

The Future of the Ptolemy Project



Edward A. Lee
UC Berkeley

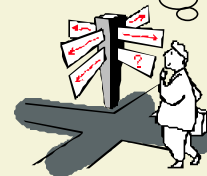
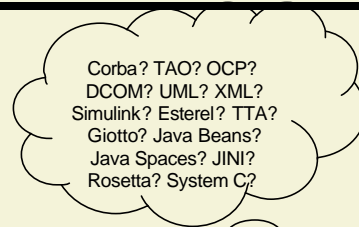
With thanks to the entire Ptolemy Team.

Ptolemy Miniconference
Berkeley, CA, March 22-23, 2001

The Problem

- Composition
- Decomposition

How do components interact?



Approaches



- Do it my way
- Do it your way
- Do it the right way



Claim: The right way is often neither your way nor my way.

Ptolemy Miniconference, Berkeley, 3

Do It My Way



- **“I have an xxx for you.”**
 - xxx = language
 - xxx = OO framework
 - xxx = model of computation
 - xxx = tool
- **“It’s really cool. It gives you yyy.”**
 - yyy = a really cool GUI
 - yyy = approval of your program manager
 - yyy = expressiveness
 - yyy = formal verification



Ptolemy Miniconference, Berkeley, 4

Obstacles



- **Most engineers resist new languages**
 - Can be tough to learn new tricks
 - Not used to thinking that way
 - The way they did the last project was just fine
 - New syntaxes are particularly scary
 - New languages are more specialized than claimed

- **Requires rethinking applications**
 - Most of the tough decisions are at the MoC level
 - Consider for example
 - a real-time O/S application redone in Esterel
 - a Matlab script redone in real-time embedded software

Ptolemy Miniconference, Berkeley, 5

Overcoming the Obstacles



- **Fool the people**
 - Masquerade as a familiar language
 - “do system-level design in C++!”
 - e.g. System C
 - Claim that your approach “avoids programming”
 - Use a block-diagram syntax
 - e.g. Simulink

- **Gentle persuasion**
 - Build a really cool GUI

Ptolemy Miniconference, Berkeley, 6

Do It Your Way



- Do whatever you want, wherever you want
- Build infinite flexibility into the tools
- If your design is incomprehensible, then it serves you right for not doing it my way

This approach is incremental, as small changes are made to previous methodologies. Thus, embedded software will remain a problem of tweaking priorities in a real-time OS, and battling priority inversion.



Ptolemy Miniconference, Berkeley, 7

Do it the Right Way



- **Requires thinking about designs at a meta level**
 - What model(s) of computation should be used?
 - CSP? Publish/Subscribe? Dataflow? Statecharts?
 - Requires that designers understand the alternatives...
 - How to partition the design
 - Which model of computation to use in which partition?
- **Requires rethinking tools and languages**
 - Cannot assume they are “it” (interoperability!)
 - How to mix models of computation?
 - Real designs are complex and heterogeneous



Ptolemy Miniconference, Berkeley, 8

Example



- Hierarchical nesting of
 - Time driven architecture for real-time
 - Priority-driven publish & subscribe for fault handling & reconfig.
 - Synchronous model for user interface & control logic
 - Dataflow model for signal processing
 - ODEs for continuous dynamics
 - Rendezvous for resource management

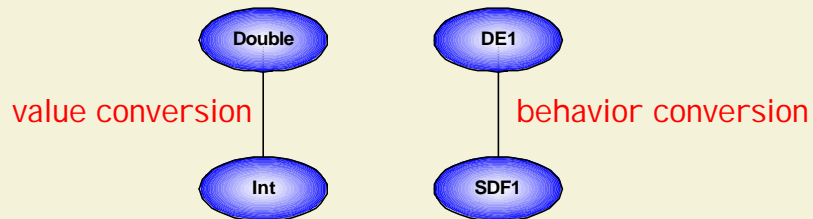
The research challenge: How do we get these modeling strategies to work together in an understandable way? How to make them efficient?

Ptolemy Miniconference, Berkeley, 9

Making Sense of the Options: Component Interfaces



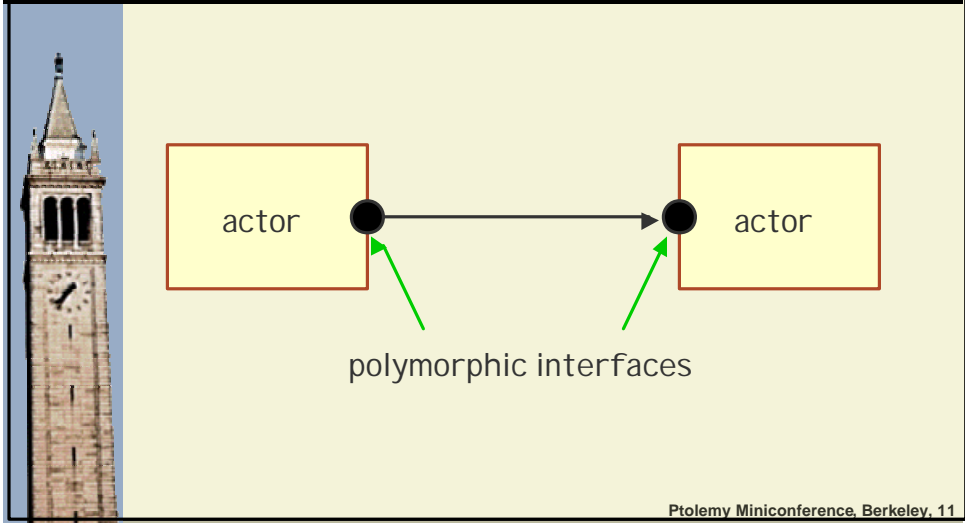
- Represent not just data types, but interaction types as well.



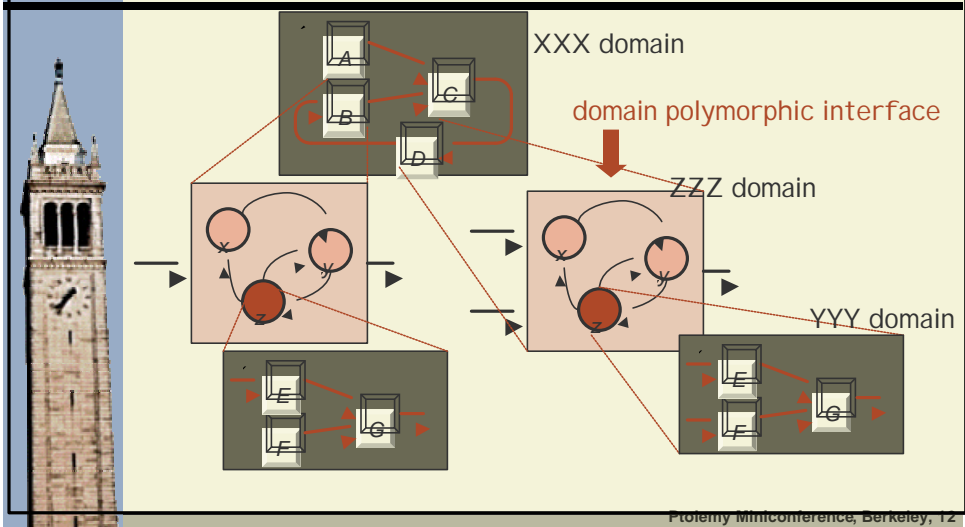
- Resulting “system-level type system” can be used to define “domain polymorphic” components

Ptolemy Miniconference, Berkeley, 10

Our Goal – Domain Polymorphic Interfaces



Hierarchical Heterogeneity: H^2



Domain Exploration

Exploration of models of computation is a major part of what we do.

- BDF/IDF – boolean/integer dataflow (Buck)
- DDF – dynamic dataflow with threaded actors
- HDF – heterochronous dataflow
- IA – interface automata (de Alfaro)
- PDF – parameterized dataflow (Bhattacharyya)
- RTOS – priority-driven real-time scheduling
- SR – synchronous/reactive

Blue domains in progress

Ptolemy Miniconference, Berkeley, 13

Getting Efficiency: Code Generators

Code generator produces code from block diagrams.

The screenshot displays the Ptolemy code generator interface. On the left, a block diagram shows a dataflow graph with a green box labeled 'SDF0', a blue trapezoidal block labeled 'Ramp1', and a blue trapezoidal block labeled 'Display2'. A menu bar at the top includes 'File', 'View', 'Edit', 'Graph', and 'Help'. A 'Code Generator' menu item is highlighted. In the foreground, a dialog box titled 'file:/C:/users/eal/talks/01/codeGeneratorDemo.xml' is open. It contains a 'Generate' button and a 'Cancel' button. Below the buttons, there are several fields: 'deep:' with a radio button, 'show:' with a radio button, 'compile:' with a radio button, 'run:' with a radio button, 'directory:' with the value 'C:\pt\ptolemy\domains\rtos', 'compileOptions:' with the value '{classpath "ptli,"}', 'runOptions:' with the value '{classpath "ptli,"}', and 'packageName:' with an empty field. At the bottom of the dialog, there is a text area showing the execution command: 'Executing: javac -classpath "/ptli;" C:\pt\ptolemy\domains\rtos\codeGeneratorDemo.java' and 'Executing: java -classpath "/ptli;" ptolemy.actor.gui.CompositeActorApplication -class codeGenera ptolemy.actor.Manager run(): elapsed time: 6079 ms'. On the right side of the dialog, there is a 'More Info' button. The background shows a window titled 'file:/C:/users/eal/talks/01/codeGeneratorDemo.xml' with a menu bar and a 'Code Generator' menu item. An orange arrow points from the 'Code Generator' menu item to the dialog box. Another orange arrow points from the dialog box to the block diagram.

Ptolemy Miniconference, Berkeley, 14

Code Generator Tasks



■ Back end languages:

- C
- C++
- Embedded Java
- VHDL, ...

■ Specific targets:

- TTA
- VxWorks
- FPGAs, ...

■ Domains

- FSM
- RTOS
- Giotto
- PDF, ...

There is enough work here to keep many people busy for a long time...

Ptolemy Miniconference, Berkeley, 15

Improving the Expressiveness of Visual Syntaxes



■ Higher-Order Functions

- Token whose value is a model
 - An actor with ports and parameters
- Parameter values can be models
- Apply actor with “model” parameter
- Apply actor with “model” input port
- Map combinator actor
- Zip combinator actor ...

■ Meta modeling

- Generation of customized visual syntaxes
- Modeling domains in Ptolemy II
- Modeling Ptolemy II in Ptolemy II

Ptolemy Miniconference, Berkeley, 16

Framework Tasks



- **Community building**
 - PtNapster for actor libraries?
 - Domain community
- **ePtolemy**
 - CORBA-based distributed modeling
 - IP-protected libraries
 - Parallelized simulation: Ptolemy @ home screensavers?
- **Support**
 - ptolemy-hackers@ptolemy.eecs.berkeley.edu
 - comp.soft-sys.ptolemy newsgroup
 - Agile Design, Inc.



Ptolemy Miniconference, Berkeley, 17

Conclusion



Stay tuned.



Ptolemy Miniconference, Berkeley, 18