Heterogeneous Modeling and Design
- Edward A. Lee (PI) -

**Staff**
Christopher Hylands
Fiona Sinclair
Mary P. Stewart

**Postdoctoral Researchers**
James Lundblad
H. John Reekie

**Students**
Albert Chen
John Davis, II
Ron Galicia
Mudit Goel
Bilung Lee
Jie Liu
Neil Smyth
Michael C. Williamson
William Wu
Yuhong Xiong
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.
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, ...)

8 - Ptolemy 3/10/98
State Machines & Block Diagrams

\[ a^\wedge: b/v \]
\[ a_: b/u,v \]
\[ a^\wedge b \]

A
B
C
D
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):

\[
\begin{align*}
x &\leq 1 \\
x &\geq 30 \\
x &= 0
\end{align*}
\]

leaking

not leaking

\[
\begin{align*}
\dot{x} &\rightarrow x \\
\int &\rightarrow \\
\dot{z} &\rightarrow \dot{z}
\end{align*}
\]

\[
\begin{align*}
\dot{y} &\rightarrow y \\
\int &\rightarrow y \\
\dot{z} &\rightarrow \dot{z}
\end{align*}
\]
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
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

- Support a variety of numerical methods for solving ordinary differential equations
  - Time marching, Waveform relaxation, Frequency domain methods, Monte-Carlo methods

- Mix with:
  - Dataflow
  - DE
  - FSM

- Applications:
  - Mixed signal design
  - MEMS
  - Hybrid 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
Flat Abstract Syntax Classes

NamedObj: Naming and synchronization services

Interface for objects with names

NamedObj

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

Nameable

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

Entity

+addPort(p : Port)
+getConnectedPorts() : Enumeration
+getContainer() : Nameable
+getFullName() : String
+getName() : String
+setName(name : String)

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

- _portList : NamedList

Port

- _container : Entity
- _relationsList : CrossRefList
+getConnectedPorts() : Enumeration
+getContainer() : Nameable
+getFullName() : String
+getName() : String
+setName(name : String)

Ev

- _portList : NamedList

Relation

- _portList : CrossRefList
+getConnectedPorts() : Enumeration
+getContainer() : Nameable
+getFullName() : String
+getName() : String
+setName(name : String)

Ev

0..n

Port: aggregation of links to Relations

Relation: aggregation of links to Ports

Entity: aggregation of 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.
Hierarchy

- Transparent ports
- Transparent entities
- Managed containers

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

- _container : Entity
  _relationsList : CrossRefList
  +getConnectedPorts() : Enumeration
  +getLinkedRelations() : Enumeration
  +link(r : Relation)
  +numLinks() : int
  +setContainer(e : Entity)
  +unlink(r : Relation)
  +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
  +deepGetConnectedPorts() : Enumeration
  +deepGetInsidePorts() : Enumeration
  +getInsidePorts() : Enumeration
  +getInsideRelations() : Enumeration
  +getPort(name : String) : Port
  +getPorts() : Enumeration
  +newPort(name : String) : Port
  +numInsideLinks() : int
  +unlink(r : Relation)
  +unlinkAll()

ComponentPort
- _insideLinks : CrossRefList
  0..n
  container

  +deepGetConnectedPorts() : Enumeration
  +deepGetInsidePorts() : Enumeration
  +getInsidePorts() : Enumeration
  +getInsideRelations() : Enumeration
  +link(r : Relation)
  +numInsideLinks() : int
  +setContainer(c : Entity)
  +unlink(r : Relation)
  +unlinkAll()

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

Composite: Design pattern
These are supported, but discouraged.
Modularity and compositionality require that relations be able to transparently span the hierarchy.
TclBlend Interface

set e0 [java::new pt.kernel.CompositeEntity E0]
set e1 [java::new pt.kernel.ComponentEntity E1]
set e2 [java::new pt.kernel.ComponentEntity E2]
set p1 [$e1 newPort P1]
set p2 [$e1 newPort P2]
set r1 [$e0 connect $p1 $p2 R1]
The Actors Package

Constraints on Relations

Make entities executable

Same interface
Phases of Execution

- Constructor
- Type resolution
- Setup invocation (when needed)
- Static scheduling (when needed)
- Execution (a series of iterations)
  - evaluation
  - update
- Wrapup

Parameters are set before type resolution or between iterations. They are observable and observers may trigger further action (such as recomputing the schedule).
Data Transfer

- Sender calls `put(token)`
- This calls `receive(token)` for each receiver
- Tokens are cloned automatically
- Receiver calls `get()`
Synthesis

- Executive 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.
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.
Additional Packages

- Token (carry data)
- Parameters (customize applications)
- Controllers (set parameters)
- Static graph package
- Dynamic graph package
- Math package
PtFilter

- 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 a 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 (in beta test now, release imminent): 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

- RMI and/or CORBA
- Java Beans and the Tcl Bean
- An open architecture
- Small, modular Java packages
- Well-defined semantics
Major Accomplishments so Far

- Ptolemy II kernel, 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 II Modules (June 1998 - December 1999)
  - Annual updates of Tycho (est. October, 1998, 1999)

- Papers
  - Reports, journal, and conference papers.
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
Publications


Publications (continued)


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):

