Heterogeneous Modeling and Design in Ptolemy II

Johan Eker
UC Berkeley

with material courtesy of Edward Lee and the Ptolemy group

ECE Seminar Series, Carnegie Mellon, November 29, 2001

Ptolemy II A Software Laboratory

Ptolemy II

- Java based
- Graphical modeling and simulation environment
- Multiple "models of computation"
- Hierarchical & heterogeneous models
- Code generator
- Actor language

Outline

- Introduction
- Ptolemy II basics
- A motivating example
- Research Issues
- Summary

Embedded Systems

- Computers not thought of as computers
- Increasingly complex designs
  - networked, fail safe, etc
- Development speed
  - time to market
- Bugs, bugs, bugs
  - hard to correct a released product
  - don't want to reboot your toaster

What is So Different With Embedded Software?

- Interaction with physical processes
  - sensors, actuators, processes
- Critical properties are not all functional
  - real-time, fault recovery, power, security, robustness
- Heterogeneous
  - hardware/software, mixed architectures
- Concurrent
  - interaction with multiple processes
- Reactive
  - operating at the speed of the environment

Component Technology

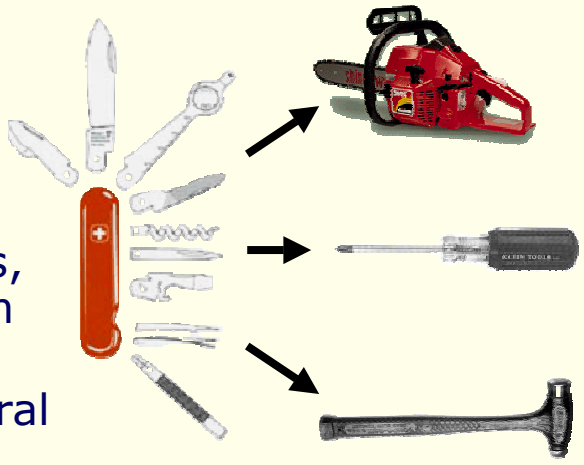
- Examples: Java beans, VB-components, etc
- Rationale
  - Encapsulation
  - Reuse
  - Divide complexity
- Successful in many areas
- Problems with concurrent components
  - Threads are not components
  - Priorities are global parameters
- Difficult to design embedded systems component with state-of-the art technology

Multipurpose tools

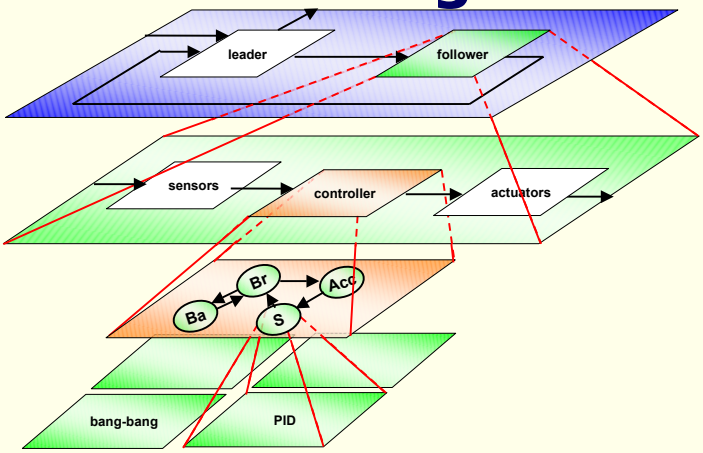
- Express almost anything, guarantee almost nothing
- You only need to know *one* programming language
  - Quick starts, but sometimes slower endings
- Programmers+language, a lifelong marriage
- Examples:
  - Java
  - C/C++ with RTOS, ADA, Modula-2
  - RMA & EDF scheduling

Sharpen your tools

- Use problem specific tools
  - Constrain the solutions
- Choice of tools, a major design decision
- Combine several tools



Hierarchical, Heterogeneous Modeling and Design



Ptolemy Background

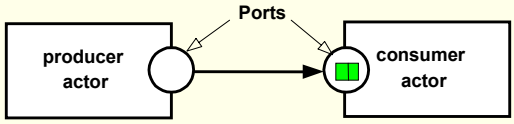
- Initially a signal processing tool
- Gabriel (Lisp) 1985-1990
- Ptolemy Classic (C++) 1990-1997
- Ptolemy II (Java) 1996-

Claudius Ptolemy

Edward A. Lee

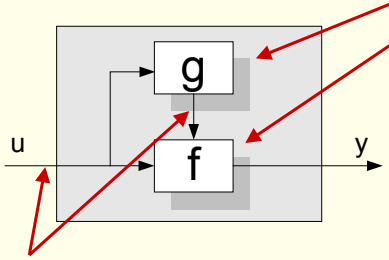
Ptolemy II Basics

- A model is a a set of interconnected *actors* and one *director*
- Actor
  - Input & output *ports*, states, & parameters
  - *Atomic* or *composite*
  - Communicates using *tokens*
  - When it is *fired* it produces and consumes tokens



Component Interaction Semantics

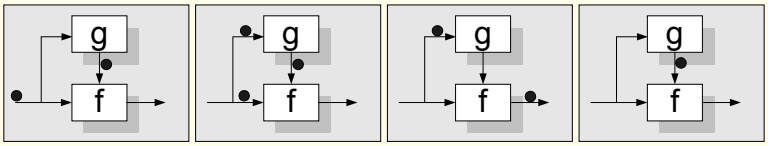
Are actors active? passive?
How is the flow of control determined?



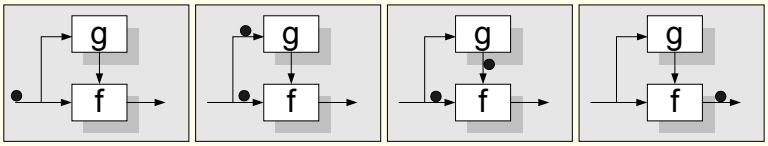
Are communications timed? synchronized? buffered?
How is the communications mediated?

Interaction Semantics
3 Different Interpretations

- Continuous time: $y(t)=f(g(u(t), u(t))$
- Discrete time : {f, g} $\Rightarrow y(k)=f(g(u(k-1)), u(k))$



- Discrete time : {g, f} $\Rightarrow y(k)=f(g(u(k), u(k))$

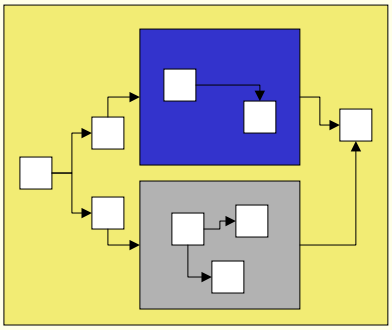


Ptolemy II Basics

- Director
  - Manages the data flow and the scheduling of the actors
  - The director fires the actors
- Receiver
  - Defines the semantics of the port buffers
- Models of Computation
  - Define the interaction semantics
  - Implemented in Ptolemy II by a *domain*
    - Director + Receiver

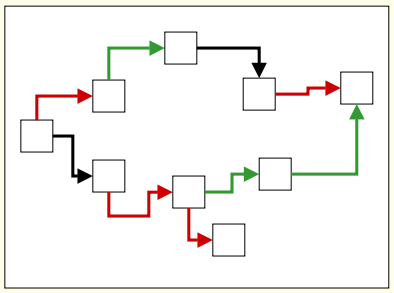
Hierarchical Heterogeneity vs. Amorphous Heterogeneity

Hierarchical



Color is a domain, which defines both the flow of control and interaction protocols.

Amorphous



Color is a communication protocol only, which interacts in unpredictable ways with the flow of control.

Available Domains

- CSP – concurrent threads with rendezvous
- CT – continuous-time modeling
- DE – discrete-event systems
- DT – discrete time
- PN – process networks
- PN' – Petri nets
- SDF – synchronous dataflow
- SR – synchronous/reactive
- GR – Graphics, 3D animations

Each is realized as a director and a receiver class in Ptolemy II

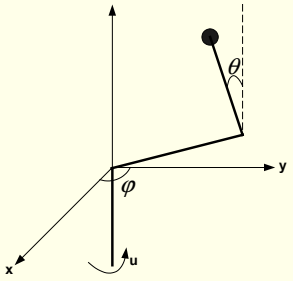
Examples of Actors+Ports Software Architectures

- Simulink (The MathWorks)
- Labview (National Instruments)
- Port-based objects (CMU/U of Maryland)
- SPW, signal processing worksystem (Cadence)
- System studio (Synopsys)
- ROOM, real-time object-oriented modeling (Rational)
- Polis & Metropolis (UC Berkeley)
- VHDL, Verilog, SystemC (Various)
- …

An Example: Controlling the Furuta Pendulum

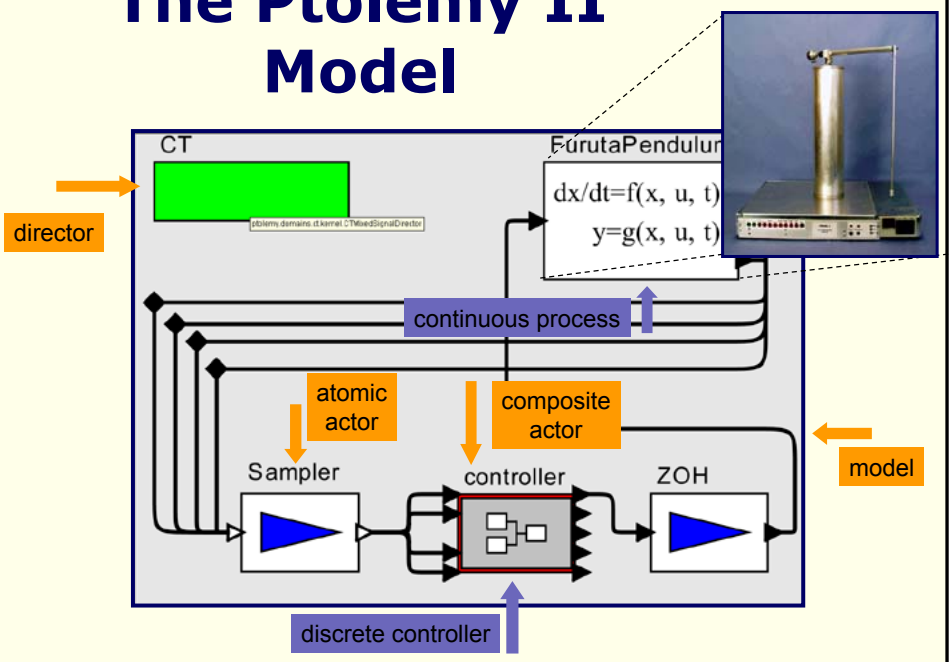
- Classic control problem
- Swing up the pendulum and then keep it in the upright position

The Example System



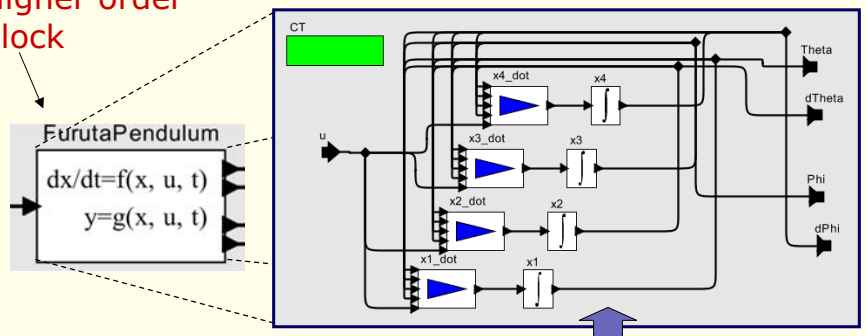
- Four states (all measurable)
  - the pendulum angle θ,
  - and its velocity $\dot{\theta}$
  - the arm angle ϕ,
  - and its velocity $\dot{\phi}$
- Input signal u is the torque on the arm
- Starts in the downright position
- Use three subcontrollers:
  - to swing it up (energy based approach)
  - to catch it (linear state feedback)
  - to stabilize it (linear state feedback)

The Ptolemy II Model



Pendulum dynamics in CT – Continuous Time

Higher order block



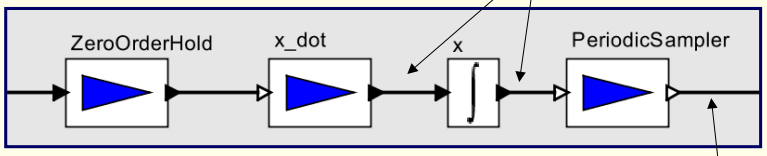
$$\left(J_p + Ml^2\right)\ddot{\theta} - Mrl\ddot{\phi}\sin\theta\cos\theta + Mrl\dot{\phi}^2\cos\theta - gl\left(M + \tfrac{m}{2}\right)\sin\theta = 0$$

$$Mrl\ddot{\theta}\cos\theta - Mrl\dot{\theta}^2\sin\theta + 2\left(J_p + ml^2\right)\dot{\theta}\dot{\phi}\sin\theta\cos\theta$$

$$+\left(J + mr^2 + Mr^2 + (Jp + ml^2)\sin^2\theta\right)\ddot{\phi} = u$$

CT Domain

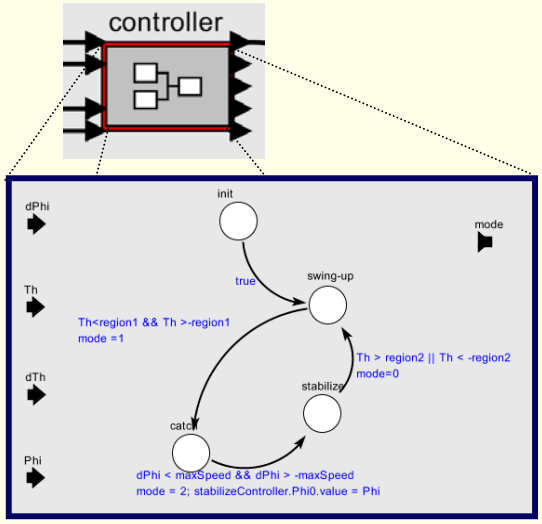
- The CT domain models components
  - interacting by continuous signals
  - described by ODE
  - network of integrators



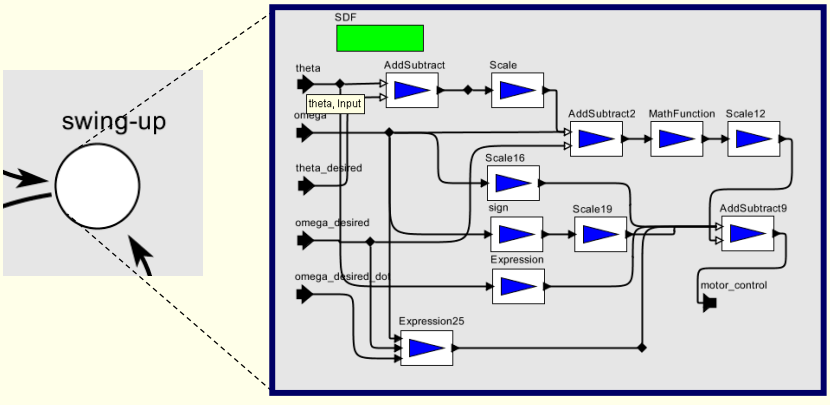
- Strengths
  - Accurate model for many physical systems
  - Established and mature simulation techniques
- Weaknesses
  - Covers a narrow application domain
  - Relatively expensive to simulate
  - Difficult to implement in software

Controller Logic in FSM - Finite State Machine

- States
  - initial
  - refinements
- Transitions
  - Guards
  - Assignments
- Natural way to express modal behavior
- Verification



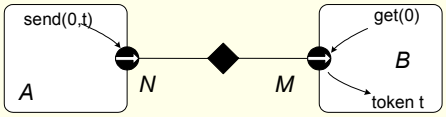
Subcontrollers in SDF - Synchronous Data flow



SDF Domain

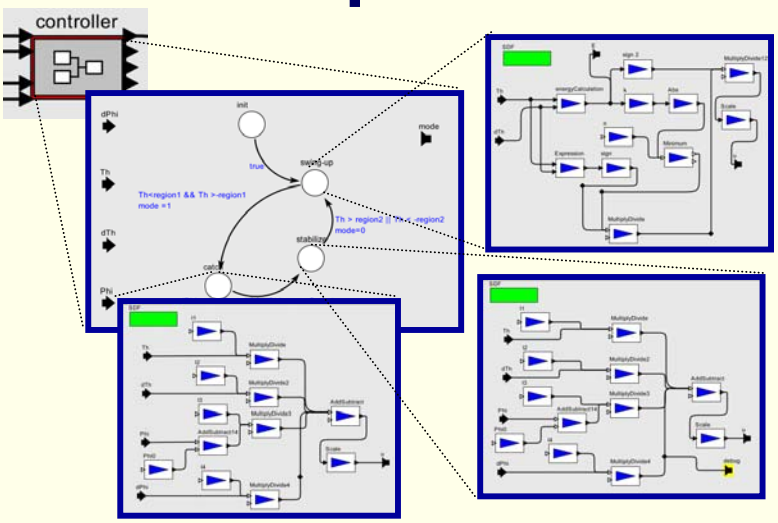
- Requires constant consumption and productions rates
- Balance equations:

  $F_A N = F_B M$



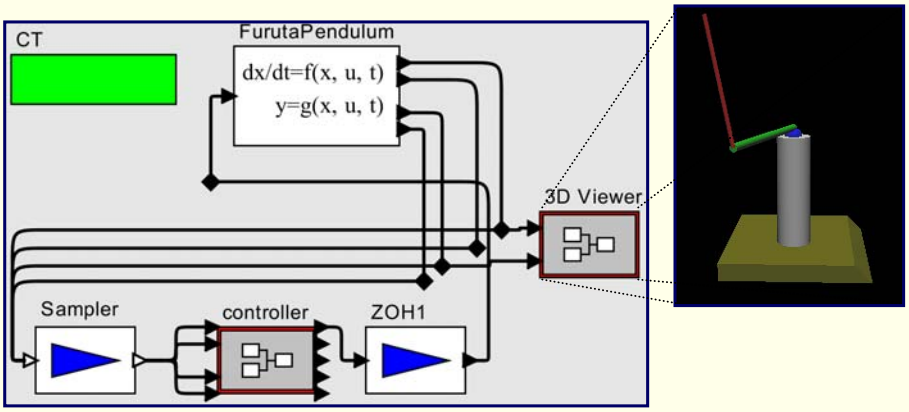
- Is statically schedulable
- Decidable resource requirements
- *Adding appropriate restrictions, increases freedom*

The Complete Controller

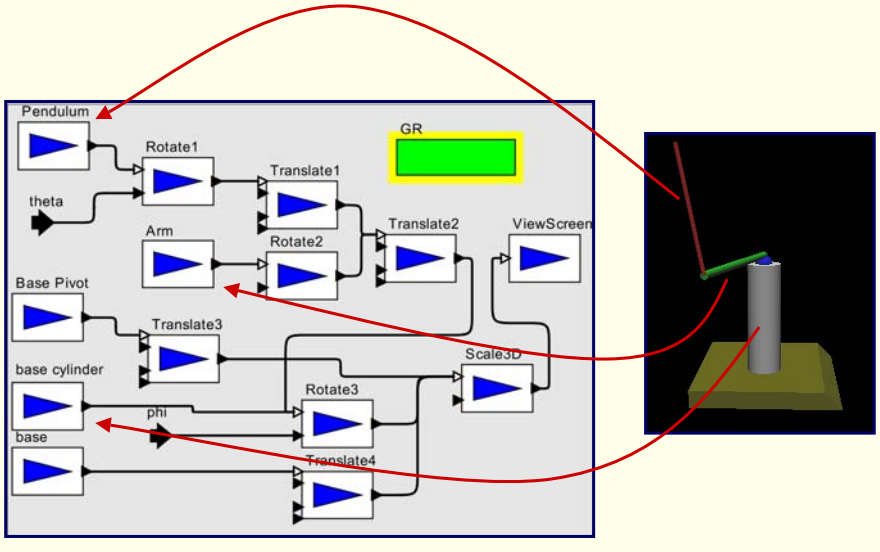


Hierarchical and heterogeneous

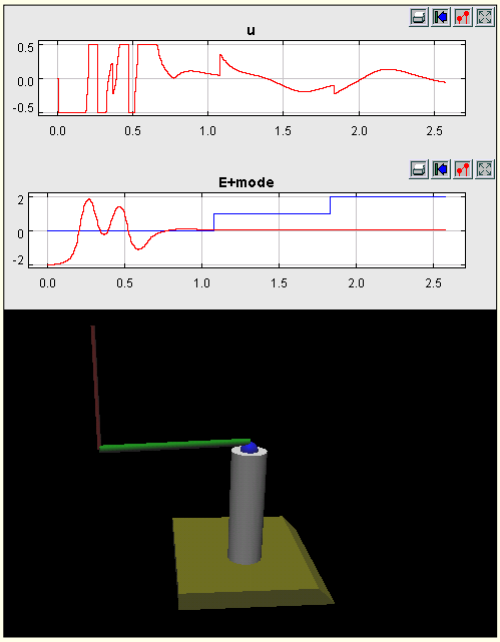
3D Visualization in GR - Graphics Domain



GR Domain



Execution [demo]



Current Research Issues

- The *Caltrop* actor language
  - Find a more concise actor description
- Code generation
  - Compile hierarchical model
- System level types
  - Go beyond data type checking
  - Extend into dynamic behavior

Summary

- Domain semantics defines
  - flow of control across actors
  - communication protocols between actors
  - implemented with directors & receivers
- Actors define:
  - functionality of components
- Hierarchy:
  - Aggregation not just syntactical
  - Composite actors are opaque, i.e. they look like atomic actors
  - Multiple domains may be used in the same model

Conclusion

- Embedded system components
- Realized in the Ptolemy II framework
- Modeling, simulation & code generation
- More information
  - Edward Lee "What's Ahead for Embedded Computing?", IEEE Computer, Sept. 2000
  - **http://ptolemy.eecs.berkeley.edu**
- Thanks to: Edward Lee, Yuhong Xiong, Jie Liu, Jörn Janneck, Steve Neuendorffer, Xiaojun Liu

THE
END