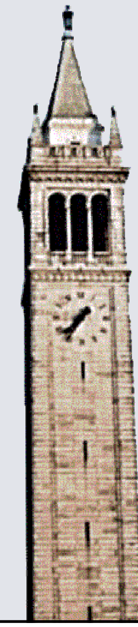
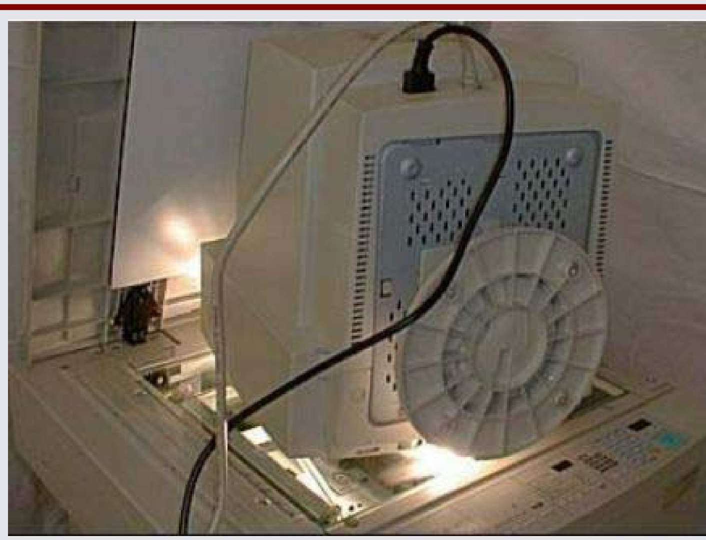
# Ptolemy Project Plans for the Future

Edward A. Lee
Professor
Ptolemy Project Director



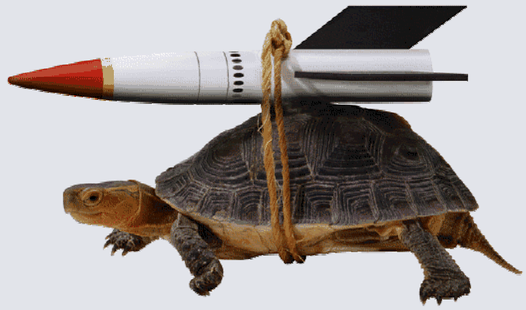
Ptolemy Miniconference
May 9, 2003
Berkeley, CA

## Composition of Components



Poor choices of interfaces lead to very awkward architectures.

# What about "real time"?



The standard model: Make it faster!

We will integrate time into computation abstractions.



# Real-Time Multitasking?

The standard model: Prioritize and Pray!

With time in the abstractions, we can get predictable deployed software.

## Chess: Center for Hybrid and Embedded Software Systems

Chess and its NSF/ITR project are a major part of our future.
The project is *Foundations of Hybrid and Embedded Software Systems.*

**Chess Board of Directors**

- Tom Henzinger, tah@eecs.berkeley.edu
- Edward A. Lee, eal@eecs.berkeley.edu
- Alberto Sangiovanni-Vincentelli, alberto@eecs.berkeley.edu
- Shankar Sastry, sastry@eecs.berkeley.edu

**Other key faculty**

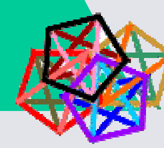
- Alex Aiken, aiken@eecs.berkeley.edu
- Dave Auslander, dma@me.berkeley.edu
- Ruzena Bajcsy, bajcsy@eecs.berkeley.edu
- Karl Hedrick, khedrick@me.berkeley.edu
- Kurt Keutzer, keutzer@eecs.berkeley.edu
- George Necula, necula@eecs.berkeley.edu
- Masayoshi Tomizuka, tomizuka@me.berkeley.edu
- Pravin Varaiya, varaiya@eecs.berkeley.edu

Chess

NSF

VANDERBILT UNIVERSITY

INSTITUTE FOR SOFTWARE INTEGRATED SYSTEMS





## Mission of Chess

To provide an environment for graduate research on the design issues necessary for supporting next-generation embedded software systems.

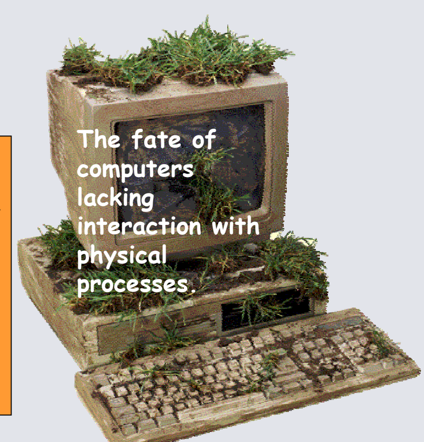
- Model-based design
- Tool-supported methodologies

For

- Real-time
- Fault-tolerant
- Robust
- Secure
- Heterogeneous
- Distributed

Software

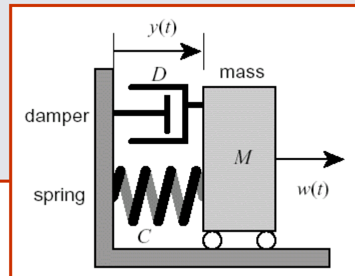
We are on the line to create a "new systems science" that is at once computational and physical.

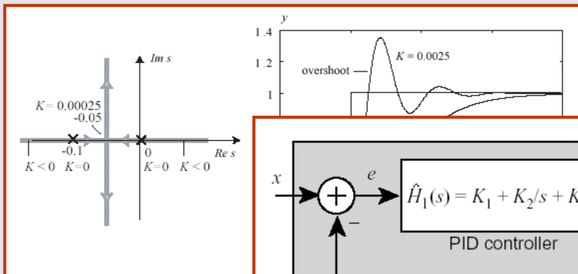


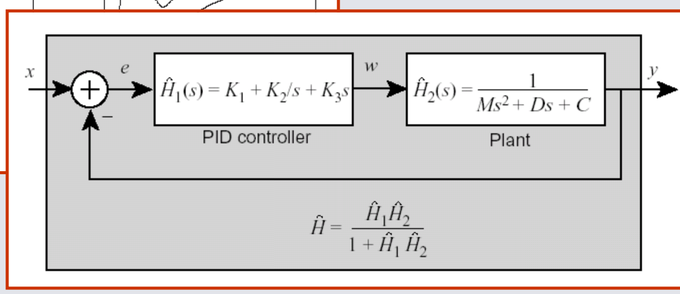
# A Traditional Systems Science – Feedback Control Systems

- Models of continuous-time dynamics
- Sophisticated stability analysis

- But not accurate for software controllers



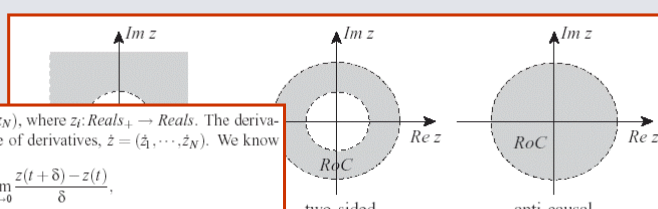






# Discretized Model – A Step Towards Software

- 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 $z_i\colon Reals_+ \to Reals$. 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

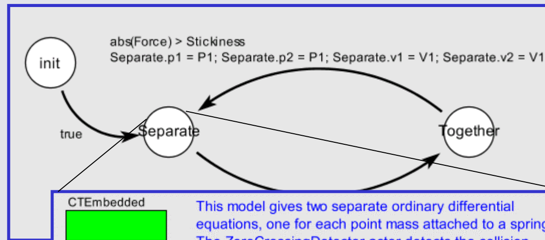
$$\dot{z}(t) \approx \frac{z(t+\delta) - z(t)}{\delta}.$$

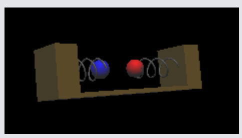
Using this for the derivative in the left-hand side of (5.50) we get

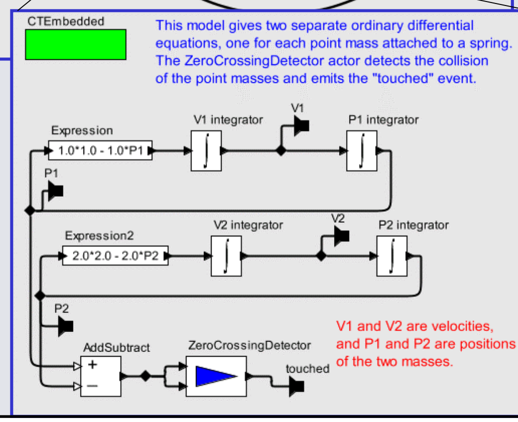
$$z(t+\delta) - z(t) = \delta g(z(t), v(t)). \qquad (5.51)$$

# Hybrid Systems – Reconciliation of Continuous & Discrete

- UCB researchers have contributed hugely to the theory and practice of blended discrete & continuous models.
- But it's *still* not accurate for software controllers



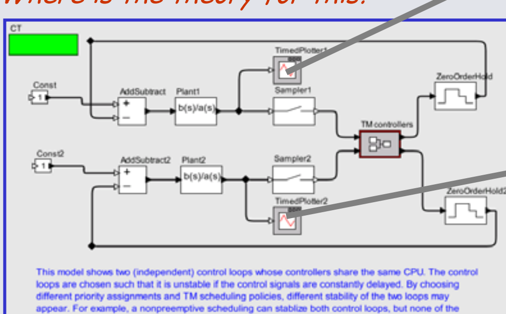



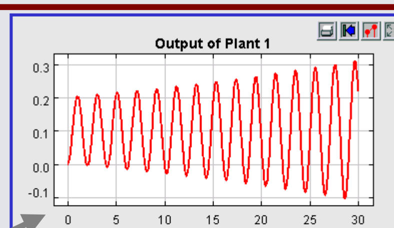


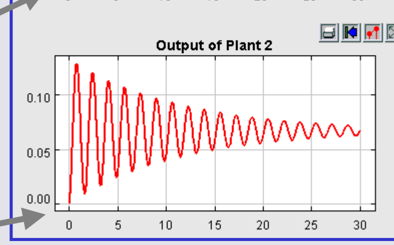

# Timing in Software is More Complex Than What the Theory Deals With

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?*



## How Safe is Our Real-Time Software?

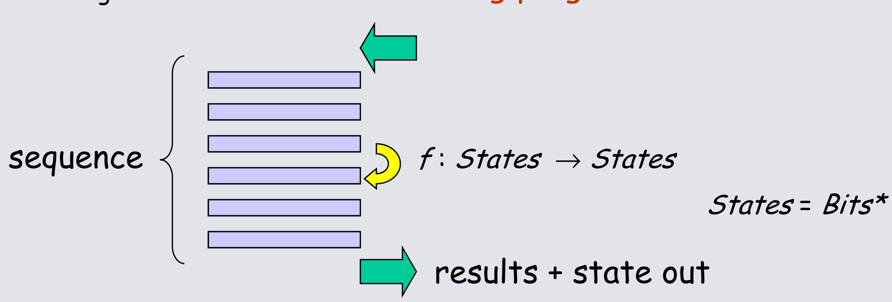


## Another Traditional Systems Science - Computation, Languages, and Semantics

Alan Turing

Everything "computable" can be given by a terminating sequential program.

- Functions on bit patterns
- Time is irrelevant
- Non-terminating programs are defective



## Current fashion – Pay Attention to "Non-functional properties"

- Time
- Security
- Fault tolerance
- Power consumption
- Memory management

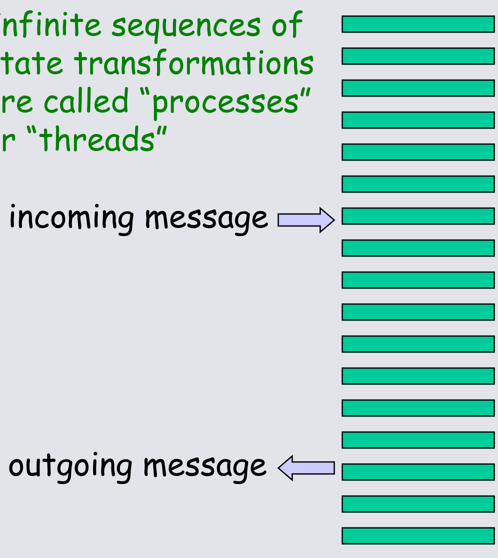
But the formulation of the question is very telling:

How is it that *when* a braking system applies the brakes is any less a *function* of the braking system than *how much* braking it applies?



## Processes and Process Calculi

Infinite sequences of state transformations are called "processes" or "threads"



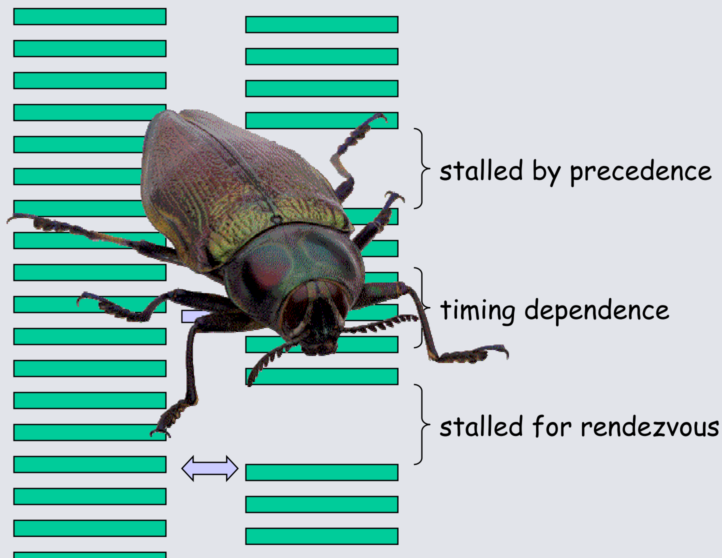

Various messaging protocols lead to various formalisms.

In prevailing software practice, processes are sequences of external interactions (total orders).

And messaging protocols are combined in ad hoc ways.

## Interacting Processes – Concurrency as Afterthought

Software realizing these interactions is written at a very low level (semaphores and mutexes). *Very* hard to get it right.

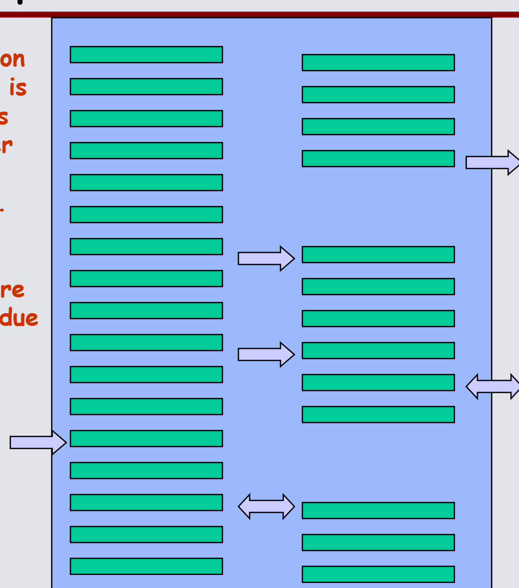





## Interacting Processes – Not Compositional

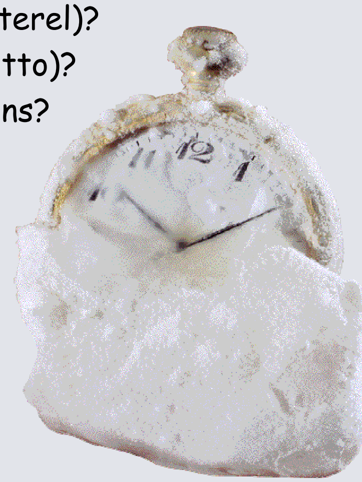
**An aggregation of processes is not a process (a total order of external interactions). What is it?**

**Many software failures are due to this ill-defined composition.**



## What Will Replace This Approach?

- Synchronous languages (e.g. Esterel)?
- Time-driven languages (e.g. Giotto)?
- Push/Pull component interactions?
- Hybrid systems?
- Timed process networks?
- Discrete-event formalisms?
- Timed CSP?



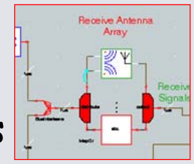

We intend to find out.



## What are we doing for Foundations?

- Hybrid systems semantics
  - with a focus on execution (vs. verification)
- Interface definitions and checkers
  - collaboration with Prof. Henzinger's group
- Reflection of behavioral types
  - admission control
  - run-time type checking
  - fault detection, isolation, and recovery (FDIR)
- Timed behavioral types
  - checking consistency becomes a type check
  - generalized schedulability analysis
- Semantic unification of push/pull & dataflow

## What are we doing for Software?




- Actor definition
- Higher-order & mobile components
- Distributed & web-integrated models
  - peer-to-peer technologies integrated
- Component specialization framework
- Runtime environments
- Many software capabilities envisioned:
  - 2-D graphics animation
  - string library
  - communications library
  - video & image processing library
  - …



## Conclusion

Stay tuned.