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

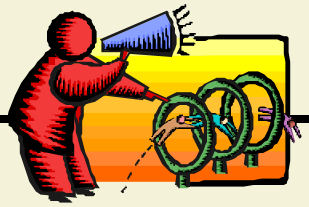
How do components interact?

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.

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

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

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

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.

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

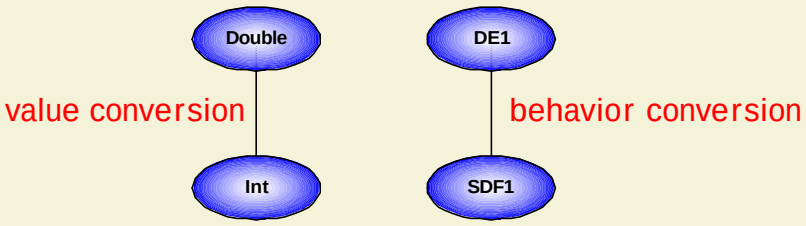
Example

- 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?

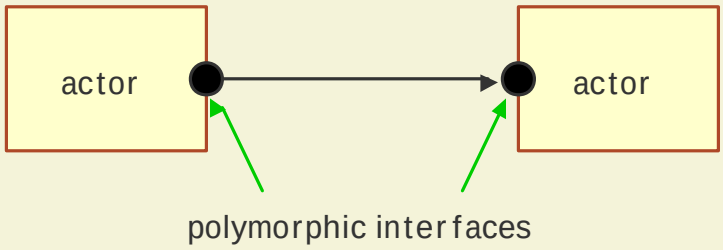
Making Sense of the Options: Component Interfaces

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

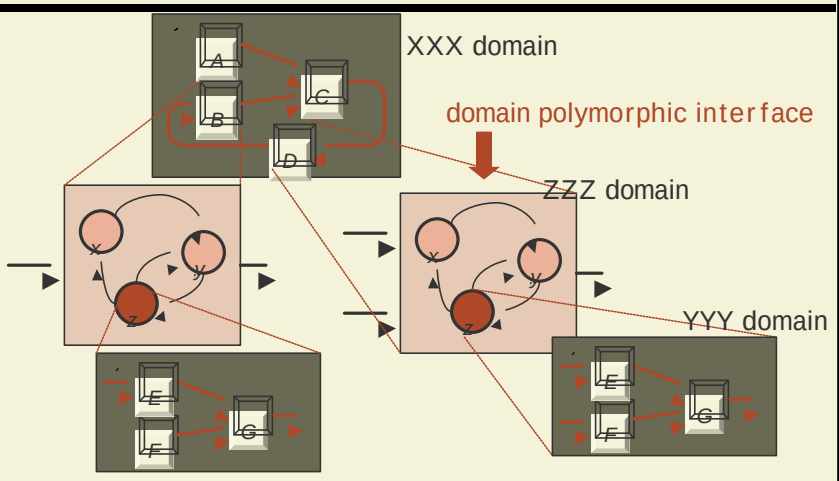


- Resulting “system-level type system” can be used to define “domain polymorphic” components

Our Goal – Domain Polymorphic Interfaces



Hierarchical Heterogeneity: H^2



Domain Exploration

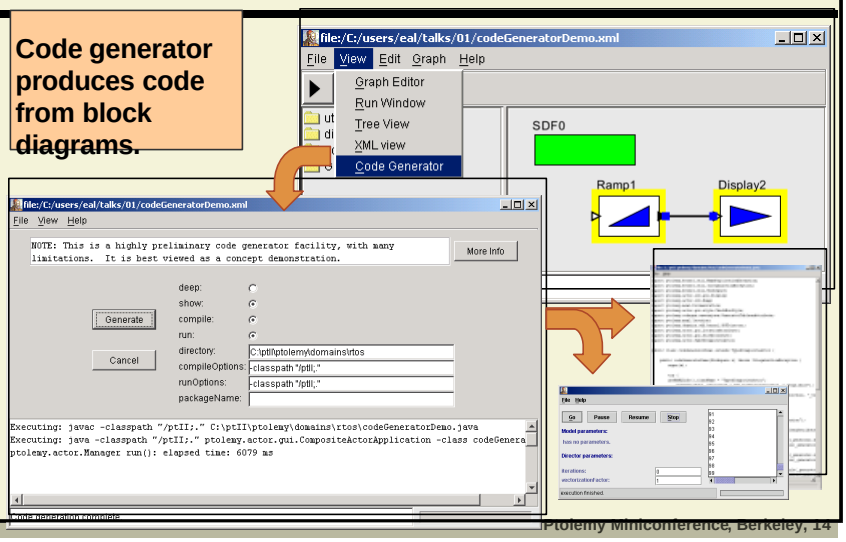
Exploration of models of computation is a major part of what we do.

- 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

Getting Efficiency: Code Generators

Code generator produces code from block diagrams.



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…

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

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.

Conclusion

Stay tuned.