

# Concurrent Models of Computation for Embedded Software

Edward A. Lee

Professor, UC Berkeley  
EECS 290n – Advanced Topics in Systems Theory  
Fall, 2004

Copyright © 2004, Edward A. Lee, All rights reserved

Lecture 5: Extending Ptolemy II

## Background for Ptolemy II

### Gabriel (1986-1991)

- Written in Lisp
- Aimed at signal processing
- Synchronous dataflow (SDF) block diagrams
- Parallel schedulers
- Code generators for DSPs
- Hardware/software co-simulators

### Ptolemy Classic (1990-1997)

- Written in C++
- Multiple models of computation
- Hierarchical heterogeneity
- Dataflow variants: BDF, DDF, PN
- C/VHDL/DSP code generators
- Optimizing SDF schedulers
- Higher-order components

### Ptolemy II (1996-2022)

- Written in Java
- Domain polymorphism
- Multithreaded
- Network integrated
- Modal models
- Sophisticated type system
- CT, HDF, CI, GR, etc.

Each of these served us, first-and-foremost, as a laboratory for investigating design.

### PtPlot (1997-??)

- Java plotting package

### Tycho (1996-1998)

- Itcl/Tk GUI framework

### Diva (1998-2000)

- Java GUI framework

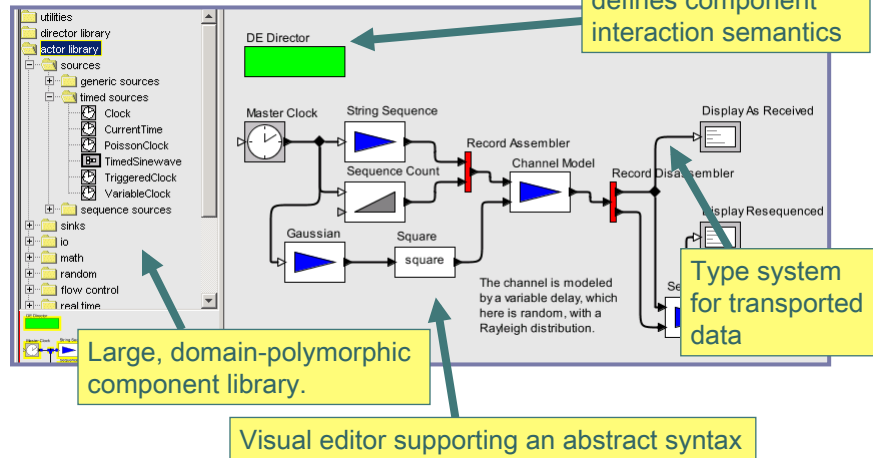
All open source.  
All truly free software (cf. FSF).

Lee 05: 2

## Framework Infrastructure that Supports Diverse Experiments with Models of Computation

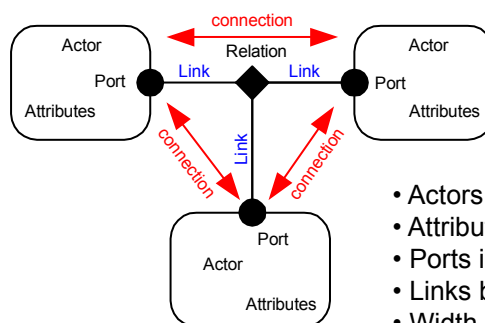
Concurrency management supporting dynamic model structure.

Director from a library defines component interaction semantics



Lee 05: 3

## The Basic Abstract Syntax



- Actors
- Attributes on actors (parameters)
- Ports in actors
- Links between ports
- Width on links (channels)
- Hierarchy

Concrete syntaxes:

- XML
- Visual pictures
- Actor languages (Cal, StreamIT, ...)

Lee 05: 4

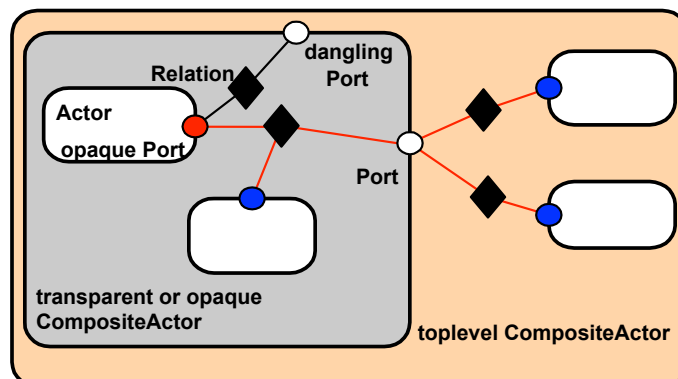
## MoML XML Schema for this Abstract Syntax

Ptolemy II designs are represented in XML:

```
...  
<entity name="FFT" class="ptolemy.domains.sdf.lib.FFT">  
  <property name="order" class="ptolemy.data.expr.Parameter" value="order">  
  </property>  
  <port name="input" class="ptolemy.domains.sdf.kernel.SDFIOPort">  
    ...  
  </port>  
  ...  
</entity>  
...  
<link port="FFT.input" relation="relation"/>  
<link port="AbsoluteValue2.output" relation="relation"/>  
...
```

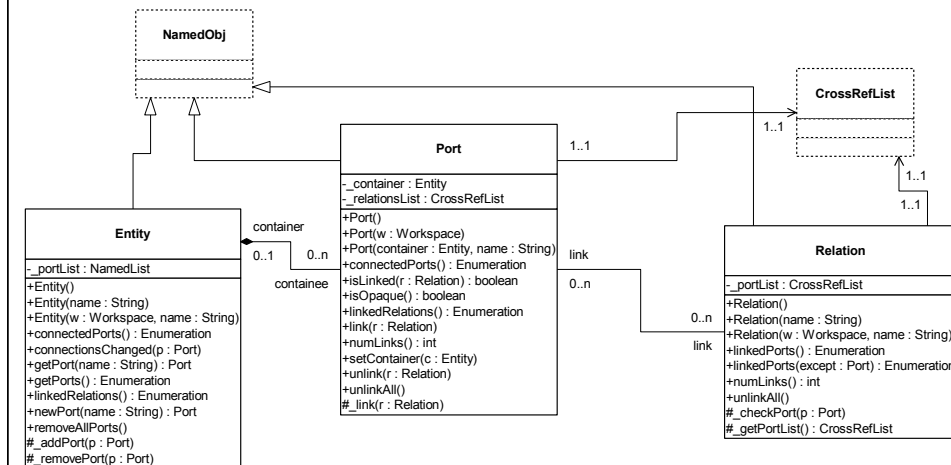
Lee 05: 5

## Hierarchy - Composite Components



Lee 05: 6

## Kernel Classes Support the Abstract Syntax



Lee 05: 7

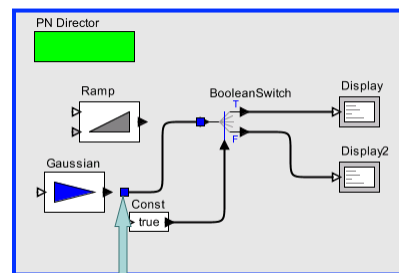
## Concurrency Management Supporting Dynamic Model Structure

Changes to a model while the model is executing:

- Change parameter values
- Change model structure

How can this be made safe?

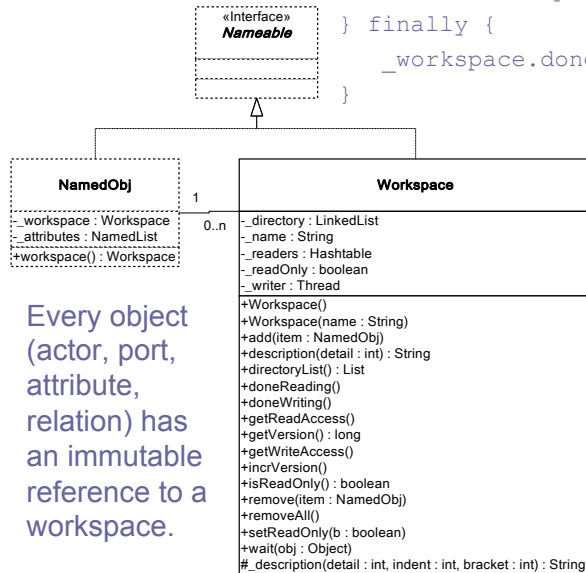
- Workspace class
- ChangeRequest class
- stopFire() method



Can dynamically modify the model while it executes... safely.

Lee 05: 8

## Workspace



Every object (actor, port, attribute, relation) has an immutable reference to a workspace.

```

try {
    _workspace.getReadAccess();
    ... actions depending on model structure
} finally {
    _workspace.doneReading();
}
  
```

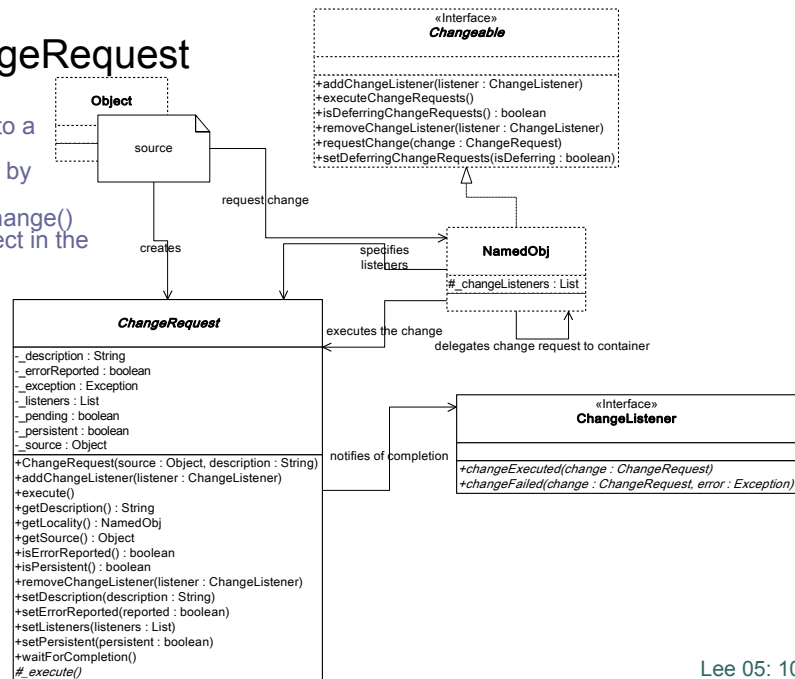
Many threads can have read access at the same time. Only one thread can have write access, and only if no other thread has read access.

Specialized wait(Object) method releases the locks during the wait().

Lee 05: 9

## ChangeRequest

Changes to a model are requested by calling requestChange() on an object in the model.



Lee 05: 10

## When to Execute Change Requests

In many models of computation, there is a natural time: between iterations.

In PN, this is not a trivial question...

- All threads must be stopped (blocked)
  - On reads
  - On writes to full buffers
  - Or block themselves with a wait()
- What happens when the model structure changes during a call to get()?

Lee 05: 11

## ProcessThread with Pauses for Mutations

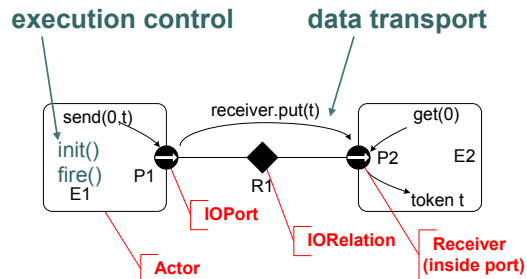
```
while(iterate) {
    if (_director.isStopFireRequested()) {
        synchronized (_director) {
            _director._actorHasStopped();
            while (_director.isStopFireRequested()) {
                try {
                    workspace.wait(_director);
                } catch (InterruptedException ex) {
                    break;
                }
            }
            _director._actorHasRestarted();
        }
    }
}

boolean iterate = true;
while (iterate) {
    if (_actor.prefire()) {
        _actor.fire();
        iterate = _actor.postfire();
    }
}
```

Specialized wait() method releases workspace locks while the thread is suspended.

Lee 05: 12

## Abstract Semantics of Actor-Oriented Models of Computation

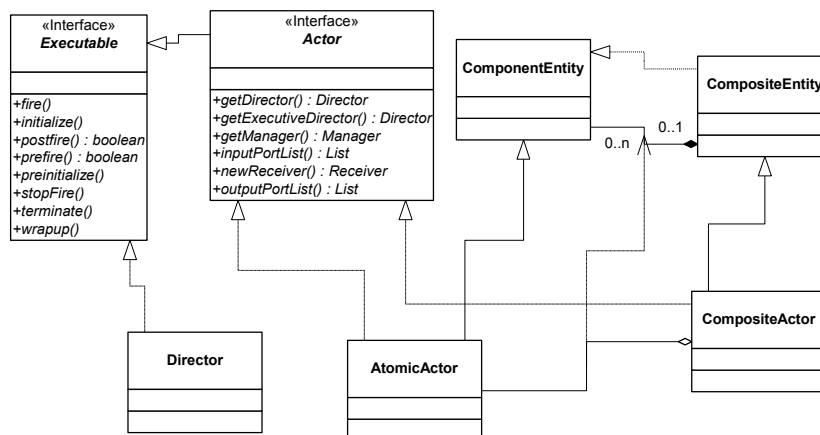


Actor-Oriented Models of Computation that we have implemented:

- dataflow (several variants)
- process networks
- distributed process networks
- Click (push/pull)
- continuous-time
- CSP (rendezvous)
- discrete events
- distributed discrete events
- synchronous/reactive
- time-driven (several variants)
- ...

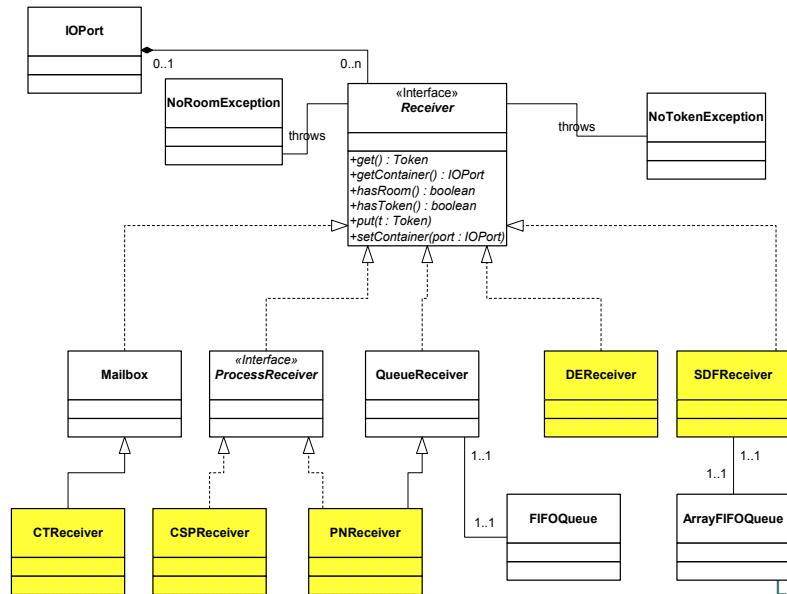
Lee 05: 13

## Object Model for Executable Components



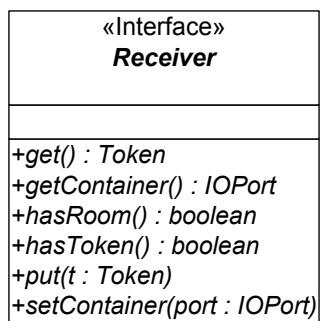
Lee 05: 14

## Object Model (Simplified) for Communication Infrastructure



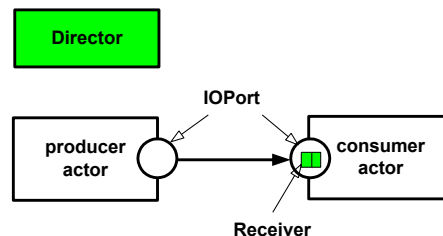
Lee 05: 15

## Object-Oriented Approach to Achieving Behavioral Polymorphism



These polymorphic methods implement the communication semantics of a domain in Ptolemy II. The receiver instance used in communication is supplied by the director, not by the component.

**Recall: Behavioral polymorphism** is the idea that components can be defined to operate with multiple models of computation and multiple middleware frameworks.



Lee 05: 16



## Extension Exercise

Build a director that subclasses `PNDirector` to allow ports to alter the “blocking read” behavior. In particular, if a port has a parameter named “`tellTheTruth`” then the receivers that your director creates should “tell the truth” when `hasToken()` is called. That is, instead of always returning true, they should return true only if there is a token in the receiver.

Parameterizing the behavior of a receiver is a simple form of communication refinement, a key principle in, for example, *Metropolis*.

Lee 05: 17

## Implementation of the New Model of Computation

```
package experiment;

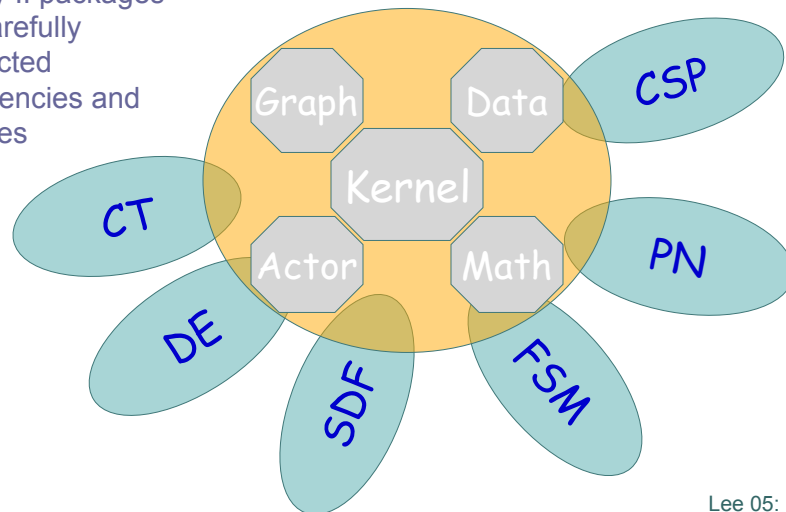
import ...

public class NondogmaticPNDirector extends PNDirector {
    public NondogmaticPNDirector(CompositeEntity container, String name)
        throws IllegalArgumentException, NameDuplicationException {
        super(container, name);
    }
    public Receiver newReceiver() {
        return new FlexibleReceiver();
    }
    public class FlexibleReceiver extends PNQueueReceiver {
        public boolean hasToken() {
            IOPort port = getContainer();
            Attribute attribute = port.getAttribute("tellTheTruth");
            if (attribute == null) {
                return super.hasToken();
            }
            // Tell the truth...
            return _queue.size() > 0;
        }
    }
}
```

Lee 05: 18

## Ptolemy II Software Architecture Built for Extensibility

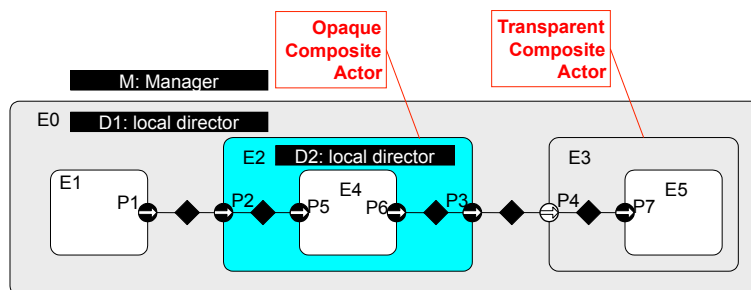
Ptolemy II packages  
have carefully  
constructed  
dependencies and  
interfaces



Lee 05: 19

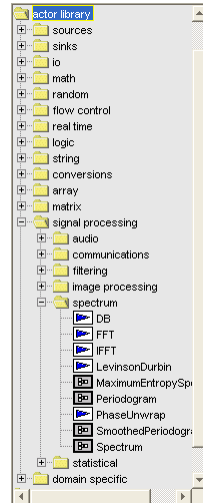
## Hierarchical Heterogeneity

Directors are domain-specific. A composite actor with a director becomes opaque. The Manager is domain-independent.

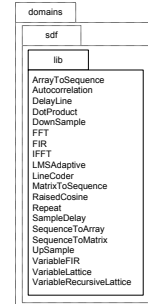
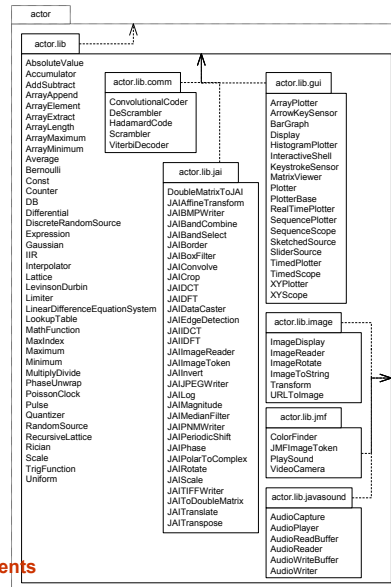


Lee 05: 20

## Ptolemy II Component Library



- Data polymorphic components
- Behaviorally polymorphic components



UML package diagram of key actor libraries included with Ptolemy II.

Lee 05: 21

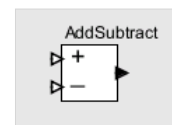
## Polymorphic Components - Component Library Works Across Data Types and Domains

### Data polymorphism:

- Add numbers (int, float, double, Complex)
- Add strings (concatenation)
- Add composite types (arrays, records, matrices)
- Add user-defined types

### Behavioral polymorphism:

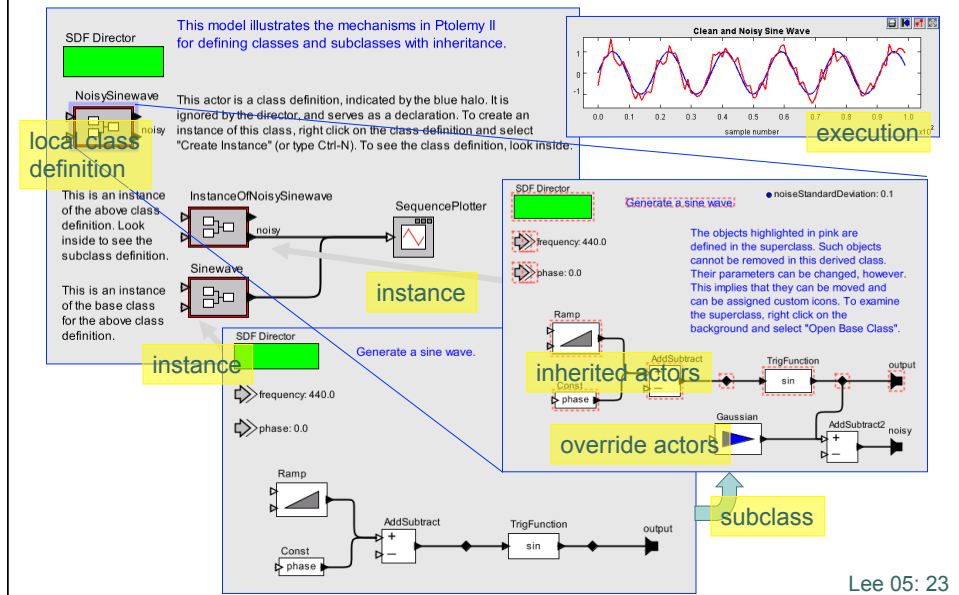
- In dataflow, add when all connected inputs have data
- In a time-triggered model, add when the clock ticks
- In discrete-event, add when any connected input has data, and add in zero time
- In process networks, execute an infinite loop in a thread that blocks when reading empty inputs
- In CSP, execute an infinite loop that performs rendezvous on input or output
- In push/pull, ports are push or pull (declared or inferred) and behave accordingly
- In real-time CORBA, priorities are associated with ports and a dispatcher determines when to add



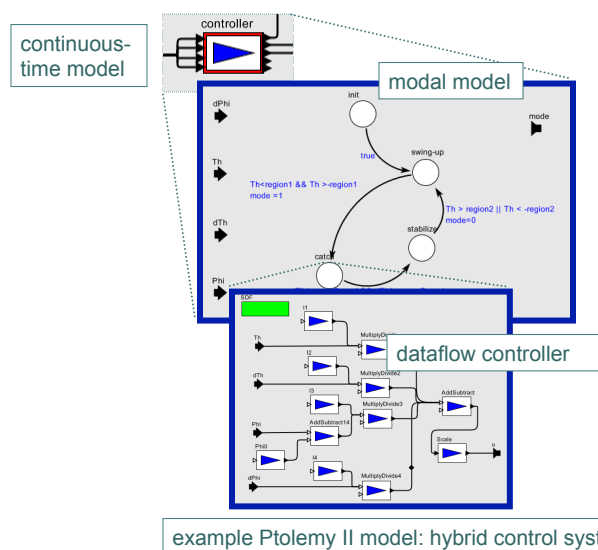
By not choosing among these when defining the component, we get a huge increment in component re-usability. But how do we ensure that the component will work in all these circumstances?

Lee 05: 22

## Shared Infrastructure Modularity Mechanisms



## More Shared Infrastructure: Hierarchical Heterogeneity and Modal Models



Lee 05: 24

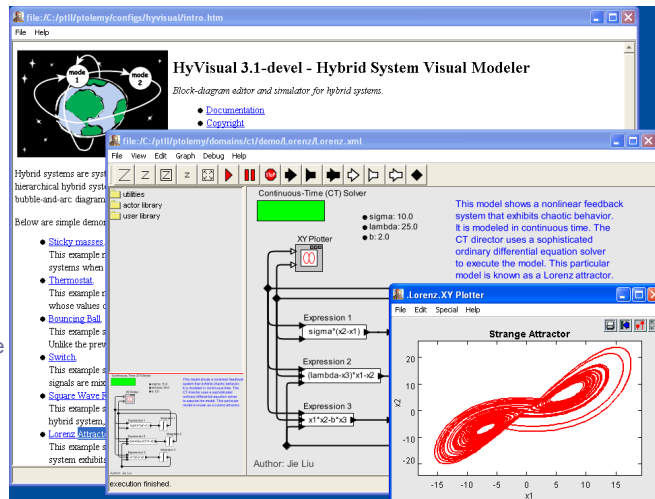
## Branding

Ptolemy II *configurations* are Ptolemy II models that specify

- welcome window
- help menu contents
- library contents
- File->New menu contents
- default model structure
- etc.

A configuration can identify its own “brand” independent of the “Ptolemy II” name and can have more targeted objectives.

An example is HyVisual, a tool for hybrid system modeling. VisualSense is another tool for wireless sensor network modeling.



Lee 05: 25

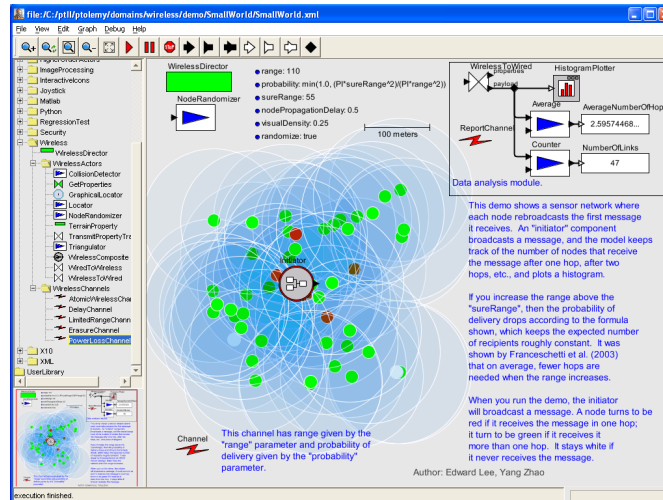
## Ptolemy II Extension Points

- Define actors
- Interface to foreign tools (e.g. Python, MATLAB)
- Interface to verification tools (e.g. Chic)
- Define actor definition languages
- Define directors (and models of computation)
- Define visual editors
- Define textual syntaxes and editors
- Packaged, branded configurations

All of our “domains” are extensions built on a core infrastructure.

Lee 05: 26

## Example Extension: VisualSense

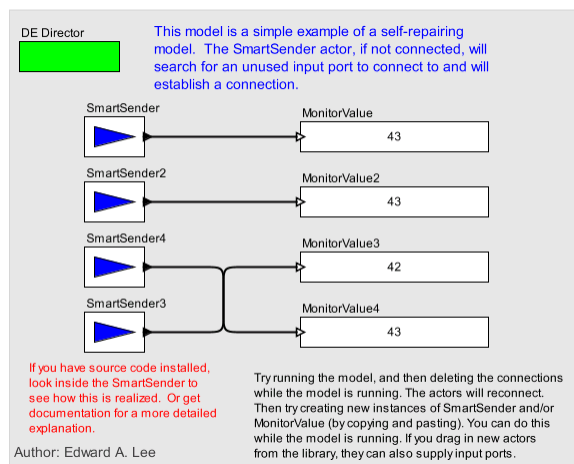


- Branded
- Customized visualization
- Customized model of computation (an extension of DE)
- Customized actor library
- Motivated some extensions to the core (e.g. classes, icon editor).

Lee 05: 27

## Example Extensions: Self-Repairing Models

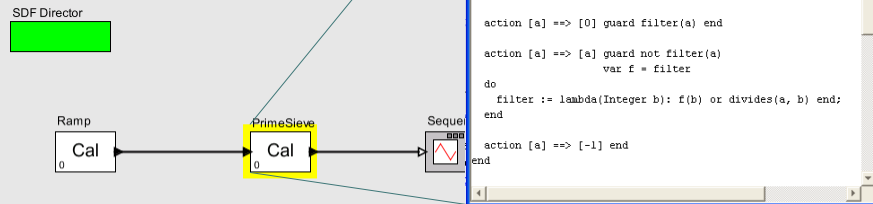
Concept demonstration built together with Boeing to show how to write actors that adaptively reconstruct connections when the model structure changes.



Lee 05: 28

## Example Extensions Python Actors and Cal Actors

Cal is an experimental language for defining actors that is analyzable for key behavioral properties.



This model demonstrates the use of function closures inside a CAL actor.

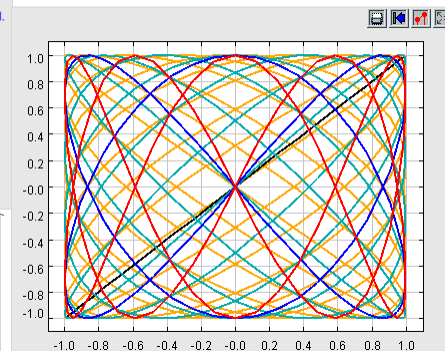
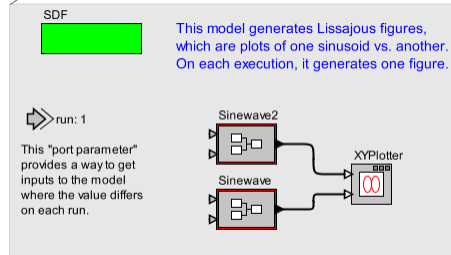
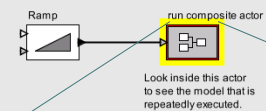
The PrimeSieve actor uses nested function closures to realize the Sieve of Eratosthenes, a method for finding prime numbers. Its state variable, "filter," contains the current filter function. If it is "false" a new prime number has been found, and a new filter function will be generated.

The PrimeSieve actor expects an ascending sequence of natural numbers, starting from 2, as input.

Lee 05: 29

## Example Extensions Using Models to Control Models

This model illustrates the use of a "run composite actor" component. That component contains another Ptolemy II model. Each time it fires, it performs a complete execution of that other Ptolemy II model, rather than just one firing as would be typical of a composite actor's

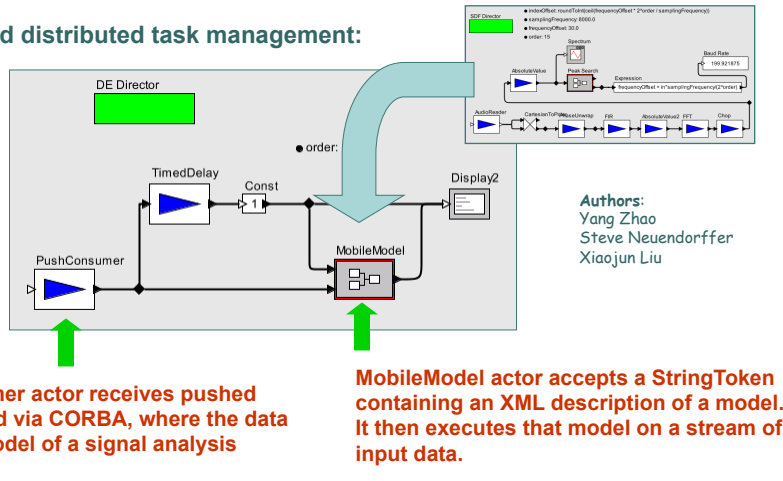


This is an example of a "higher-order component," or an actor that references one or more other actors.

Lee 05: 30

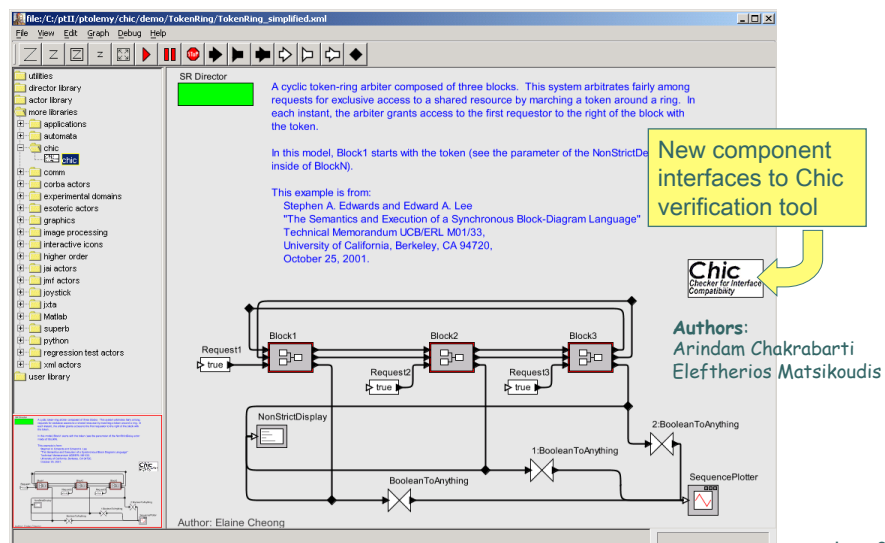
## Examples of Extensions Mobile Models

### Model-based distributed task management:



Lee 05: 31

## Examples of Extensions Hooks to Verification Tools



Lee 05: 32



## Examples of Extensions Hooks to Verification Tools

**Synchronous assume/  
guarantee interface  
specification for  
Block1**

```

interface Block1
  input TI, FI, R;
  output T0, P0, G;

  state b;
  assume !TI;
  guarantee T0;
  true -> a;
  
```

Lee 05: 33

## Examples of Extensions Hooks to Verification Tools

**Chic**  
Checker for Interface Compatibility




Chic version 1.0  
Copyright 2002 Regents of the University of California  
ALL RIGHTS RESERVED  
Send bug reports to arindam@CS.Berkeley.EDU  
Visit <http://www.cs.berkeley.edu/~arindam/Chic> for updates

Welcome to Chic version 1.0  
Copyright 2002 Regents of the University of California  
ALL RIGHTS RESERVED  
Interface Request1 was read. Checking well formedness.  
Interface Block2 was read. Checking well formedness.  
Interface Block3 was read. Checking well formedness.  
Interface Block1 was read. Checking well formedness.  
Interface Request2 was read. Checking well formedness.  
Interface Request3 was read. Checking well formedness.

Configure (Ctrl-E) Ctrl+E  
Customize Name  
Get Documentation  
Set Icon  
Chic: Asynchronous I/O  
Chic: Synchronous A/G  
Look Inside (Ctrl+L) Ctrl+L

Lee 05: 34

## Getting More Information: Design Document

 <p><b>PTOLEMY II</b> HETEROGENEOUS CONCURRENT MODELING AND DESIGN IN JAVA</p> <p>Edited by: Christopher Hylinski, Edward A. Lee, Xu Liu, Xiaojun Liu, Steve Neuendorffer, Yuhong Xiong, Haipeng Zhang</p> <p><b>VOLUME 1: INTRODUCTION TO PTOLEMY II</b></p> <p>Authors: Shawn S. Bhattacharya Ramesh Chandra John Davis, II Mukul Garg Bart Kienhuis Christopher Hylinski Edward A. Lee Xu Liu Xiaojun Liu Zohar Moshch Steve Neuendorffer John Reule Neil Smith Jed Tap Brian Vogel Whitney Williams Yuhong Xiong Tong Zhao Haipeng Zhang</p> <p>Department of Electrical Engineering and Computer Sciences University of California at Berkeley <a href="http://ptolamp.eecs.berkeley.edu">http://ptolamp.eecs.berkeley.edu</a></p> <p>Document Version 1.0 For use with Ptolemy II 1.0 June 8, 2003</p> <p>Memorandum UC/BERE.M0578A</p> <p>Earlier versions: • UC/BERE.M05021 • UC/BERE.M05040 • UC/BERE.M05072</p> <p>This project is supported by the Defense Advanced Research Projects Agency (DARPA), the National Science Foundation, Chou's Center for Hybrid and Embedded Software Systems, the State of California MICRO program, and the following companies: Agilent, Intel, Cadence, Alcatel, Renesas, National Semiconductor, Philips, and Wind River Systems.</p>	 <p><b>PTOLEMY II</b> HETEROGENEOUS CONCURRENT MODELING AND DESIGN IN JAVA</p> <p>Edited by: Christopher Hylinski, Edward A. Lee, Xu Liu, Xiaojun Liu, Steve Neuendorffer, Yuhong Xiong, Haipeng Zhang</p> <p><b>VOLUME 2: PTOLEMY II SOFTWARE ARCHITECTURE</b></p> <p>Authors: Shawn S. Bhattacharya Ramesh Chandra John Davis, II Mukul Garg Bart Kienhuis Christopher Hylinski Edward A. Lee Xu Liu Xiaojun Liu Zohar Moshch Steve Neuendorffer John Reule Neil Smith Jed Tap Brian Vogel Whitney Williams Yuhong Xiong Tong Zhao Haipeng Zhang</p> <p>Department of Electrical Engineering and Computer Sciences University of California at Berkeley <a href="http://ptolamp.eecs.berkeley.edu">http://ptolamp.eecs.berkeley.edu</a></p> <p>Document Version 1.0 For use with Ptolemy II 1.0 June 8, 2003</p> <p>Memorandum UC/BERE.M0578A</p> <p>Earlier versions: • UC/BERE.M05021 • UC/BERE.M05040 • UC/BERE.M05072</p> <p>This project is supported by the Defense Advanced Research Projects Agency (DARPA), the National Science Foundation, Chou's Center for Hybrid and Embedded Software Systems, the State of California MICRO program, and the following companies: Agilent, Intel, Cadence, Alcatel, Renesas, National Semiconductor, Philips, and Wind River Systems.</p>	 <p><b>PTOLEMY II</b> HETEROGENEOUS CONCURRENT MODELING AND DESIGN IN JAVA</p> <p>Edited by: Christopher Hylinski, Edward A. Lee, Xu Liu, Xiaojun Liu, Steve Neuendorffer, Yuhong Xiong, Haipeng Zhang</p> <p><b>VOLUME 3: PTOLEMY II DOMAINS</b></p> <p>Authors: Shawn S. Bhattacharya Ramesh Chandra John Davis, II Mukul Garg Bart Kienhuis Christopher Hylinski Edward A. Lee Xu Liu Xiaojun Liu Zohar Moshch Steve Neuendorffer John Reule Neil Smith Jed Tap Brian Vogel Whitney Williams Yuhong Xiong Tong Zhao Haipeng Zhang</p> <p>Department of Electrical Engineering and Computer Sciences University of California at Berkeley <a href="http://ptolamp.eecs.berkeley.edu">http://ptolamp.eecs.berkeley.edu</a></p> <p>Document Version 1.0 For use with Ptolemy II 1.0 June 8, 2003</p> <p>Memorandum UC/BERE.M0578A</p> <p>Earlier versions: • UC/BERE.M05021 • UC/BERE.M05040 • UC/BERE.M05072</p> <p>This project is supported by the Defense Advanced Research Projects Agency (DARPA), the National Science Foundation, Chou's Center for Hybrid and Embedded Software Systems, the State of California MICRO program, and the following companies: Agilent, Intel, Cadence, Alcatel, Renesas, National Semiconductor, Philips, and Wind River Systems.</p>
--	--	---

Volume 1:  
User-Oriented

Volume 2:  
Developer-Oriented

Volume 3:  
Researcher-Oriented

Lee 05: 35

## Summary

Ptolemy II provides considerable infrastructure for experimenting with models of computation.

Lee 05: 36