

Heterogeneous Modeling and Design DARPA Composite CAD PI Meeting



UC Berkeley, EECS Dept.

Staff

Jennifer Basler
Christopher Hylands
Edward A. Lee, PI
Mary P. Stewart

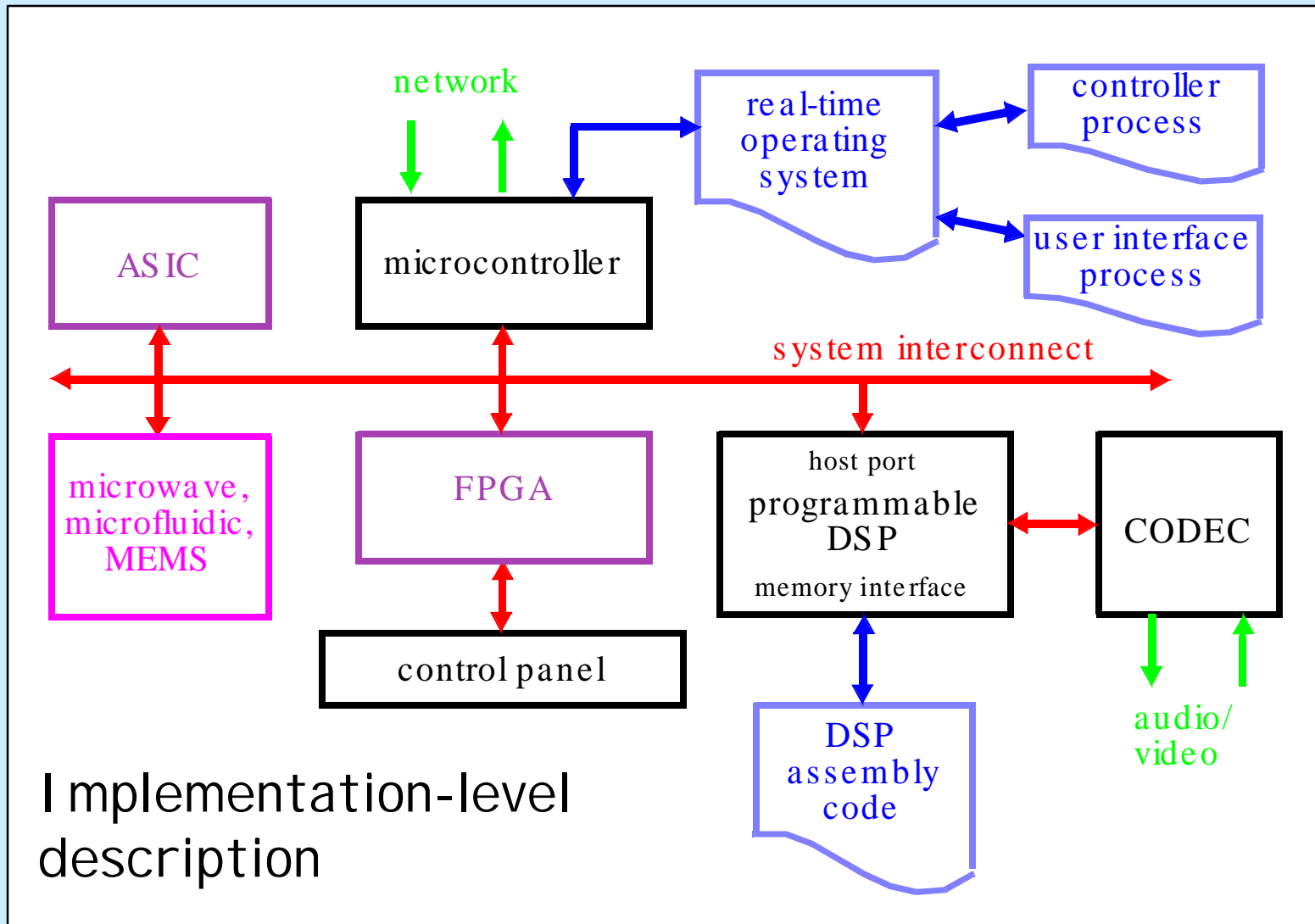
Postdoctoral Researchers

H. John Reekie

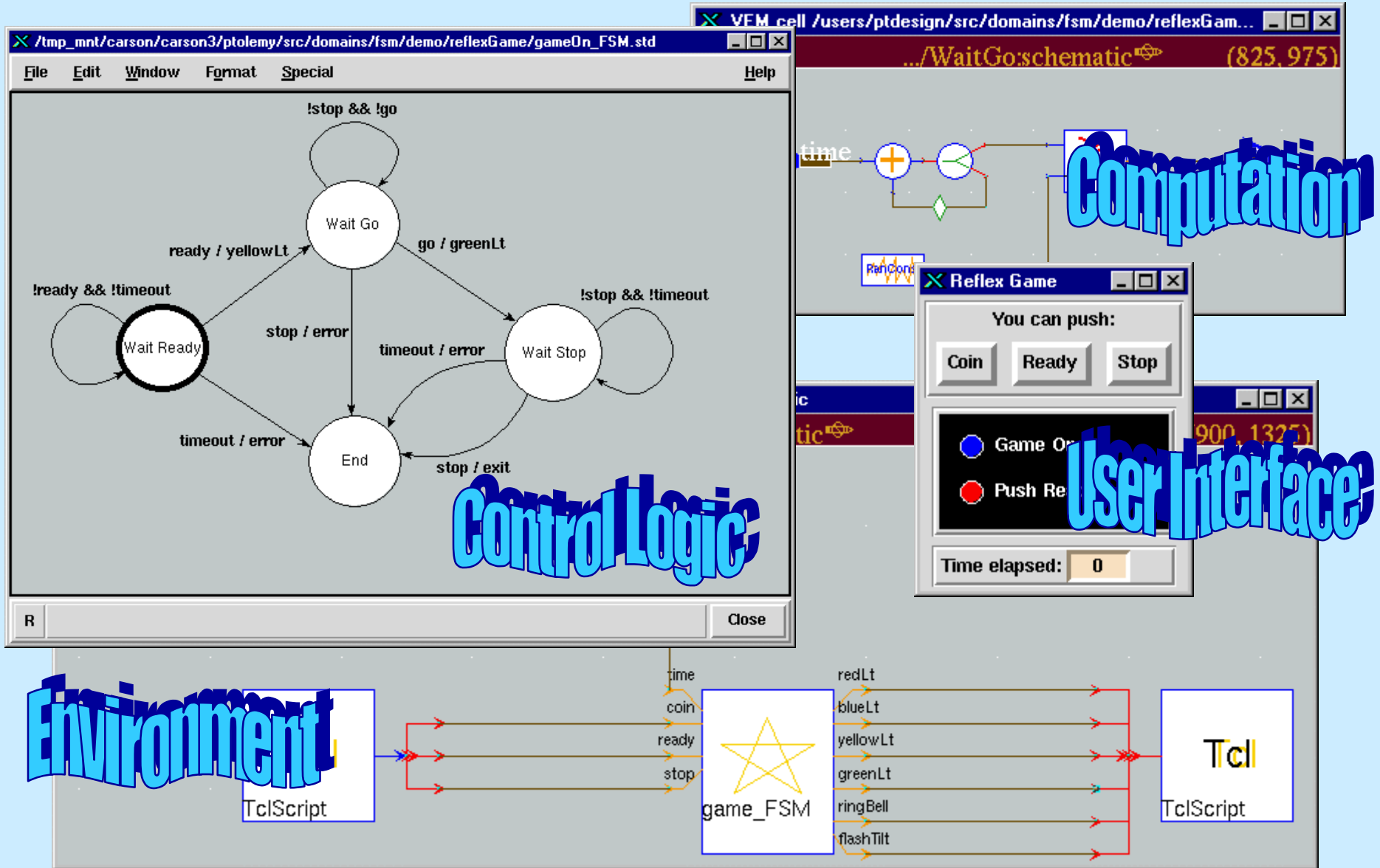
Students

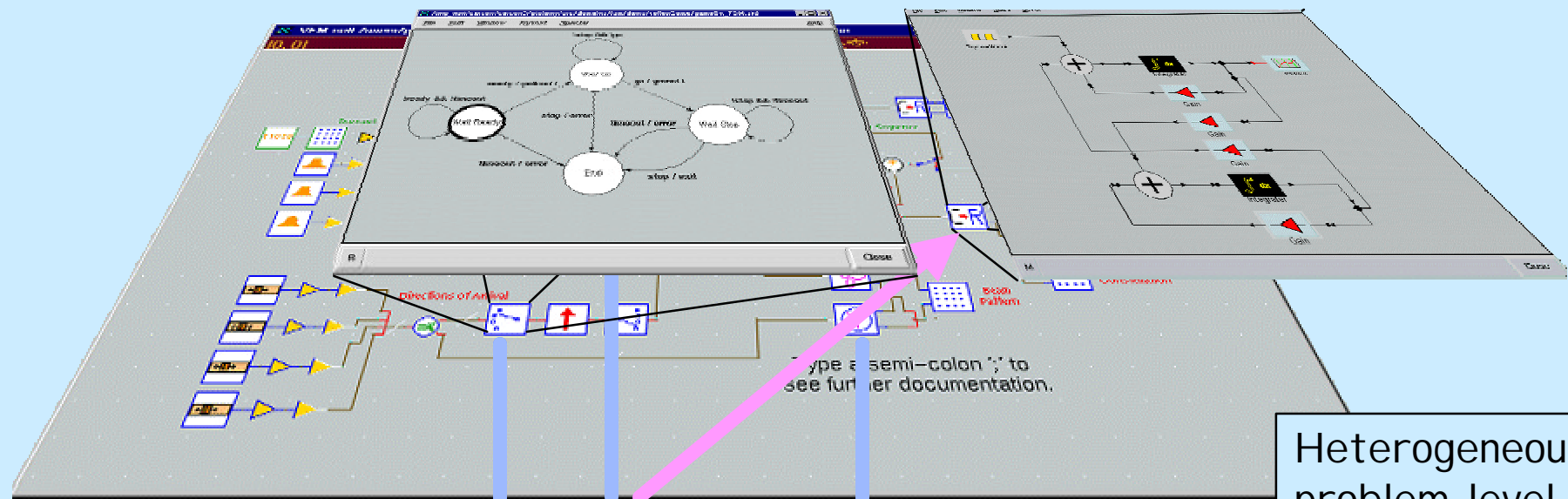
Albert Chen
John Davis, II
Mudit Goel
Bilung Lee
Jie Liu
Xiaojun Liu
Stephen Neuendorffer
Neil Smyth
William Wu
Yuhong Xiong

A Typical Embedded System

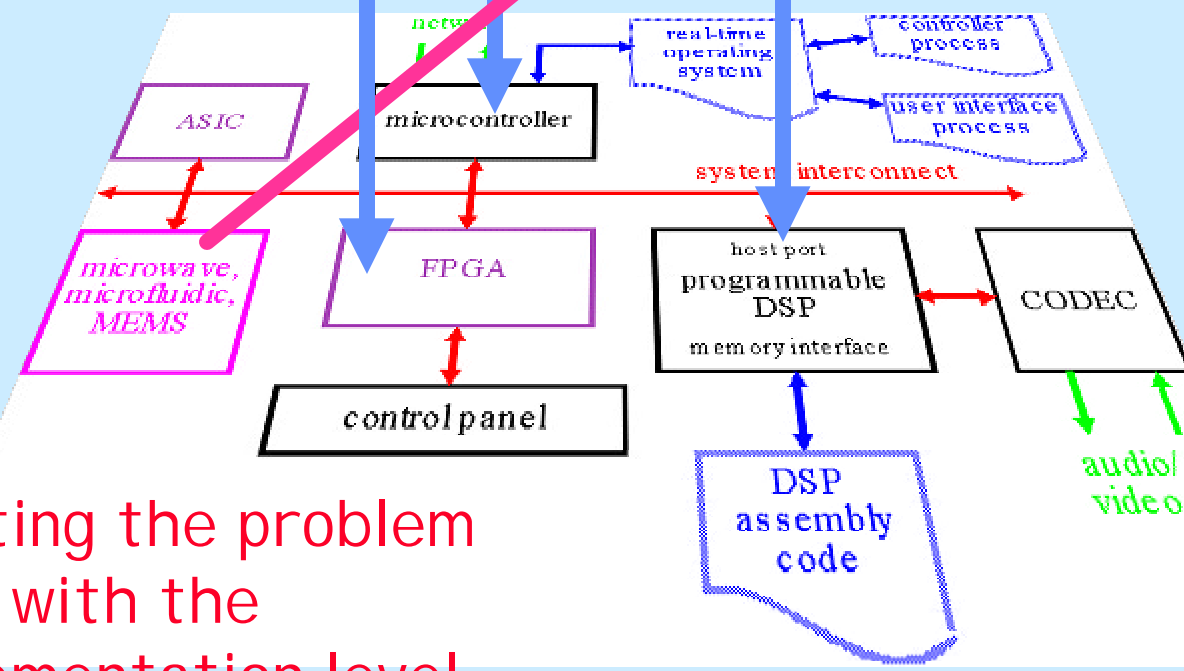


Conceptual (Problem-level) Modeling





Heterogeneous, problem-level description



Heterogeneous, implementation-level description

Modeling ↑
Synthesis ↓

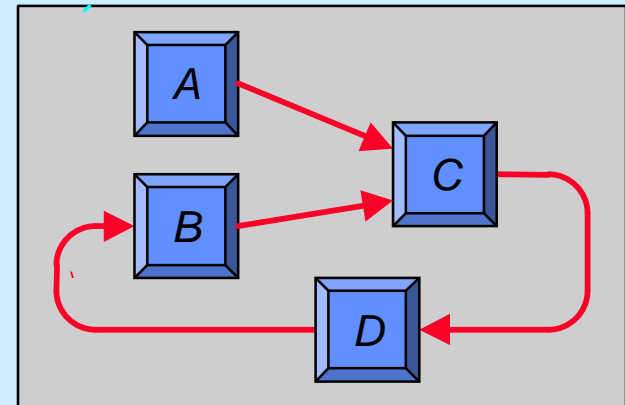
Relating the problem level with the implementation level

Approach

- Theory and techniques for mixing diverse models of computation, e.g. mixed signal, hybrid systems, discrete and continuous events.
- Software architecture for modular, distributed, and heterogeneous design, modeling and visualization tools.
- Theory and software for domain-specific modeling of composite concurrent systems.
- Use of programming language concepts (semantics, type theories, and concurrency theories) for modeling and design of composite systems.
- Emphasis on visual representations.

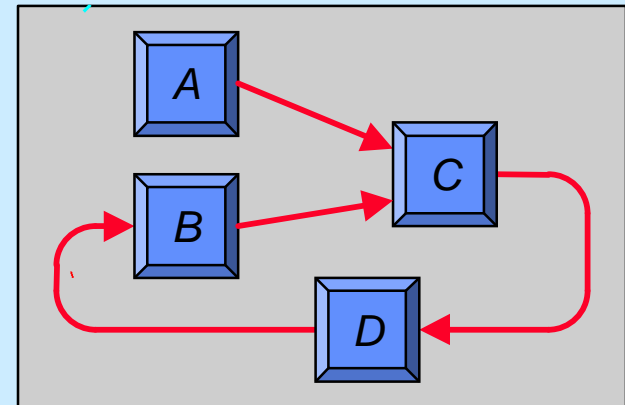
Models of Computation

- Analog computers (differential equations)
- Discrete time (difference equations)
- Discrete-event systems
- Synchronous-reactive systems
- Sequential processes with rendezvous
- Process networks
- Dataflow
- Finite state machines



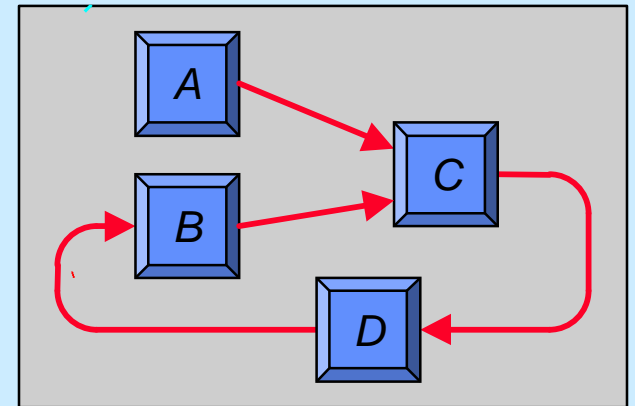
Shared Properties

- Strengths and weaknesses (no silver bullet)
- Domain-specific
- Modular
- Amenable to visual syntaxes
- Hierarchical
- Concurrent (except FSMs)
- Abstract



Issues Being Addressed

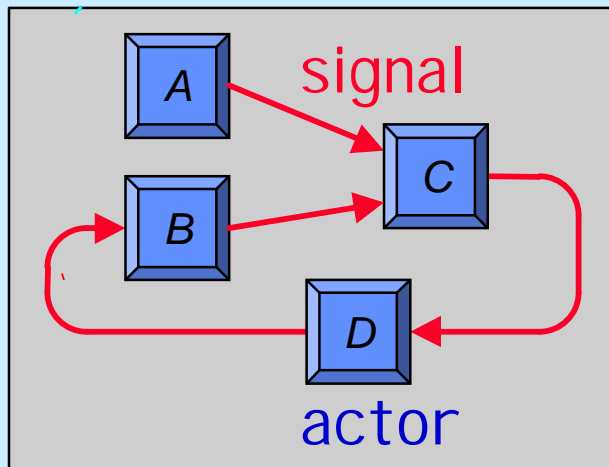
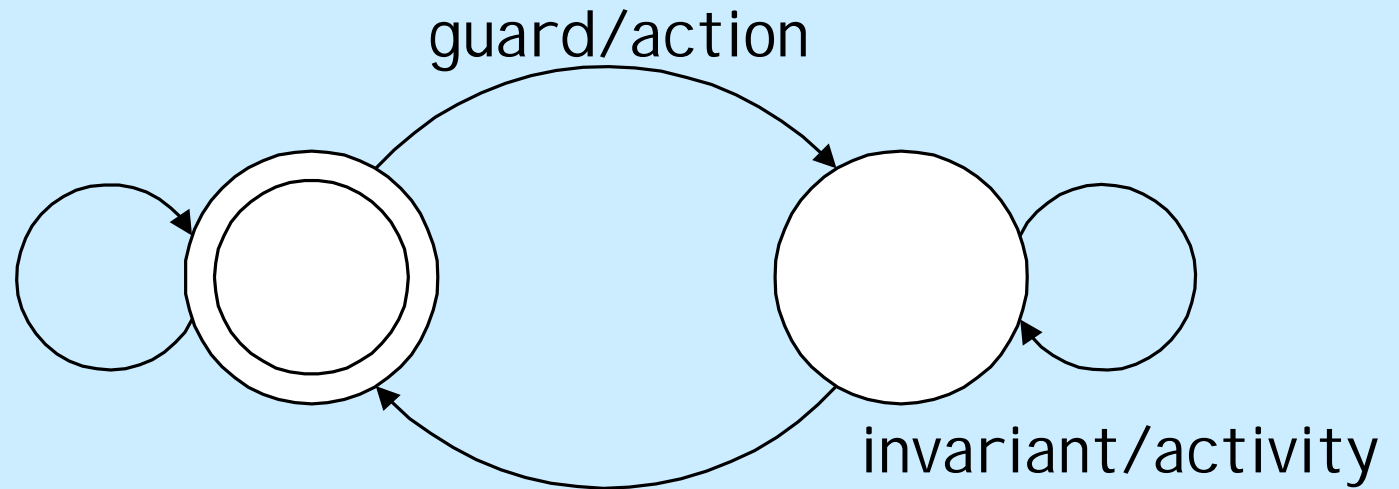
- Semantics (what is a behavior)
- Determinacy (how many behaviors are there)
- Simulation (finding a behavior)
- Analysis (finding properties of behaviors)
- Compositionality (encapsulating subsystems)
- Synthesis (translation to implementation)
- Design (choosing implementations)
- Heterogeneity



Examples Requiring Heterogeneity

- MEMS device with a discrete controller (differential equations plus discrete-event models)
- Modal models, with regimes of operation (differential equations plus finite-state machines)
- Mixed signal systems (differential equations plus discrete-time and/or discrete-event systems)
- Hardware/software systems (differential equations, discrete-events, discrete-time, finite-state machines, dataflow, rendezvous, process networks, ...)

State Machines & Block Diagrams

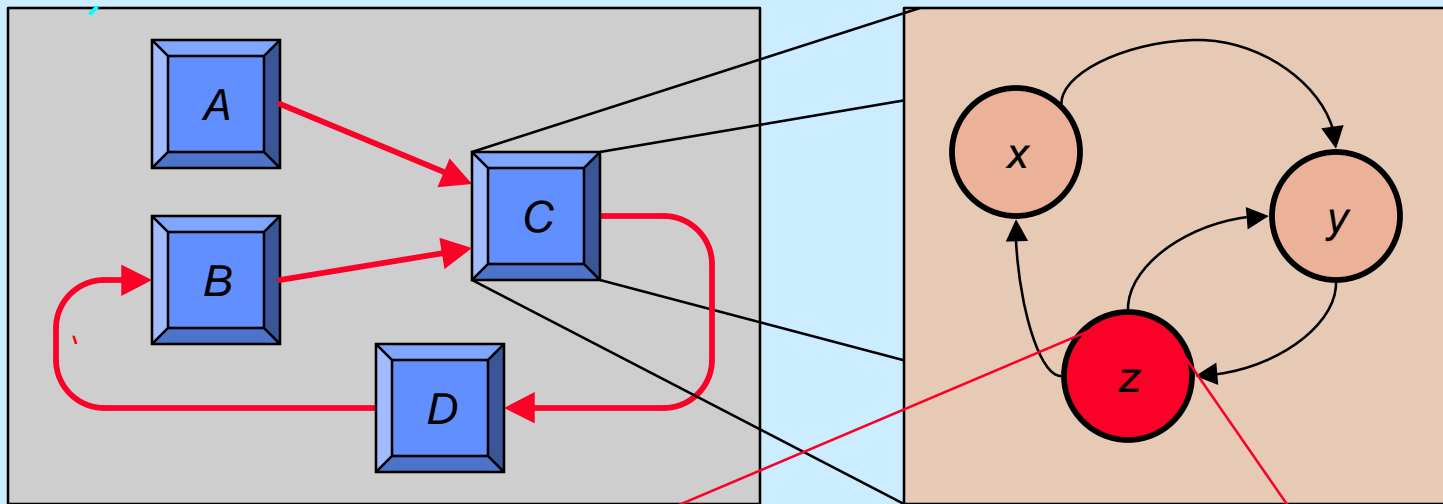


Sequential

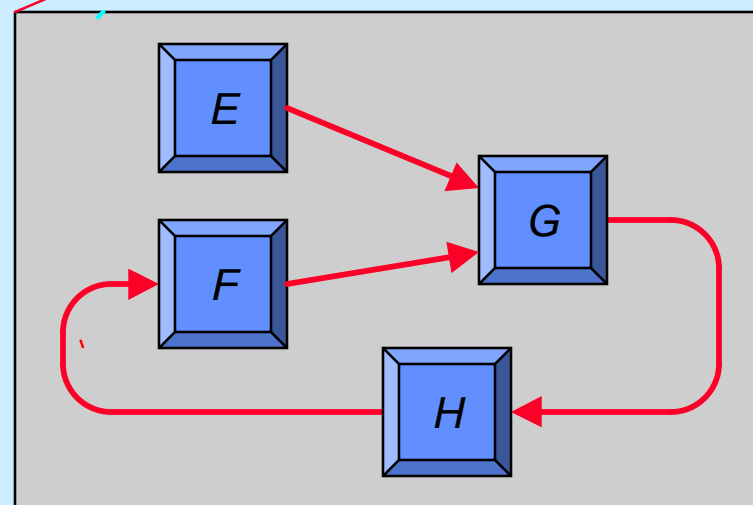
Concurrent

Generalized Hybrid Systems

Choice of domain here determines concurrent semantics

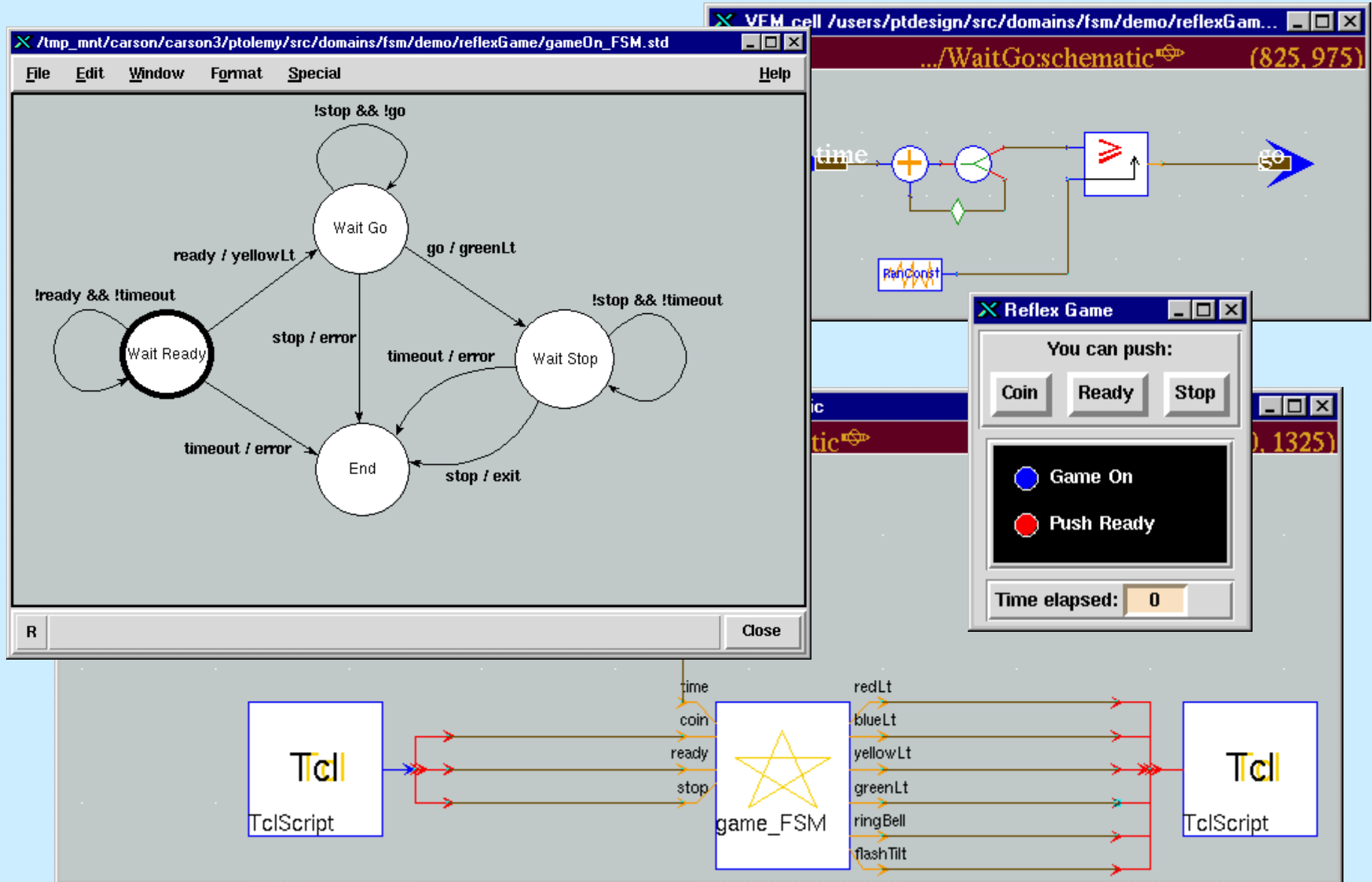


We have formalized the semantics of FSMs combined with discrete-event, dataflow, and synchronous-reactive models.

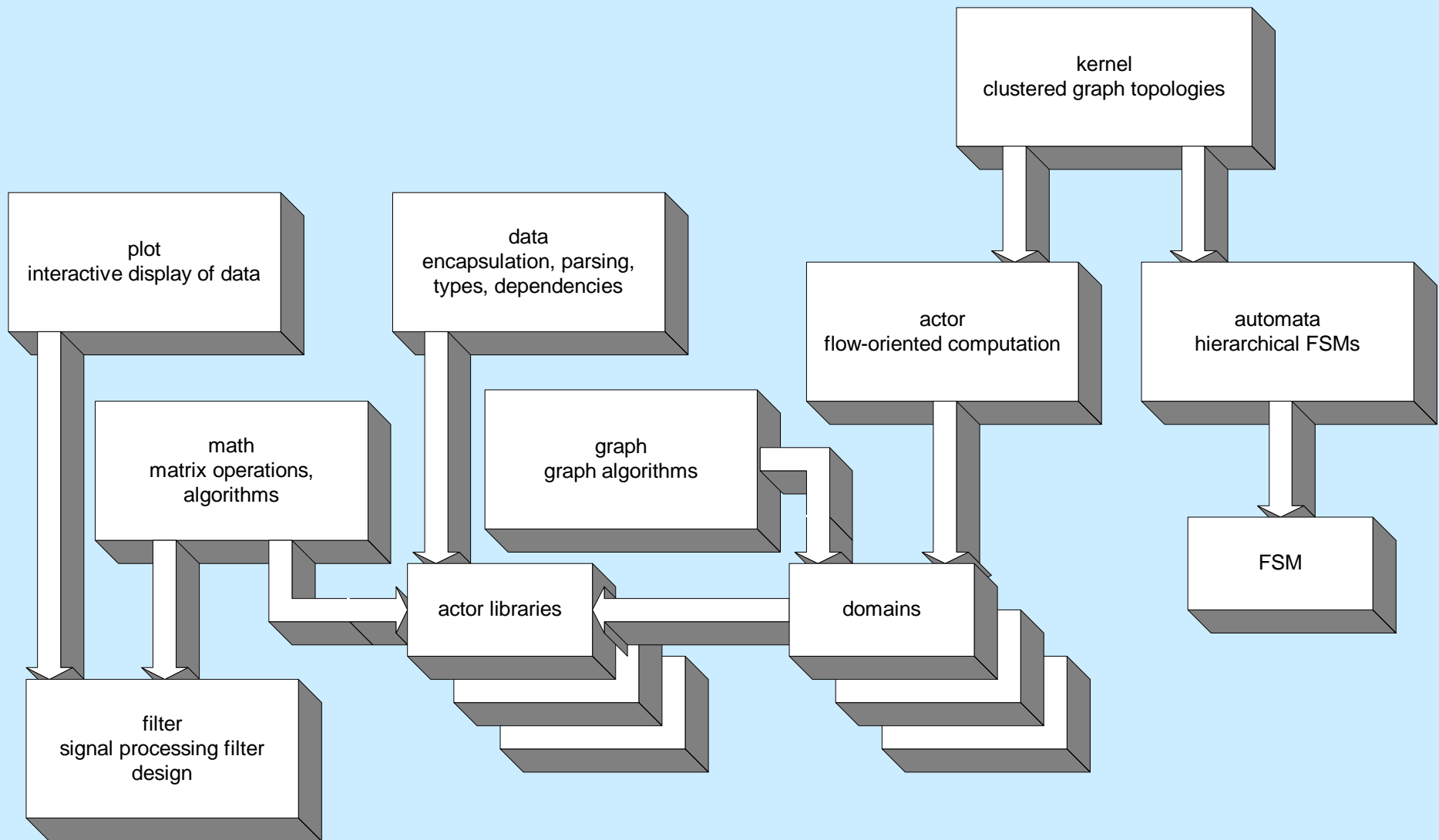


FSM for control

Ptolemy 0.7 Prototype



Package Structure of Ptolemy II

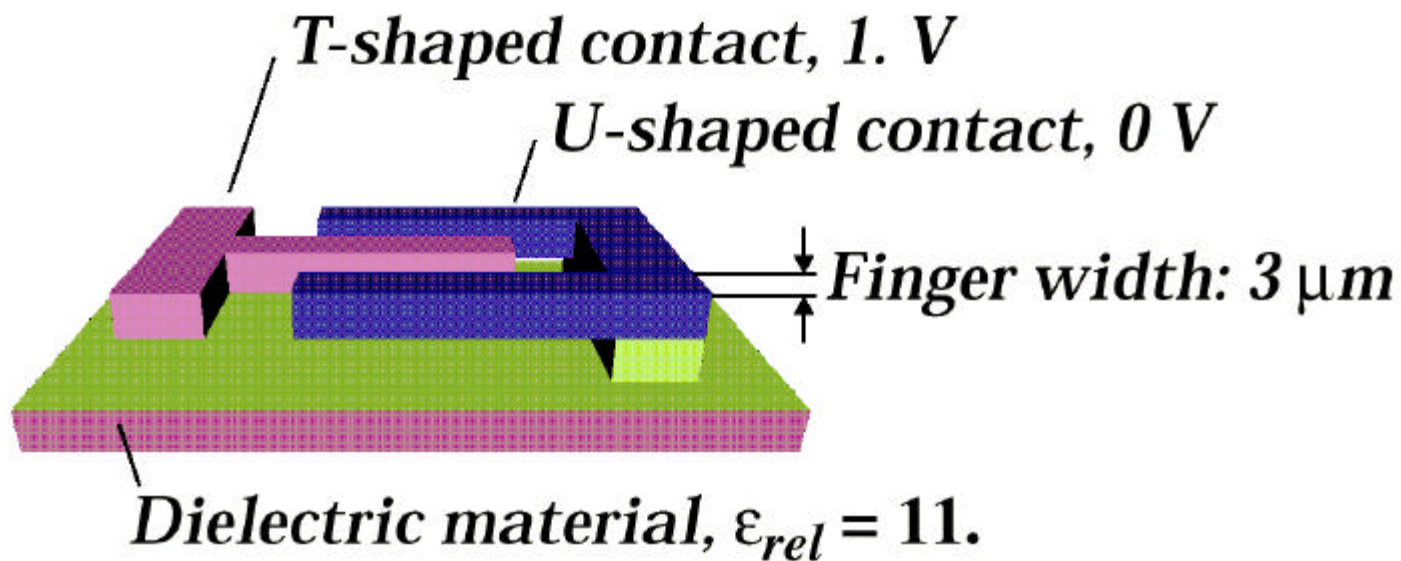


Status of Ptolemy II

- Kernel package complete
- Plot package complete
- Actors package mostly complete
- CT (continuous-time) domain operational
 - Uses a 4-th order Runge-Kutta solver
- PN (process networks) domain operational
 - Uses Java threads
- Data package started
 - Expression evaluator, token classes
- Graph, math, filter packages started

Macro Modeling from Coyote

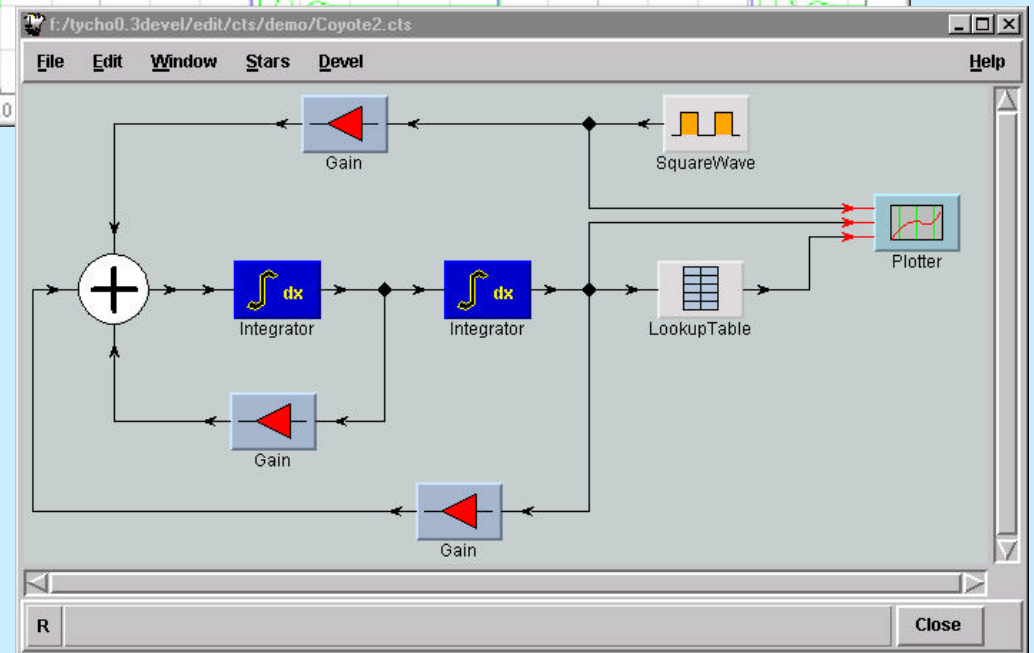
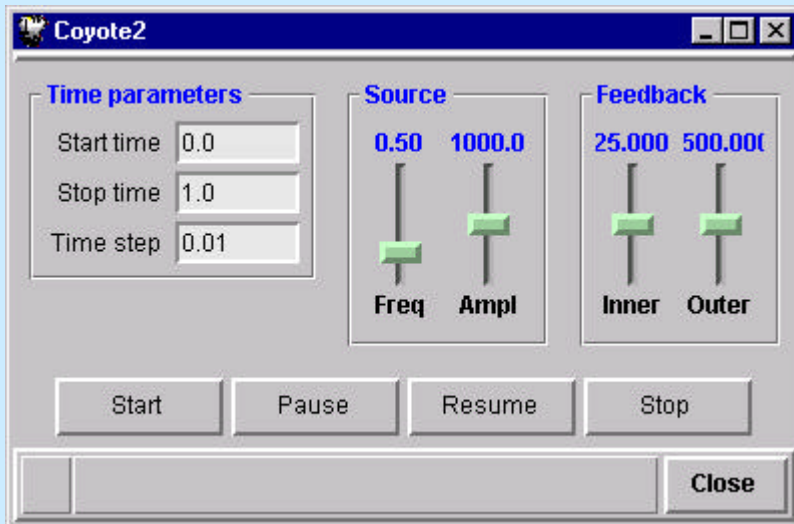
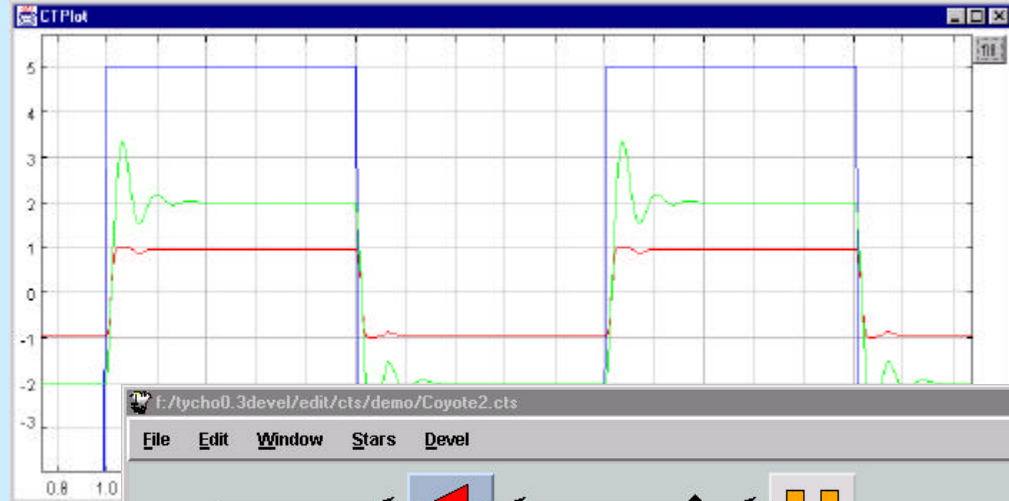
Infinite domain, $\epsilon_{rel} = 1$.



We have incorporated a macro-model of a mechanical comb constructed by Coyote using their electrostatic BEM method.

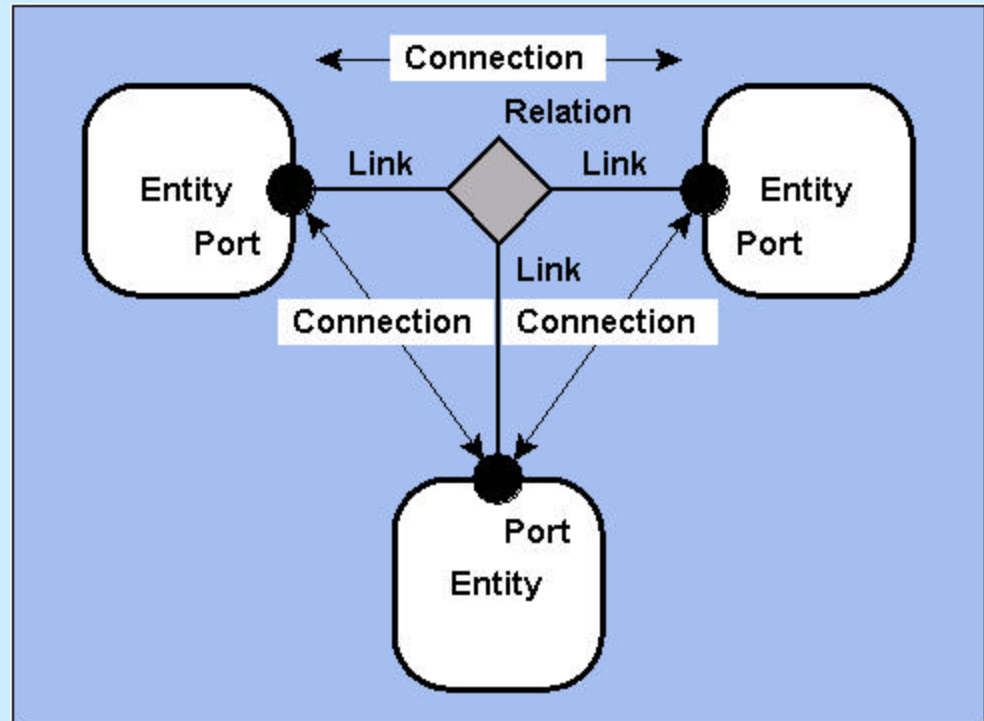
Continuous-Time Domain in Ptolemy II

- Macro model of MEMS comb structure from Coyote Systems



Ptolemy II Abstract Syntax

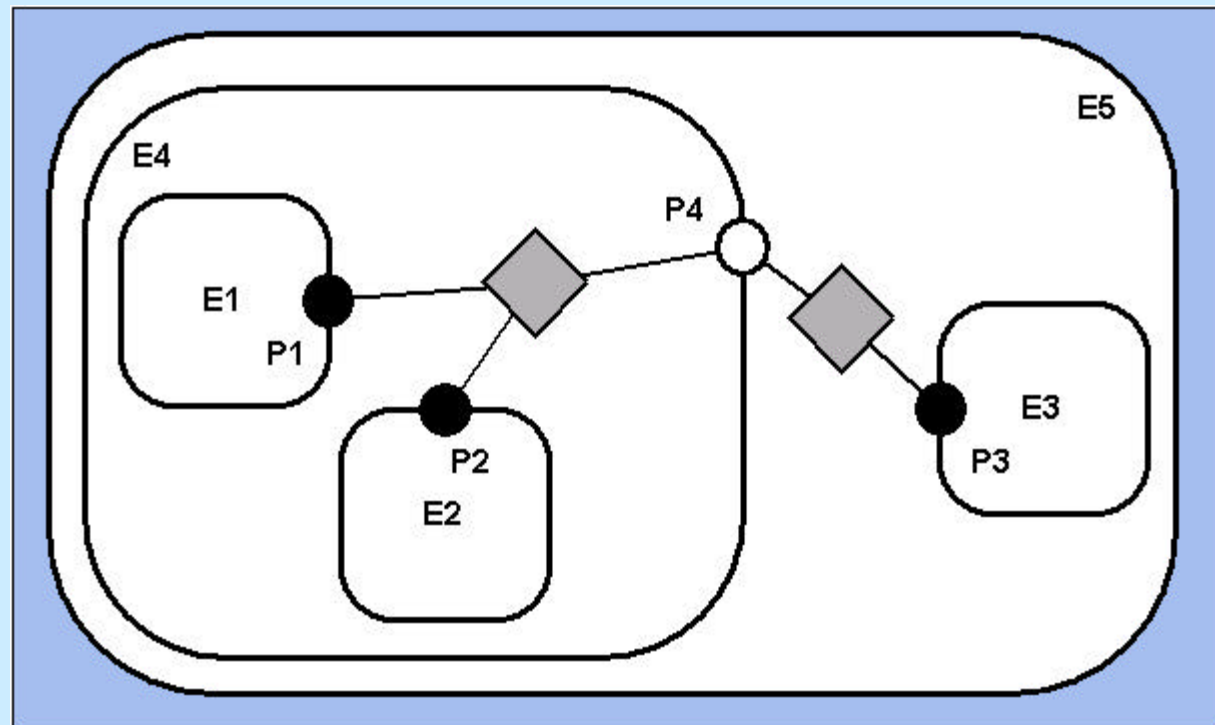
- Entity/Relation bipartite graphs
- Ports are named aggregations of links
- A topology is a linked collection of entities and relations



Clustered Graphs

- Transparent ports
- Transparent entities
- Managed containers

Every object has zero or one containers. If zero, then it is known to its workspace

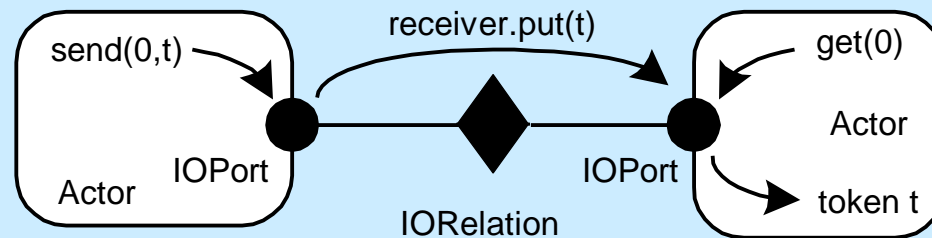


The Actors Package

- Actor
 - Executable model component
- Director
 - Supervisor for model execution
- IOPort
 - Polymorphic message passing

Point-to-Point Transport

Polymorphic transport (behavior depends on the model of computation):



- Sender calls `send(channel,token)`
- This calls `receiver.put(token)` for each receiver
- Receiver calls `get(channel)`
- Architecture also supports:
 - broadcast
 - multicast
 - abstraction

The Data Package

- Tokens
 - encapsulate data for transport
- Parameters
 - attach names and dependencies to tokens
- Expressions
 - operate on tokens
- Type system
 - maximize polymorphism

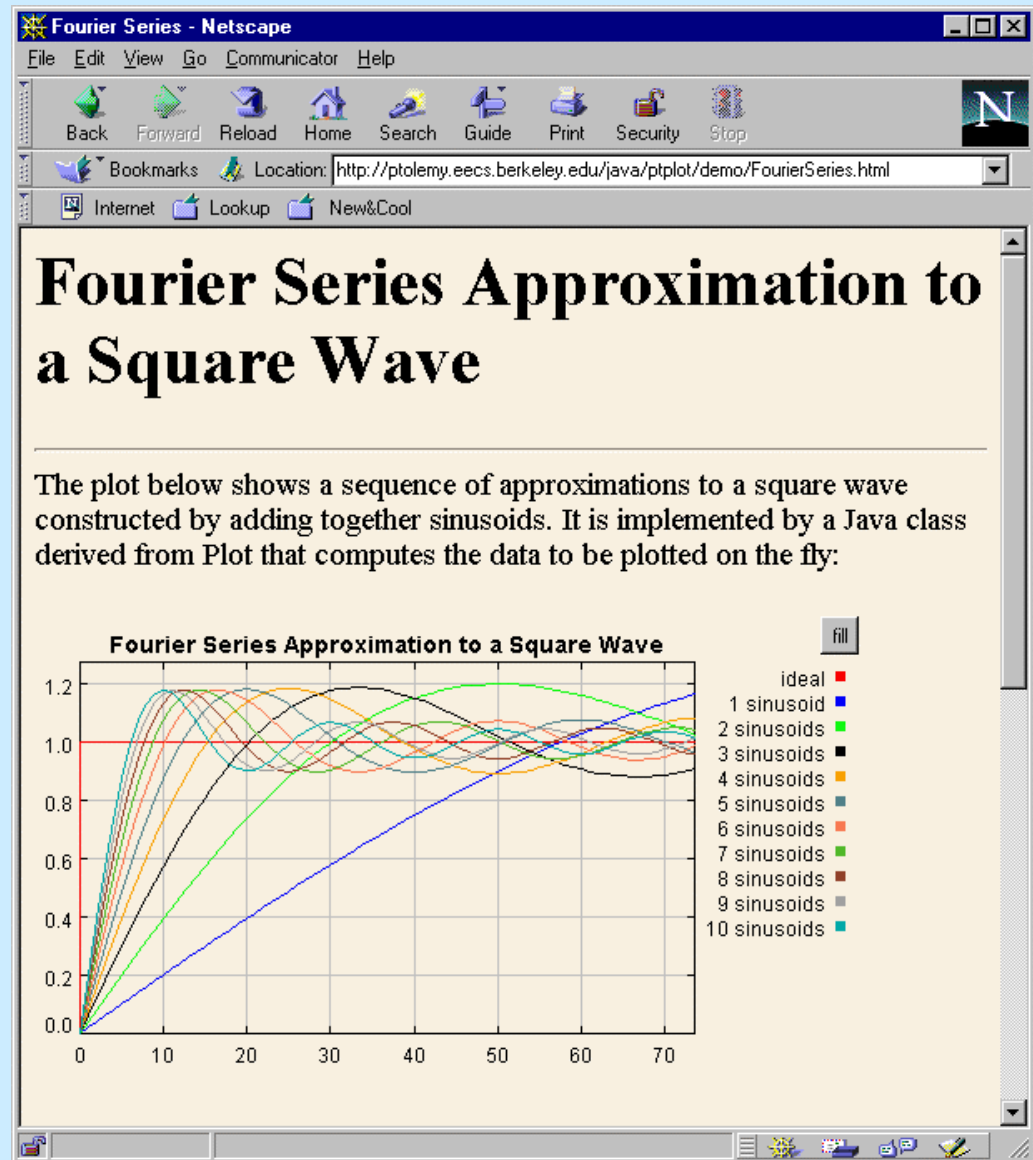
Additional Packages

- Graph (Leda-style graph algorithms)
- Math (Matrix operations, solvers, signal processing)
- Filter (Linear time-invariant systems)
- Plot (interactive, animated signal display)
- HOF (Higher-order functions)
- GUI (Tycho, swing-based tools)

The Plot Package

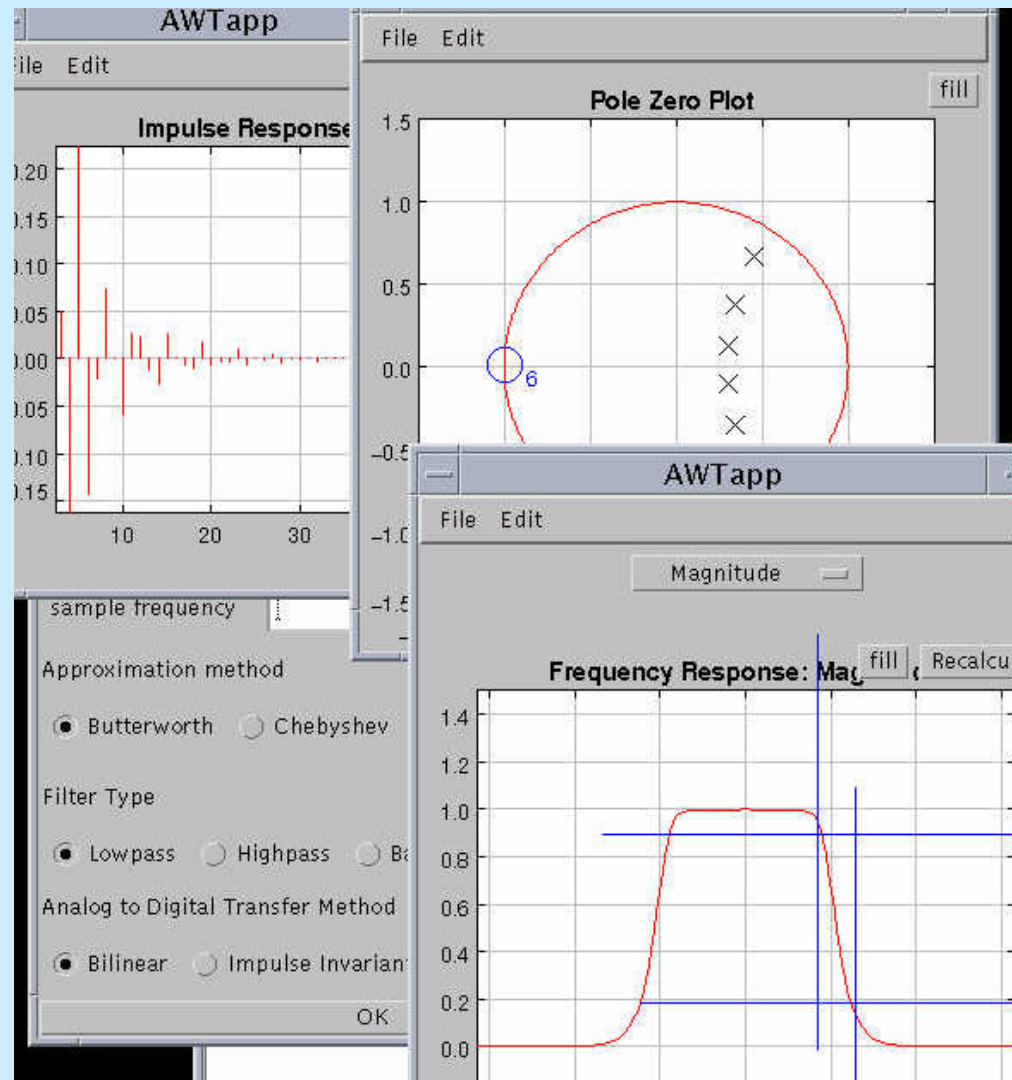
PtPlot is a Java package for interactive, animated signal plotting on the web.

We have used it to learn about Java applets as an interchange and modularization format, and will distribute Ptolemy modules similarly.



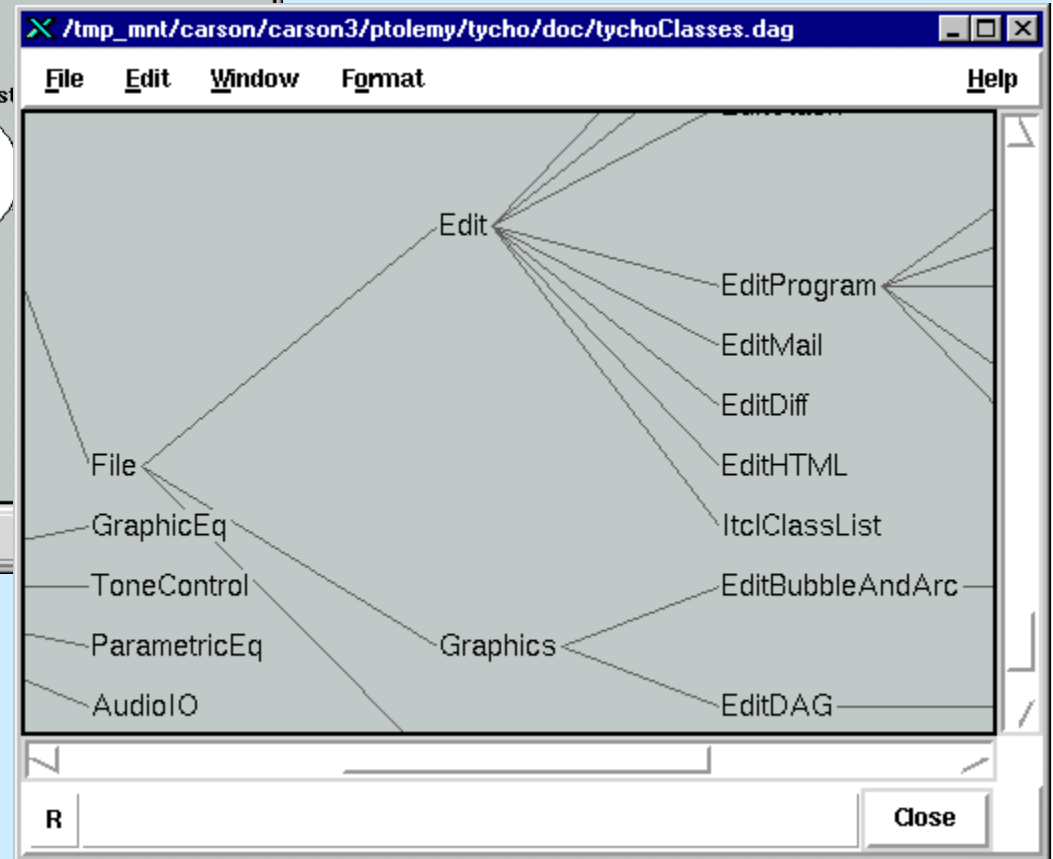
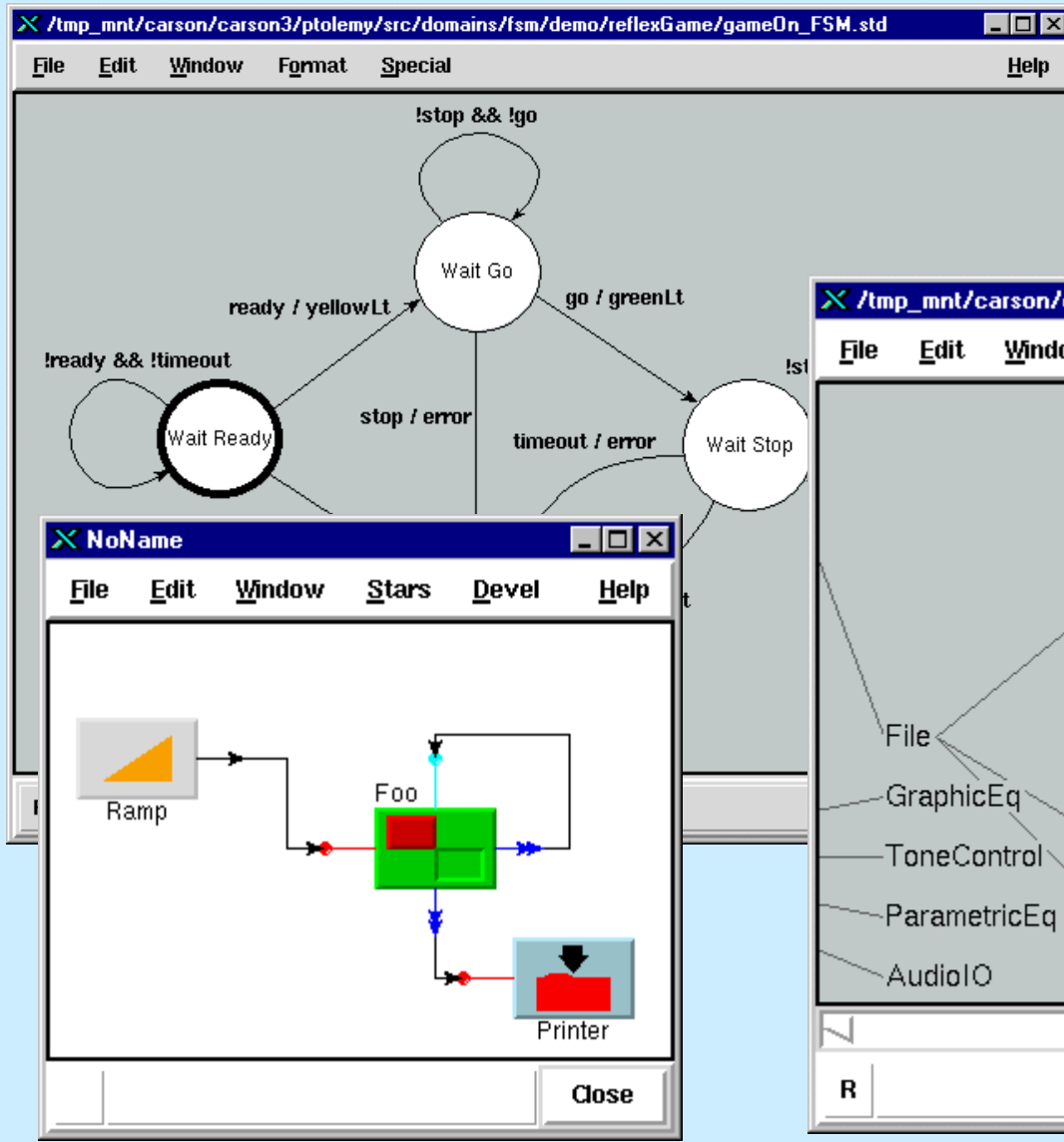
The Filter Package

- Filter design tool.
- Highly interactive.
- Based on Ptpplot.
- Model/view architecture.
- Uses math library.
- Designed to interface to Ptolemy filters.
- Web compatible.



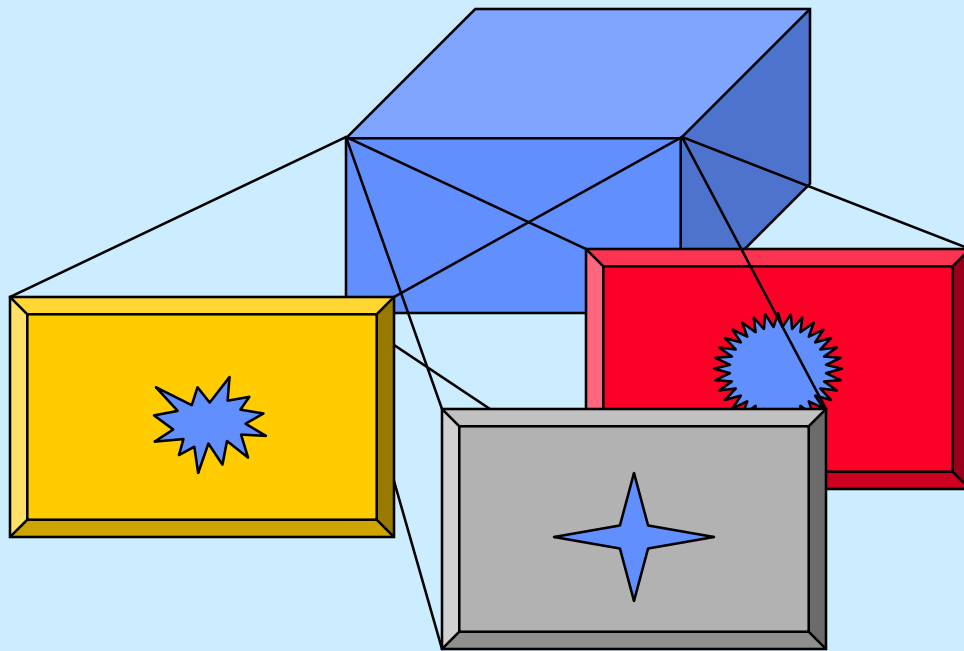
Tycho - GUI package

Tycho is suite of I tcl classes for design representation, manipulation, and visualization.



Model/View Architecture

- Abstract data types
- Publish & subscribe



Major Accomplishments so Far

- Ptolemy II kernel, actors, data packages, CT, and PN domains.
- Semantics for hierarchical interaction of finite-state controllers with several models of computation.
- Formal semantics for DE systems.
- Demonstration of a client-server, web-based mechanism supporting Ptolemy simulations.
- Construction of a network-integrated, scripted design management environment (Tycho).
- Design of an "information model" and an associated "model-view" software architecture (Tycho).
- Release on the net of our first Java module, a multipurpose signal plotter.
- Java/Tycho integration.
- A well-attended Ptolemy miniconference.

Actual Deliverables

- Reports
 - monthly reports
 - annual reports
- Software
 - Tycho 0.2 released (May, 1997)
 - PtPlot 0.1 released (October, 1997)
 - Ptolemy 0.7.1 alpha (May, 1998)
 - Ptolemy II Modules (September 1998 - December 1999)
 - Annual updates of Tycho (est. October, 1998, 1999)
- Papers
 - Reports, journal, and conference papers.

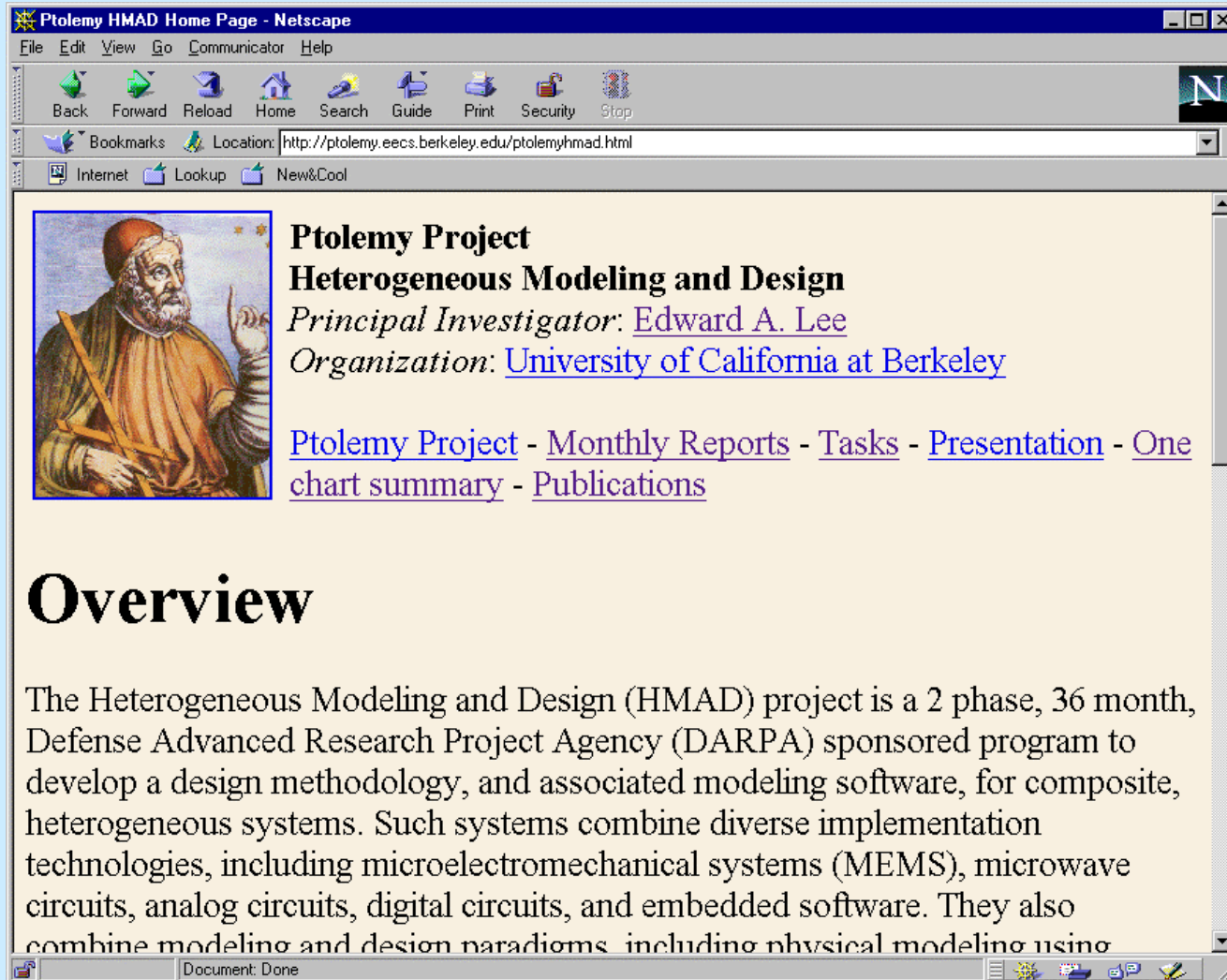
Next Steps

- Finish actors, math, graph, data packages
- Design the wormhole interface
- Implement FSMs in Ptolemy II
- Create hybrid systems modeling
- Create CORBA interface
- Implement dataflow, DE, CSP domains
- Support nondeterminism in PN
- Add time to PN and CSP
- Improve UI (convert to pure Java?)

Technology Transfer

- HP Ptolemy (released in March '98): Supports mixed-signal modeling (DSP + RF).
- BNeD, in cooperation with HP: Design and simulation of optical communication systems based on Ptolemy.
- Cadence: SDF and DE technology in SPW 3.0 and higher.

More Information



Ptolemy Project
Heterogeneous Modeling and Design
Principal Investigator: [Edward A. Lee](#)
Organization: [University of California at Berkeley](#)

[Ptolemy Project](#) - [Monthly Reports](#) - [Tasks](#) - [Presentation](#) - [One chart summary](#) - [Publications](#)

Overview

The Heterogeneous Modeling and Design (HMAD) project is a 2 phase, 36 month, Defense Advanced Research Project Agency (DARPA) sponsored program to develop a design methodology, and associated modeling software, for composite, heterogeneous systems. Such systems combine diverse implementation technologies, including microelectromechanical systems (MEMS), microwave circuits, analog circuits, digital circuits, and embedded software. They also combine modeling and design paradigms, including physical modeling using

<http://ptolemy.eecs.berkeley.edu/ptolemyhmad.html>