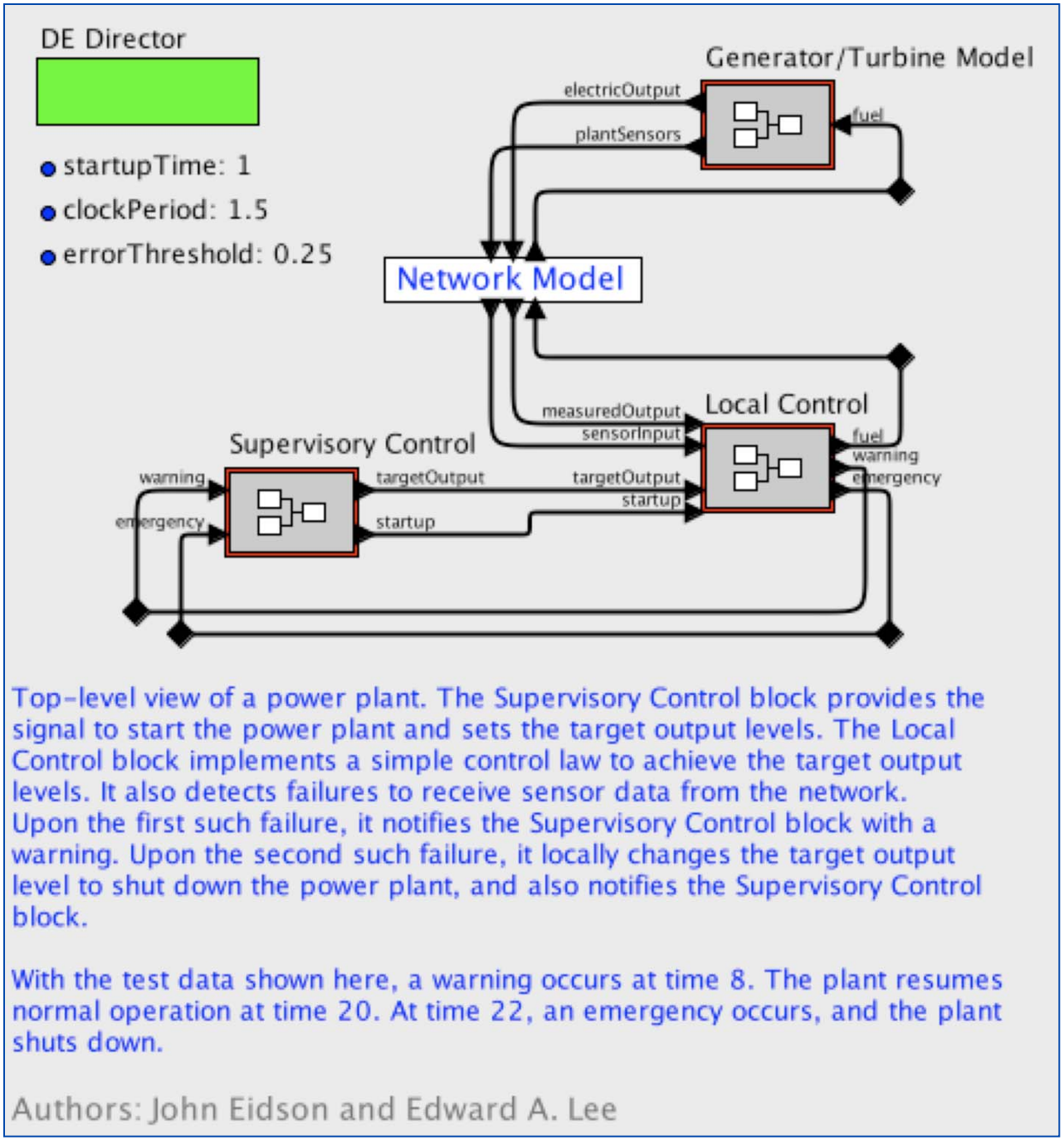
Network Model

HW in the Loop Simulator



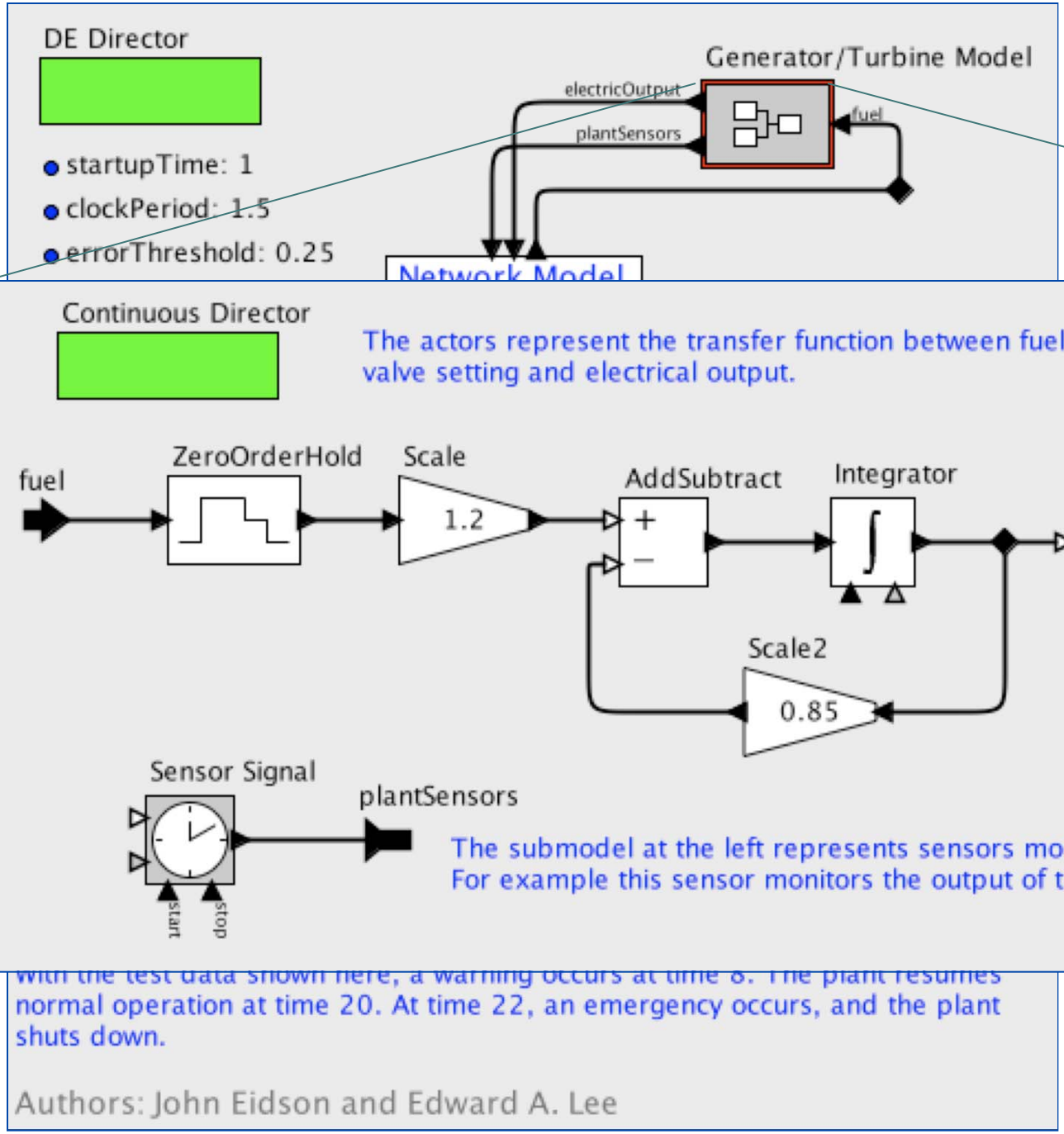
Luminary Micro 8962

IEEE 1588 Network time protocol



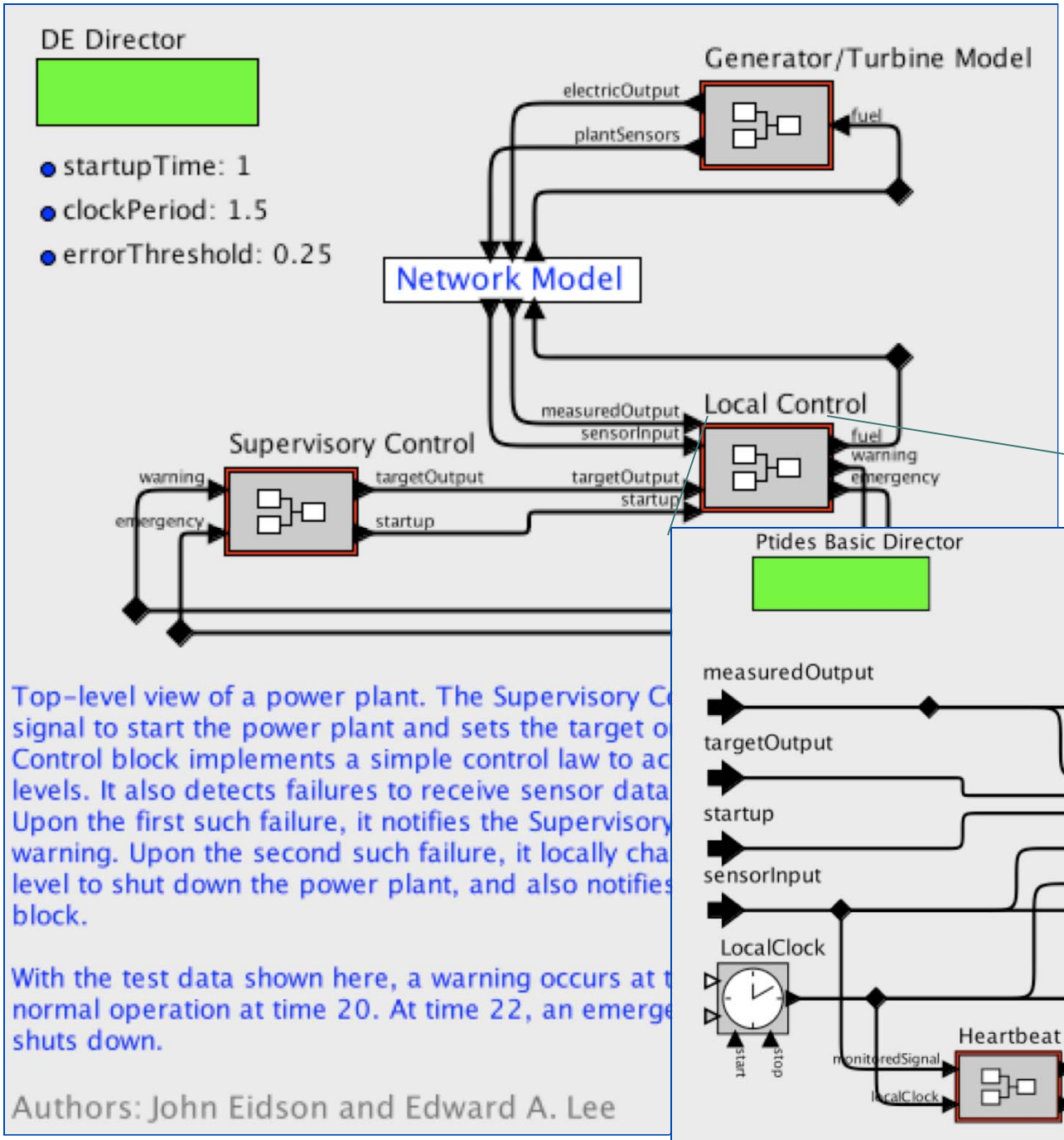
Example: Power Plant Control

Example: Power Plant Control

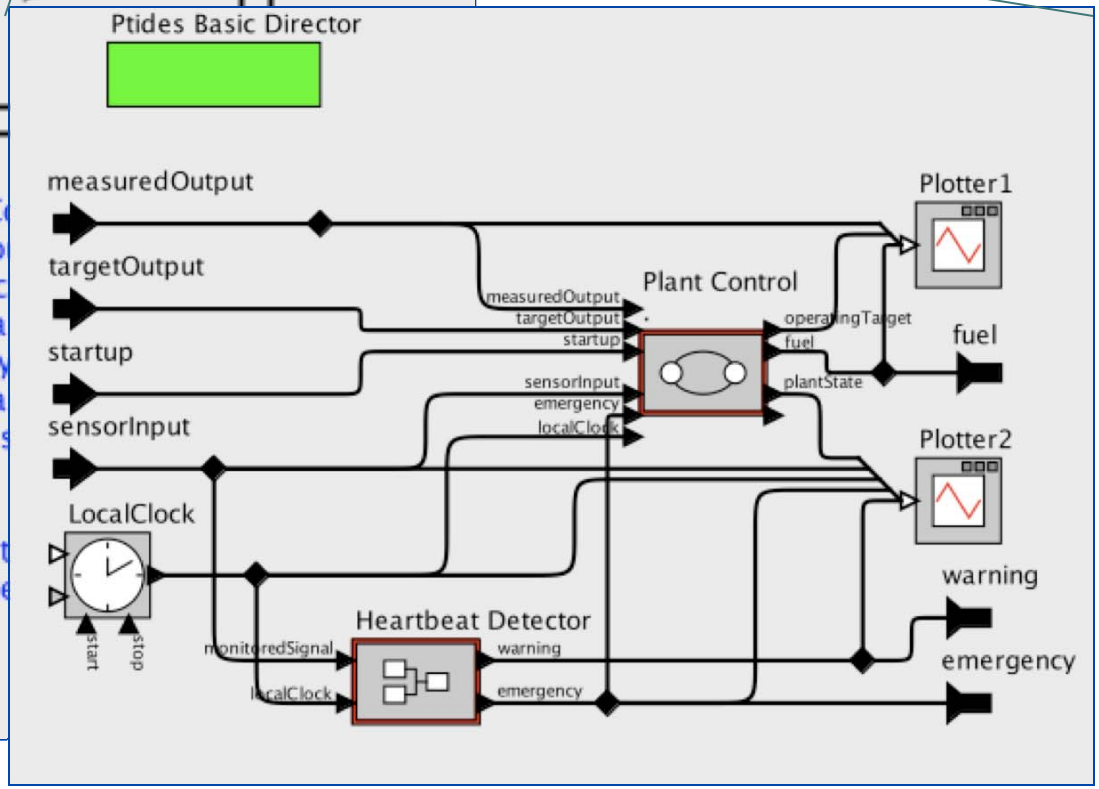


Model of the continuous dynamics of a generator

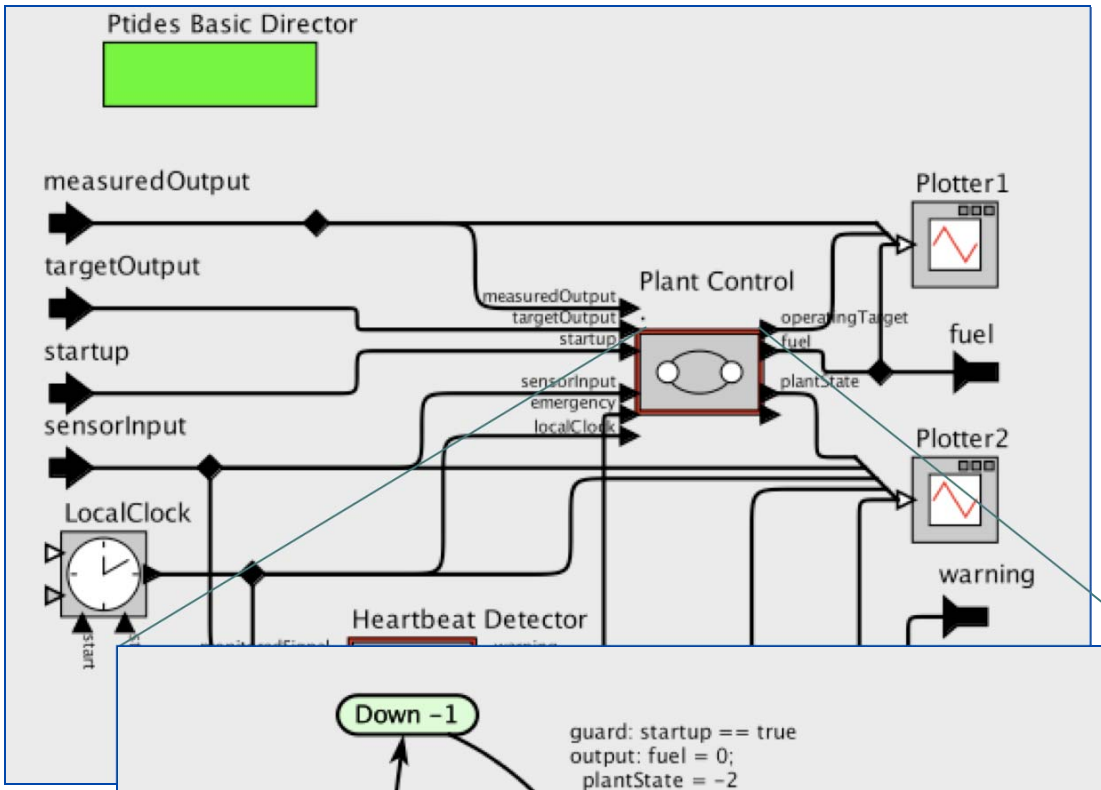
Example: Power Plant Control

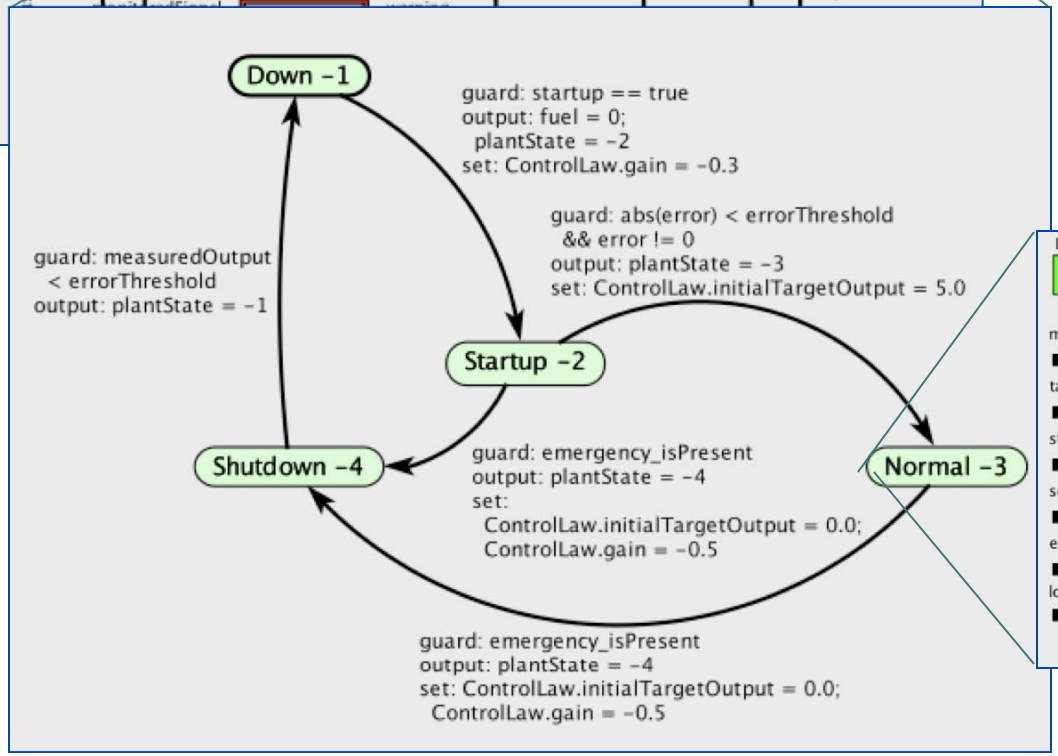


Model of the embedded software controller

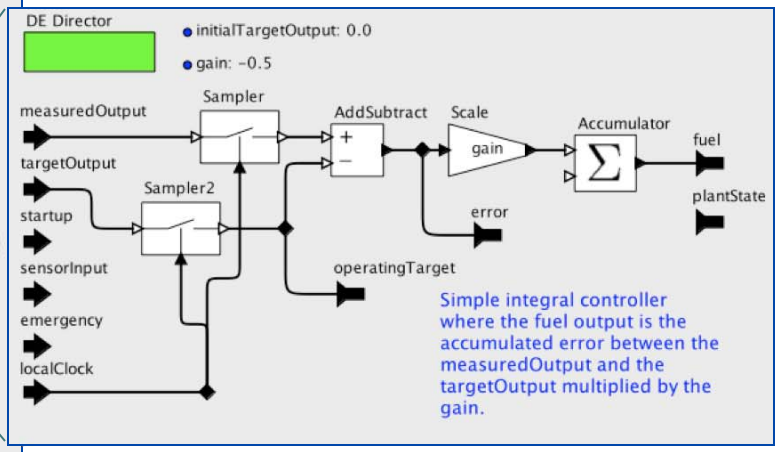


Example: Power Plant Control

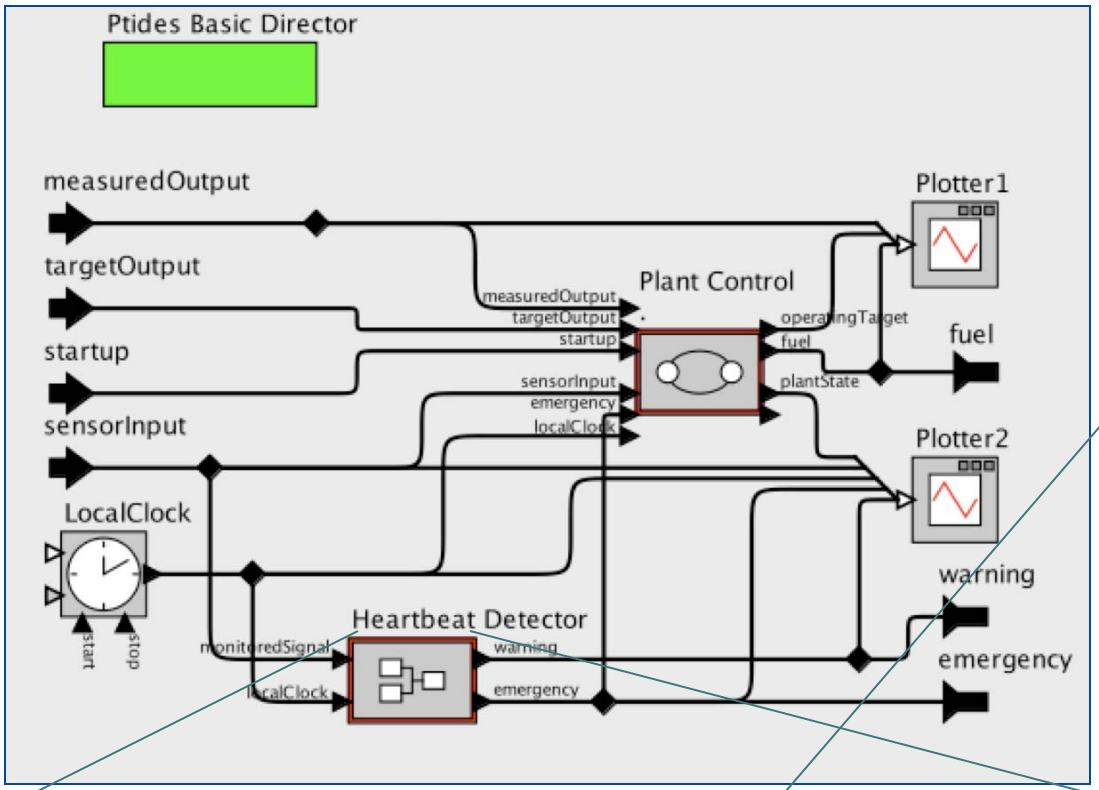


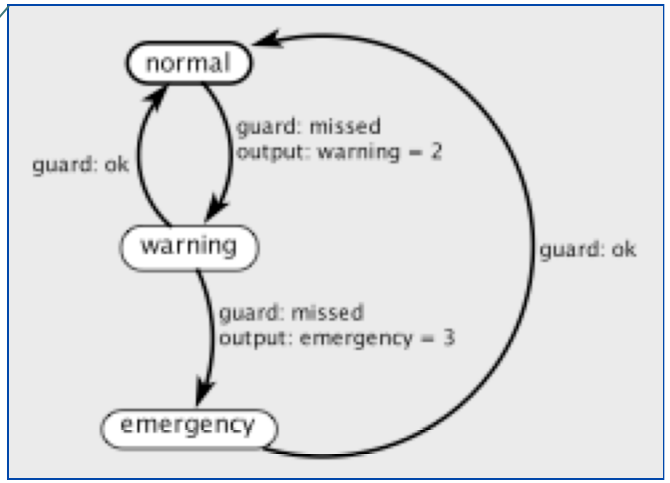


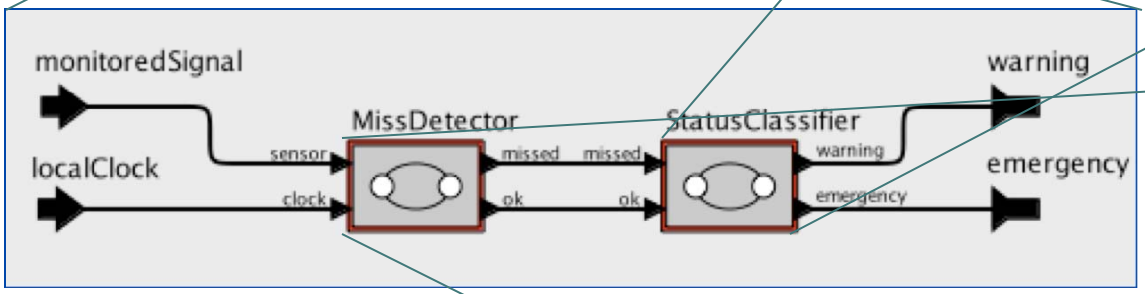
Modal controller behavior.

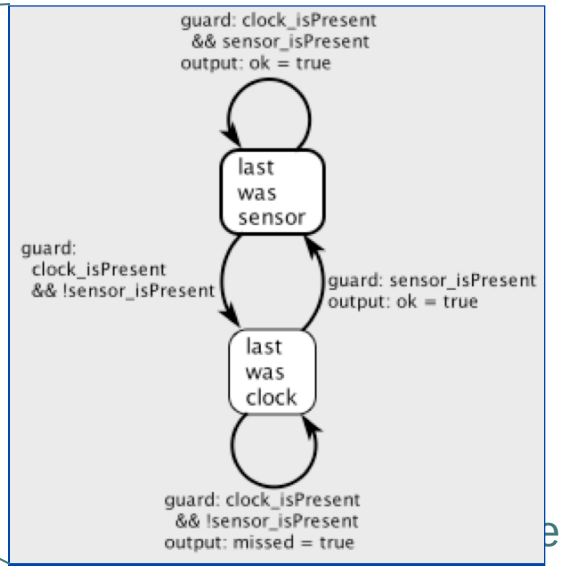


Example: Power Plant Control







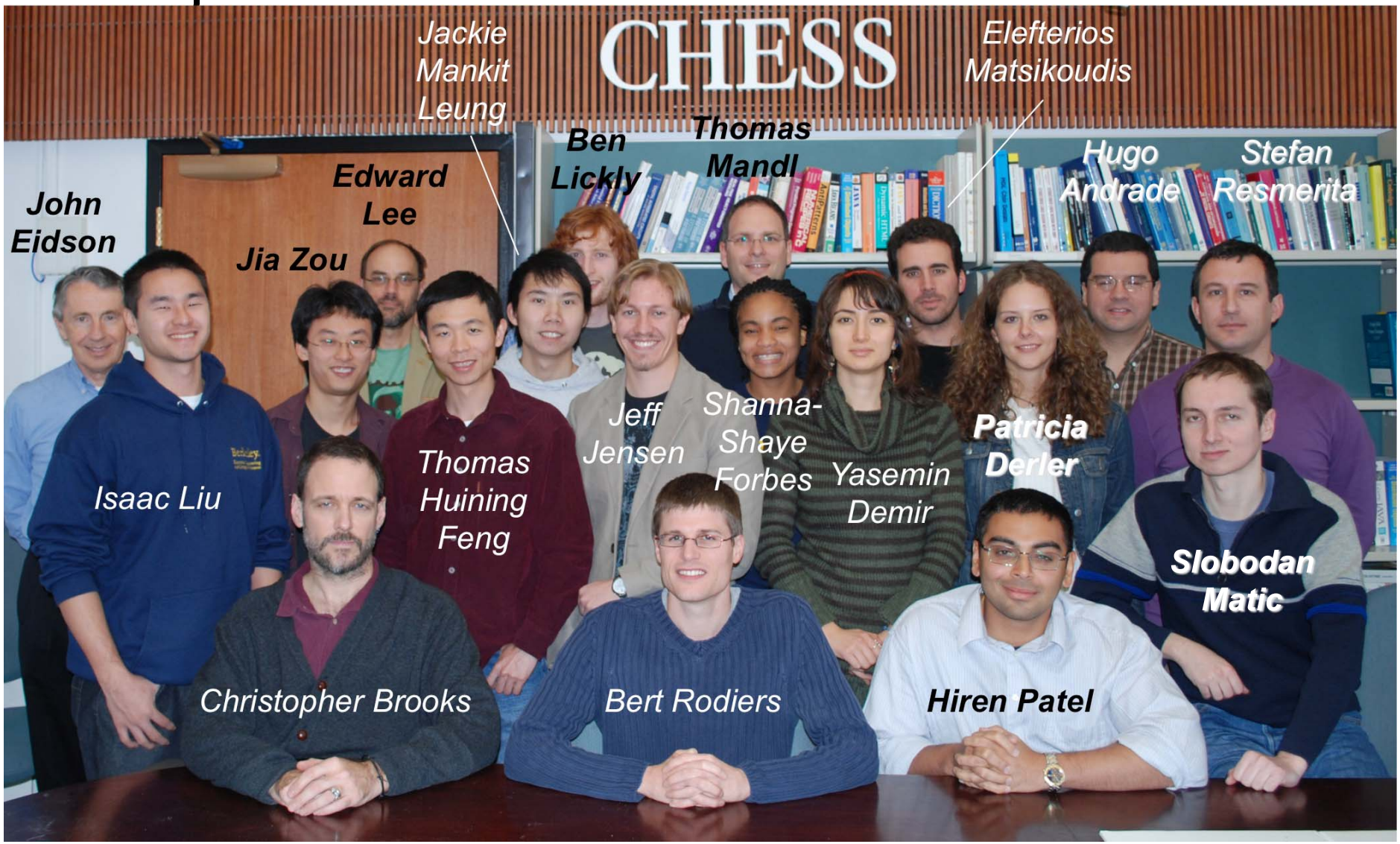


Reasoning about time

Conclusions

- Established models for signals that prevail in signal processing are not expressive enough to model the behavior of nontrivial software and networks.
- *Superdense time* uses tags that have a real-valued *time-stamp* and a natural number *index*, thus supporting sequences of causally-related simultaneous events.
- Discrete-event (DE) systems can provide a foundation for model-based design of nontrivial signal processing systems that integrate software and networks.
- An extension of DE called PTIDES provides a distributed model coupling software and physical behaviors.

The Ptolemy Pteam



A Few References

http://ptolemy.eecs.berkeley.edu

Ph.D. Theses:

[1] *On the Design of Concurrent, Distributed Real-Time Systems*, Yang Zhao [2009]

[2] *Operational Semantics of Hybrid Systems*, Haiyang Zheng [2007]

[3] *Interface Theories for Causality Analysis in Actor Networks*, Ye Zhou [2007]

[4] *Semantic Foundation of the Tagged Signal Model*, Xiaojun Liu, [2005]

[5] *Responsible Frameworks for Heterogeneous Modeling and Design of Embedded Systems*, Jie Liu [2001]

Papers:

[1] E. A. Lee, "Computing Needs Time," Communications of the ACM, 52(5), May 2009.

[2] X. Liu, E.A. Lee, "CPO semantics of timed interactive actor networks," Theoretical Computer Science 409 (1): pp.110-25, 2008..

[3] Zhou and Lee. "Causality Interfaces for Actor Networks," ACM Trans. on Embedded Computing Systems, April 2008.

[4] Lee, "Application of Partial Orders to Timed Concurrent Systems," article in *Partial order techniques for the analysis and synthesis of hybrid and embedded systems*, in CDC 07.

[5] Lee and Zheng, "Leveraging Synchronous Language Principles for Heterogeneous Modeling and Design of Embedded Systems," EMSOFT '07.

[6] Liu, Matsikoudis, and Lee. "Modeling Timed Concurrent Systems," CONCUR '06.

[7] Cataldo, Lee, Liu, Matsikoudis and Zheng, "A Constructive Fixed-Point Theorem and the Feedback Semantics of Timed Systems," WODES'06

[8] Lee and Zheng, "Operational Semantics of Hybrid Systems," HSCC '05.

etc. ...