Agenda

• 9:00 Introduction and overview of the project
• 9:30 Hierarchical FSMs as a model of discrete control
• 9:45 Ptolemy II architectural overview
• 10:15 break
• 10:30 process networks for concurrent systems modeling
• 10:45 continuous-time modeling in Ptolemy II
• 11:00 Tycho-based user interface toolkit
• 11:45 Wrapup and future plans
• 12:00 Adjourn
Heterogeneous, problem-level description

Modeling, mapping, synthesis

Heterogeneous, implementation-level description
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.
### Staff schedule:

<table>
<thead>
<tr>
<th>ID</th>
<th>Task Name</th>
<th>Start Date</th>
<th>End Date</th>
<th>1998</th>
<th>1999</th>
<th>2000</th>
<th>2001</th>
<th>2002</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Albert Chen</td>
<td>1/20/98</td>
<td>5/25/98</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>John Davis</td>
<td>8/15/96</td>
<td>8/27/99</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Ron Galicia</td>
<td>9/15/94</td>
<td>8/27/99</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Mudit Goel</td>
<td>1/21/97</td>
<td>8/24/01</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Bilung Lee</td>
<td>8/15/93</td>
<td>5/27/99</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Jie Liu</td>
<td>4/20/97</td>
<td>8/24/01</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Xiaojun Liu</td>
<td>5/26/98</td>
<td>5/24/02</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>James Lundblad</td>
<td>1/28/98</td>
<td>1/31/00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Lukito Muliadi</td>
<td>5/26/98</td>
<td>8/27/99</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Steve Neuendorffer</td>
<td>8/26/98</td>
<td>5/24/00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>John Reekie</td>
<td>8/15/95</td>
<td>11/12/99</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Neil Smyth</td>
<td>5/25/97</td>
<td>12/31/98</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Jeff Tsay</td>
<td>8/26/98</td>
<td>5/24/00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Nils Wemhöner</td>
<td>8/26/98</td>
<td>8/24/99</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>Mike Williamson</td>
<td>1/1/92</td>
<td>5/25/98</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>William Wu</td>
<td>8/15/97</td>
<td>12/31/98</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>Yuhong Xiong</td>
<td>1/1/97</td>
<td>8/24/01</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Having other support

- **Graduate students**
- **Undergraduate students**
- **Visiting scholars**
- **Possible graduate students**

#### Permanent staff:

- Jennifer Basler
- Christopher Hylands
- Edward A. Lee
- Mary Stewart
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, ...)

10 - Ptolemy 5/21/98
State Machines & Block Diagrams

Sequential

Concurrent

signal
actor

guard/action

invariant/activity
Hybrid Systems

A discrete program combined with an analog system. A combination of automata and analog computers.

Traditional syntax (classic example: leaking gas burner):

Here, the differential equations hardly look like a concurrency model, but in fact they are.
Alternative View of Hybrid Systems

Analog computers hierarchically combined with automata.

Classic example (leaking gas burner):

\[ \int y \]
\[ \int x \]
\[ z \]
\[ \leq 1 \]
\[ x = 0; \dot{z} = 0 \]
\[ x \geq 30 \]
\[ x = 0; \dot{z} = 1 \]
\[ x \leq 1 \]
\[ x \geq 30 \]

leaking
not leaking

13 - Ptolemy 5/21/98
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.
Ptolemy 0.7 Prototype
Ptolemy II Hybrid Systems

- CT domain: ODE solver in continuous time.
- Generalized wormhole mechanism.
- Emphasis on specification and simulation (not verification).
- Hierarchical visual specifications.
- Interactive, animated simulations.
Continuous-Time Domain in Ptolemy II

• Will support a variety of numerical methods for solving ODEs
  • Time marching, Waveform relaxation, Frequency domain methods, Monte-Carlo methods

• Will mix with:
  • Dataflow
  • DE
  • FSM

• Applications:
  • Mixed signal design
  • MEMS
  • Hybrid systems
Macro Modeling from Coyote

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

T-shaped contact, 1 V

U-shaped contact, 0 V

Finger width: 3 $\mu$m

Dielectric material, $\varepsilon_{rel} = 11$.

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

- **kernel**
  - clustered graph topologies

- **data**
  - encapsulation, parsing, types, dependencies

- **math**
  - matrix operations, algorithms

- **actor**
  - flow-oriented computation

- **automata**
  - hierarchical FSMs

- **plot**
  - interactive display of data

- **graph**
  - graph algorithms

- **actor libraries**

- **domains**

- **filter**
  - signal processing filter design

- **FSM**
Ptolemy II Abstract Syntax

• Entity/Relation bipartite graphs

• Ports are named aggregations of links

• A topology is a linked collection of entities and relations
Flat Abstract Syntax Classes

Interface for objects with names

NamedObj: Naming and synchronization services

NamedObj

+workspace() : Workspace
-name : String
+_workspace : Workspace

Nameable

+description() : String
+getContainer() : Nameable
+getFullName() : String
+getName() : String
+setName(name : String)

Entity

+addPort(p : Port)
+getConnectedPorts() : Enumeration
+getLinkedRelations() : Enumeration
+getPort(name : String) : Port
+getPorts() : Enumeration
+newPort(name : String) : Port
+removeAllPorts()
+removePort(p : Port)

Port

- _container : Entity
- _relationsList : CrossRefList
+getConnectedPorts() : Enumeration
+getLinkedRelations() : Enumeration
+link(r : Relation)
+numLinks() : int
+setContainer(e : Entity)
+unlink(r : Relation)
+unlinkAll()

Relation

- _portList : CrossRefList
+getLinkedPorts() : Enumeration
+getLinkedPortsExcept(p : Port) : Enumeration
+unlink(p : Port)
+unlinkAll()

Entity: aggregation of Ports

Port: aggregation of links to Relations

Relation: aggregation of links to Ports
Every NamedObj has an immutable association with an instance of Workspace. A monitor on Workspace is used for thread synchronization, and the workspace tracks versions of a topology.
Clustered Graphs

- Transparent ports
- Transparent entities
- Managed containers

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

Entity
- _portList : NamedList
+addPort(p : Port)
+getConnectedPorts() : Enumeration
+getLinkedRelations() : Enumeration
+getPort(name : String) : Port
+getPorts() : Enumeration
+newPort(name : String) : Port
+removeAllPorts()
+removePort(p : Port)

Port
- _portList : NamedList
- _container : Entity
- _relationsList : CrossRefList
+getConnectedPorts() : Enumeration
+getLinkedRelations() : Enumeration
+link(r : Relation)
+numLinks() : int
+setContainer(e : Entity)
+unlink(r : Relation)
+unlinkAll()

Relation
- _portList : CrossRefList
- _container : Entity
+getLinkedPorts() : Enumeration
+getLinkedPortsExcept(p : Port) : Enumeration
+numLinks() : int
+unlink(p : Port)
+unlinkAll()

ComponentEntity
- _container : CompositeEntity
+isAtomic() : boolean
+setContainer(c : CompositeEntity)

CompositeEntity
- _containedEntities : NamedList
- _containedRelations : NamedList
+addEntity(e : ComponentEntity)
+addRelation(r : componentRelation)
+allowLevelCrossingConnect(b : boolean)
+connect(p1 : ComponentPort, p2 : ComponentPort)
+connect(p1 : ComponentPort, p2 : ComponentPort, name : String)
+deepContains(e : ComponentEntity) : boolean
+deepGetEntities() : Enumeration
+getEntity(name : String) : ComponentEntity
+getEntities() : Enumeration
+getRelation(name : String) : ComponentRelation
+getRelations() : Enumeration
+newRelation(name : String) : ComponentRelation
+numEntities() : int
+numRelations() : int
+removeAllEntities()
+removeAllRelations()
+removeEntity(e : ComponentEntity)
+removeRelation(r : ComponentRelation)

ComponentPort
- _insideLinks : CrossRefList
+deepGetConnectedPorts() : Enumeration
+deepGetInsidePorts() : Enumeration
+getInsidePorts() : Enumeration
+getInsideRelations() : Enumeration
+getLinkedPorts() : Enumeration
+getLinkedPortsExcept(p : Port) : Enumeration
+liberalLink(r : Relation)
+link(r : Relation)
+numInsideLinks() : int
+unlink(r : Relation)
+unlinkAll()

ComponentRelation
- _container : CompositeEntity
+deepGetLinkedPorts() : Enumeration
+setContainer(c : CompositeEntity)

Composite: Design pattern
Level-Crossing Links

These are supported, but discouraged.
Modularity and compositionality require that relations be able to transparently span the hierarchy.
We have an extensive, automated test suite that uses this.
Make entities executable

**Actors**

```
+fire()
+initialize()
+postfire()
+prefire() : boolean
+wrapup()
```

**Director**

```
_complete() : boolean
_invalidateSchedule() 
_iterate()
+prefire() : boolean
+run()
+scheduleValid() : boolean
+setActors(subsystem : CompositeActor)
```

**Executable**

```
+fire()
+initialize()
+postfire()
+prefire() : boolean
+wrapup()
```

**ComponentEntity**

```
0..n
```

**CompositeEntity**

```
0..1
```

**Actor**

```
+getDirector() : Director
+restructure()
+refineType(t : Type) : Type
```

**CompositeActor**

```
- _director : Director
+getDirector() : Director
+setDirector(d : Director)
```

**NamedObj**

```

```
Director

- Provides a standard template for execution of a model.
- Most of the work is done in the iterate method.

flowchart AQ
  start  [run method]
  next [initialize]
  next [iterate]
  decision  [complete?] [Yes]  [No]
   [Yes]  [wrapup]  [return]
   [No]  [invoke wrapup methods of all actors]
  end

return
Iterate Method

- Fire methods may be invoked multiple times in one iteration.
- Prefire and postfire methods are invoked exactly once.
Prefire Method

- Supports graph mutation

Schedule valid?

- No
  - Topology changed?
    - Yes
      - Restructure topology
      - Invoke restructure methods of all new actors.
    - No
      - Schedule valid?
        - Yes
          - Type resolution
        - No
          - Invoke initialize methods of all new actors

- Yes
  - Static Scheduling
  - end

Evaluate type refinement functions of all actors until fixed point

Invoke initialize methods of all new actors
Support for:

- Inputs and Outputs
- Broadcast and Multicast
- Polymorphic in communication protocols
Polymorphic Message Passing

The type of port used depends on the model of computation.
Point-to-Point Transport

Simple transport:

• Sender calls `send(channel, token)`
• This calls `receiver.put(token)` for each receiver
• Receiver calls `get(channel)`
Multiports

Multiple (scalable) distinct channels:

• Sender calls send(channel,token)
• This calls receiver.put(token) for each receiver
• Receiver calls get(channel)
Multicast Transport

- Multiple destinations for the same data
- Tokens are cloned automatically
Sets of destination receivers are cached at the sender for efficiency.
The Data Package

• Tokens
  - encapsulate data for transport

• Parameters
  - attach names and dependencies to tokens

• Expressions
  - operate on tokens

• Type system
  - maximize polymorphism
PtParser

• Created using JavaCC & JJTree
  
  foo.jjt => JJTREE => foo.jj => JAVACC => foo.java

• Recognizes full range of arithmetic, relational and logical operators

• overloading in some cases to “do the right thing”
  
  eg 3+4 ==> 7
  3+" hello" ==> “3 hello”
PtParser

• allows reference to Ptolemy II Params, passed as an argument to the parser
  • eg if clkFreq and duration are Params in the current Entity
    - then delay Param could be given the value duration/clkFreq or (duration*2)/clkFreq etc.

• using reflection, all the functionality of java.lang.Math is available

• extendable: easy to add new functionality
  eg tcl(...) to invoke tcl to evaluate a string
  readFile("foo.bar") to read the given file as input
Type System

- Two-levels of types:
  - data type of atomic exchanges
  - signal type governing the exchange protocol
- Type hierarchy:
  - a lattice
- Type specification:
  - a monotonic function on this lattice that refines the estimated types.
- Type resolution:
  - iterate to a fixed point.
Use of Infrastructure in CT

• **CTDirector**
  - uses graph package for topological sort
  - overrides Director methods to implement a 4-th order Runge-Kutta solver.

• **IOPort**
  - use the default Mailbox receiver

• **Data**
  - use data package for parameters, signals

• **Plot**
  - use ptplot for interactive, animated plots
Use of Infrastructure in PN

• PNDirector
  - Kahn process networks model of computation
  - uses Java threads, one for each actor
  - manages deadlocks

• PNPort
  - Uses unbounded FIFO queues
  - Blocking reads
  - Bounded memory, when possible

• Very different from CT, hence stresses the kernel & actors package designs
Applications of PN

• Concurrent digital control processes with dynamic behavior.
• Provides a higher-level concurrency model than threads for programming in Java.
• (Future) With the addition of nondeterminism, model resource management problems.
• (Future) With the addition of time, model mappings of applications to hardware.
Synthesis

- Director manages synthesis process (vs. simulation)
- Separate interface from implementation.
- Support migration from simulation to implementation.

Note: We have a subcontract from Lockheed-Sanders in the adaptive computing program to develop technology for FPGA synthesis.
Additional Packages

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

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

- Use Java and Itcl
- Split Ptolemy into Java packages
- Split Tycho into Itcl packages
- Make everything network aware
- Use object modeling
- Use the model-view design pattern
- Use object-request broker technology
- Experiment with reflection, remote method invocation, etc.
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.
Tycho is suite of Itcl classes for design representation, manipulation, and visualization.
Model/View Architecture

- Abstract data types
- Publish & subscribe
Software Engineering

- Code rating system: red, yellow, green, blue.
- Author/reviewer division of responsibilities.
- Automated test suites (scripted, in Tcl).
- Code coverage measurements.
- Integrated documentation.
- Tycho support.

The Ptolemy group has a tradition of emphasizing code and documentation quality.
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.
Interoperability

- CORBA
- Java Beans and the Tcl Bean
- An open architecture
- Small, modular Java packages
- Well-defined semantics
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.
Future Work

- 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
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...
Publications


- A. Girault, B. Lee, and E. A. Lee, ``A Preliminary Study of Hierarchical Finite State Machines with Multiple Concurrency Models,' Memorandum UCB/ERL M97/57, Electronics Research Laboratory, University of California, Berkeley, CA 94720, August 1997.


Publications (continued)


- Richard S. Stevens (Naval Research Laboratory), Marlene Wan, Peggy Laramie (UCB), Thomas M. Parks (MIT Lincoln Labs), and Edward A. Lee (UCB), "Implementation of Process Networks in Java," UCB/ERL Tech. Report, number pending, November 1997.
Publications (continued)

Under subcontract to UT Austin (Brian Evans):

