

Ptolemy Project Status and Overview

Edward A. Lee

Ptolemy Project Director, UC Berkeley



**6th Biennial Ptolemy
Miniconference**

**Berkeley, CA
May 12, 2005**



Software Legacy of the Project

- 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++
 - Abstract Actor Semantics
 - 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
 - Behavioral polymorphism
 - Multithreaded
 - Network integrated and distributed
 - Modal models
 - Sophisticated type system
 - CT, HDF, CI, GR, etc.

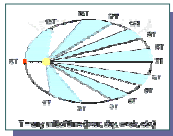
Each of these served us, first-and-foremost, as a laboratory for investigating design.

Focus has always been on embedded software.



And Most Recently...

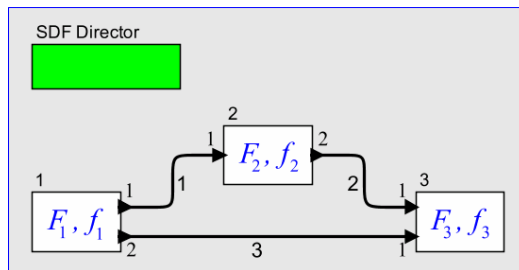
- o Kepler (2003-?)
 - Scientific workflows
 - Web services harvester
 - Computational grid integration
 - Semantic types
 - Browser interface
 - Database integration
 - "R" integration
 - Sensor data streaming
 - XML and XSLT integration
 - ...



Lee, Berkeley 3



Where it started: SDF: Synchronous Dataflow and the Balance Equations (1985-86)



production/consumption matrix

$$\Gamma = \begin{bmatrix} 1 & -1 & 0 \\ 0 & 2 & -1 \\ 2 & 0 & -1 \end{bmatrix}$$

Actor 1

Connector 1

$$q = \begin{bmatrix} q_1 \\ q_2 \\ q_3 \end{bmatrix}$$

firing vector

balance equations

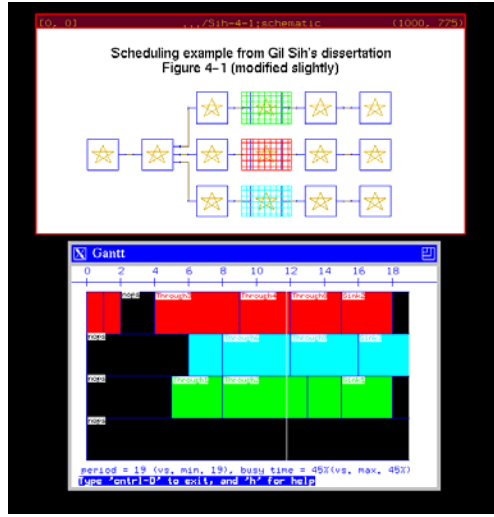
$$\Gamma q = \vec{0} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Lee, Berkeley 4



Gabriel and Ptolemy Classic Leveraged SDF to Generate Parallel Code

SDF model, parallel schedule, and synthesized DSP assembly code (1990)



```
codeblock(Std) {
  // initialize address registers for coef and
  delayLineove #saddr(coef)+$val(coefLen)-1,r3
: insert here
  move $ref(delayLineStart),r5
: delayLine
  move #sval(stepSize),x1
  move $ref(operand),x0
  mpr x0,x1,u
  move a,x0
  move x:(r3),b y:(r5)+,y0
}

codeblock(loop) {
  do #sval(loopVal), $label(endlloop)
  mopr x0,y0,b
  move b,x:(r3)-
  move x:(r3),b y:(r5)+,y0
$label(endlloop)
}

codeblock(nolloop) {
  mopr x0,y0,b
  move b,x:(r3)-
  move x:(r3),b y:(r5)+,y0
}
```

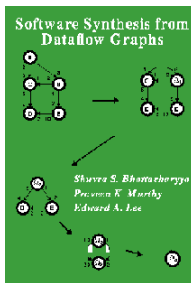
It is an interesting (and rich) research problem to minimize interlocks in complex multirate applications.

Lee, Berkeley 5



Many Scheduling and Optimization Problems (and Some Solutions) Resulted

- Optimization criteria that might be applied:
- Minimize buffer sizes.
- Minimize the number of actor activations.
- Minimize the size of the representation of the schedule (code size).



See S. S. Bhattacharyya, P. K. Murthy, and E. A. Lee, *Software Synthesis from Dataflow Graphs*, Kluwer Academic Press, 1996, for a summary of the single processor optimization problems.

Lee, Berkeley 6



Example: Heterogeneous Architecture with DSP and Sun Sparc Workstation (1995)

DSP card in a Sun Sparc Workstation runs a portion of a Ptolemy model; the other portion runs on the Sun.

The screenshot displays a Ptolemy model for FM synthesis. It includes a control panel for 'synthDisplayFFT0' with parameters like 'Number of Iterations' and 'FM_Index'. The main schematic, titled 'Chowning FM Synthesis', shows a complex signal flow involving FM synthesis blocks, multipliers, and adders. A box at the bottom right highlights the hardware components: 'Sparc C' and 'DSP Card M56K'. A keyboard is visible at the bottom of the interface.

Lee, Berkeley 9



Ptolemy Classic Example Showing Higher-Order Components (adaptive nulling in an antenna array, 1995)

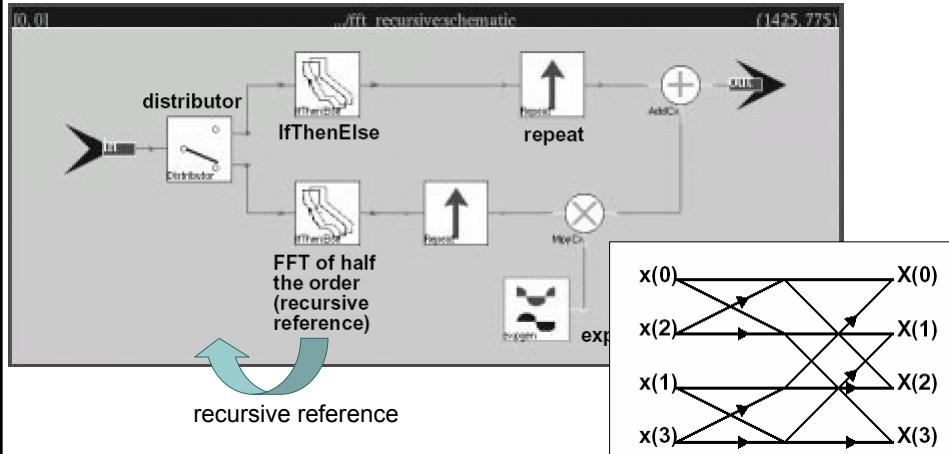
The screenshot shows a Ptolemy model for an adaptive array processor. It includes a 'Beam Pattern' plot showing a four-lobed pattern, an 'Output Signal' plot, and various signal processing blocks like 'Transmit Signal', 'Receive Antenna Array', and 'Adaptive Array Processor'. Labels highlight 'streams', 'hierarchical components', and 'higher-order components'. The title is 'An Adaptive Array Processor with a 4 Element Uniform Circular Array suppresses three Cochannel Interferers'. The Ptolemy application was developed by Uwe Trautwein, Technical University of Ilmenau, Germany.

Lee, Berkeley 10



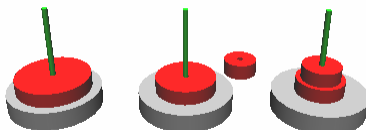
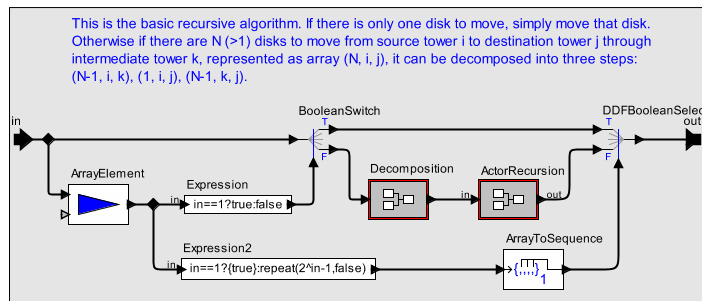
Higher-Order Components Realizing Recursion in Ptolemy Classic

FFT implementation in Ptolemy Classic (1995) used a partial evaluation strategy on higher-order components.



Higher-Order Components in Ptolemy II

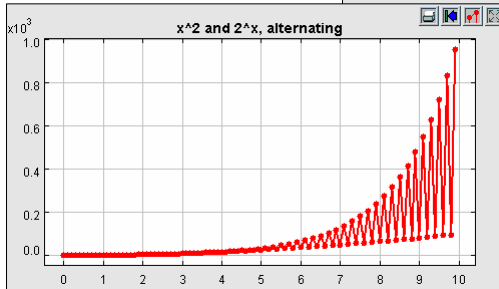
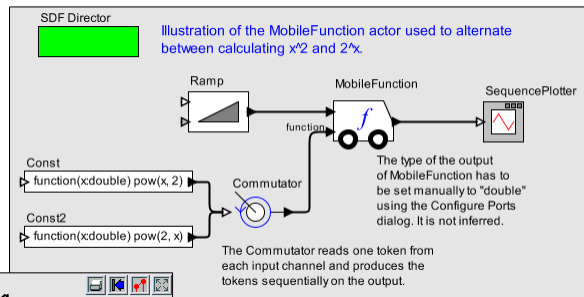
The dynamic dataflow (DDF) domain (new to Ptolemy II in v. 5.0) implements recursion in a similar way [due to Gang Zhou].





Higher-Order Expression Language in Ptolemy II

Higher-order components (actor-oriented) coupled with a higher-order expressions (functional) are a potentially powerful combination.



Neuendorffer and Zhao

Lee, Berkeley 13



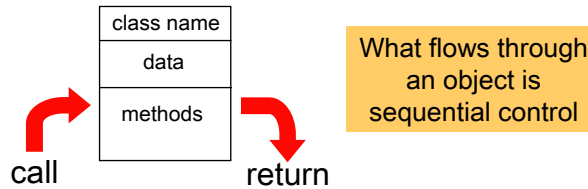
What is the Ptolemy Project Really About?

Lee, Berkeley 14

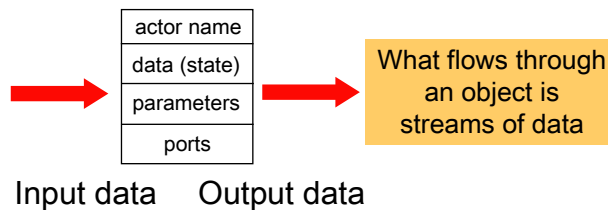


The Ptolemy Project is About Actor-Oriented Design

Object orientation:



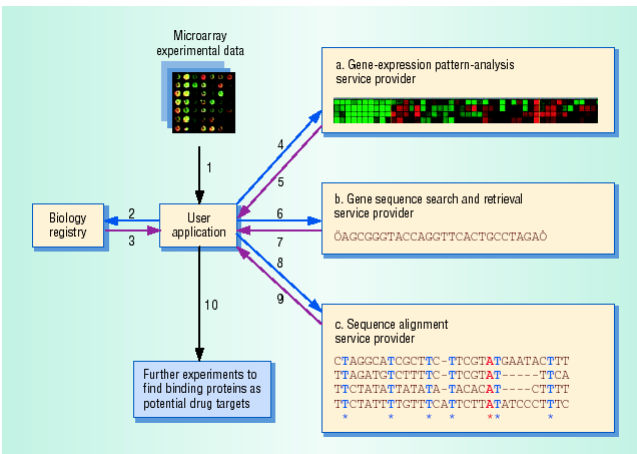
Actor orientation:



Lee, Berkeley 15



Actor-Oriented vs. Object-Oriented



The figure at the left shows the use of object-oriented web services for a “microarray data-analysis scenario for identifying targets in drug discovery.” The authors explain, “the numbered lines are the steps in the analysis path.”

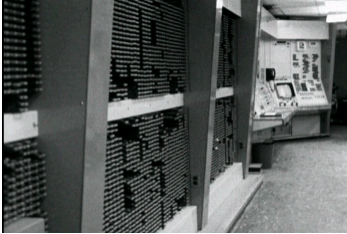
From Gao & Hayes, “Integrating Biological Research through Web Services,” *Computer*, March, 2005.

Lee, Berkeley 16



The First (?) Actor-Oriented Platform

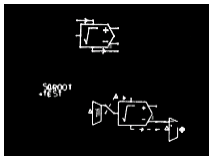
The On-Line Graphical Specification of Computer Procedures
W. R. Sutherland, Ph.D. Thesis, MIT, 1966



MIT Lincoln Labs TX-2 Computer



Bert Sutherland with a light pen



Bert Sutherland used the first acknowledged object-oriented framework (Sketchpad, created by his brother, Ivan Sutherland) to create the first actor-oriented programming framework.

Partially constructed actor-oriented model with a class definition (top) and instance (below).

Lee, Berkeley 17



Your Speaker in 1966



Lee, Berkeley 18



Modern Examples of Actor-Oriented Platforms

- Simulink (The MathWorks)
- LabVIEW (National Instruments)
- Modelica (Linköping)
- OPNET (Opnet Technologies)
- Giotto and xGiotto (UC Berkeley)
- Polis & Metropolis (UC Berkeley)
- Gabriel, Ptolemy, and Ptolemy II (UC Berkeley)
- OCP, open control platform (Boeing)
- GME, actor-oriented meta-modeling (Vanderbilt)
- SPW, signal processing worksystem (Cadence)
- System studio (Synopsys)
- ROOM, real-time object-oriented modeling (Rational)
- Easy5 (Boeing)
- Port-based objects (U of Maryland)
- I/O automata (MIT)
- VHDL, Verilog, SystemC (Various)
- ...

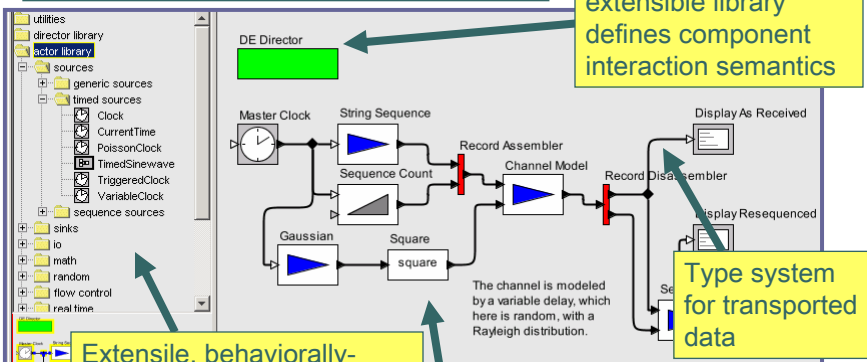
Lee, Berkeley 19



Ptolemy II: Our Laboratory for Actor-Oriented Models of Computation

Concurrency management supporting dynamic model structure.

Director from an extensible library defines component interaction semantics



Extensible, behaviorally-polymorphic component library.

Type system for transported data

Visual editor supporting an abstract syntax

Lee, Berkeley 20



Models of Computation Implemented in Ptolemy II

- CI – Push/pull component interaction
- Click – Push/pull with method invocation
- CSP – concurrent threads with rendezvous
- CT – continuous-time modeling
- DDF – Dynamic dataflow
- DE – discrete-event systems
- DDE – distributed discrete events
- DPN – distributed process networks
- FSM – finite state machines
- DT – discrete time (cycle driven)
- Giotto – synchronous periodic
- GR – 2-D and 3-D graphics
- PN – process networks
- SDF – synchronous dataflow
- SR – synchronous/reactive
- TM – timed multitasking

Most of
these are
actor
oriented.

Lee, Berkeley 21



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

“Domains” are extensions built on the core infrastructure.

Lee, Berkeley 22

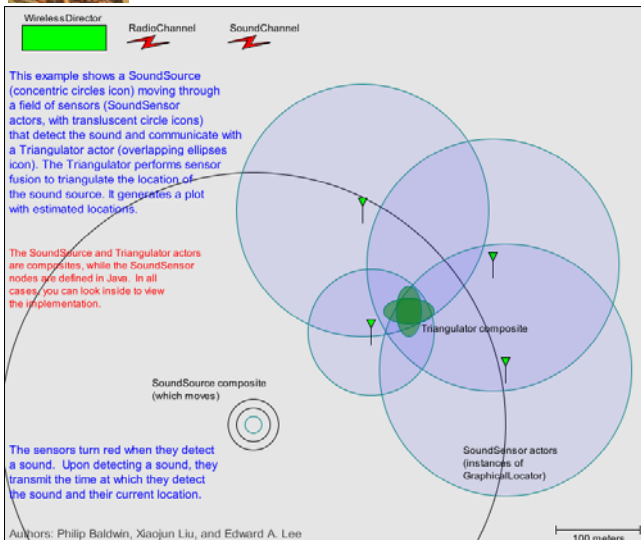


What have we done recently?

Lee, Berkeley 23



Modeling and Design of Wireless Networked Systems



VisualSense: Modeling of wireless sensor networks as an extension of DE. [Baldwin, Kohli, Liu, Zhao]

VIPTOS: Design of software for wireless sensor network motes in TinyOS/nesc. [Cheong, coming soon]

Lee, Berkeley 24



Actor-Oriented Type Systems Classes, Subclasses, and Inheritance

This model illustrates the mechanisms in Ptolemy II for defining classes and subclasses with inheritance.

NoisySineWave This actor is a class definition, indicated by the blue halo. It is ignored by the director, and serves as a declaration. To create an instance of this class, right click on the class definition and select "Create Instance" (or type Ctrl-N). To see the class definition, look inside.

SineWave This is an instance of the above class definition. Look inside to see the subclass definition.

InstanceOfNoisySineWave This is an instance of the base class for the above class definition.

SequencePlotter

SDF Director Generate a sine wave.
frequency: 440.0
phase: 0.0

Clean and Noisy Sine Wave (Graph showing sample number vs amplitude)

inherited actors (Ramp, Const, AddSubtract, TrigFunction)

override actors (Gaussian, AddSubtract2, noisy)

subclass

The objects highlighted in pink are defined in the superclass. Such objects cannot be removed in this derived class. Their parameters can be changed, however. This implies that they can be moved and can be assigned custom icons. To examine the superclass, right click on the background and select "Open Base Class".

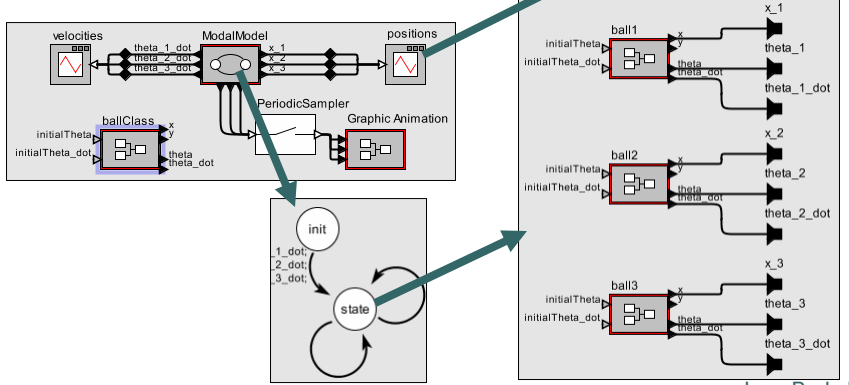
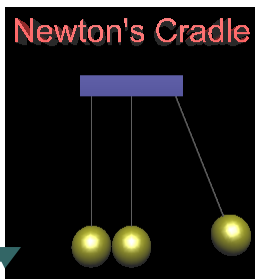
This type system builds on abstract syntax (not semantics) so it applies very broadly to actor-oriented models, including hybrid systems.

Lee, Berkeley 25



Semantics

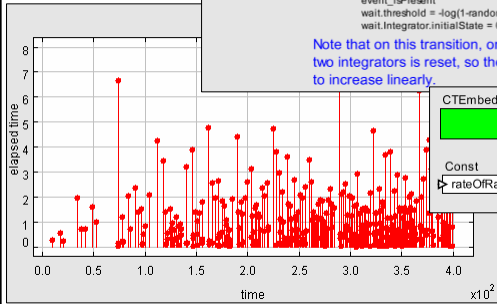
Clean, clear, and rigorous semantics for discrete-event, continuous-time, and hybrid systems [Cataldo, Liu, Matsikoudis, Zheng]



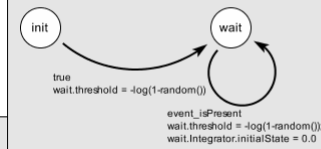


Stochastic Hybrid Systems

Stochastic hybrid systems in Ptolemy II are Monte-Carlo models of nondeterminism



On each transition, generate a new random number with an exponential distribution. In the "wait" state, wait an amount of time that is the value of this random variable multiplied by the current (increasing) rate.

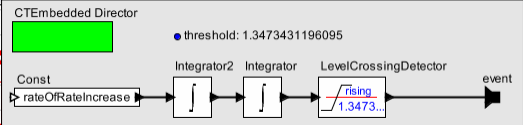


Note that on this transition, only one of the two integrators is reset, so the rate continues to increase linearly.

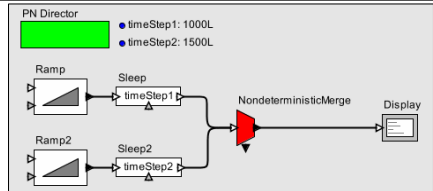
CT Director
 rateOfRateIncrease: 0.005

This model generates a Poisson process with a linearly increasing rate using the CT domain. The model plots the events vs. time and a histogram of the time between events. The technique here was suggested by John Lygeros.

Example of random "spontaneous transitions" by Lee and Zheng, based on suggestion by John Lygeros.



Other Key Results



- Reconfiguration analysis [Neuendorffer]
- Generalized dependency analysis [Neuendorffer, Zheng]
- The Cal actor language [Eker and Janneck]
- Java code generation [Neuendorffer]
- Modal model semantics [Liu, Zhou (Rachel)]
- Mixed procedural and event semantics [Cheong]
- Unbounded time, controlled precision [Zheng]
- Nondeterministic merge in PN [Lee, Xiaowen Xin (LLNL)]
- Giotto + Ptolemy II package [Brooks]
- Communications library [Zhou (Rachel)]
- Image and video library [Yeh]
- Scratchpad memory management from SDF [Kohli]



Acknowledgements

- Current students
 - Adam Cataldo
 - Elaine Cheong
 - Thomas Huining Feng
 - Xiaojun Liu
 - Eleftherios Matsikoudis
 - Yang Zhao
 - Haiyang Zheng
 - Gang Zhou
 - Rachel Zhou
- Staff
 - Christopher Brooks
 - Mary Margaret Sprinkle
 - Mary Stewart
- Recent PhD graduates
 - Steve Neuendorffer (Xilinx)
 - Yuhong Xiong (HP Labs)
- Recent Postdocs
 - Jörn Janneck (Xilinx)
- Recent masters graduates
 - Vinay Krishnan
 - Sanjeev Kohli
 - James Yeh
- Current sponsors
 - Agilent
 - Hewlett-Packard
 - Escher Institute
 - National Science Foundation
 - Toyota