Overview of Ptolemy II

Edward A. Lee
Professor
UC Berkeley

October 9, 2003

Ptolemy Project Participants

Director:
- Edward A. Lee

Staff:
- Christopher Hylands
- Susan Gardner (Chess)
- Nuala Mansard
- Mary P. Stewart
- Neil E. Turner (Chess)
- Lea Turpin (Chess)

Postdocs, Etc.:
- Joern Janneck, Postdoc
- Rowland R. Johnson, Visiting Scholar
- Kees Vissers, Visiting Industrial Fellow
- Daniel Lázaro Cuadrado, Visiting Scholar

Graduate Students:
- J. Adam Cataldo
- Chris Chang
- Elaine Cheong
- Sanjeev Kohli
- Xiaojun Liu
- Eleftherios D. Matsikoudis
- Stephen Neuendorffer
- James Yeh
- Yang Zhao
- Haiyang Zheng
- Rachel Zhou

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++
  - 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.

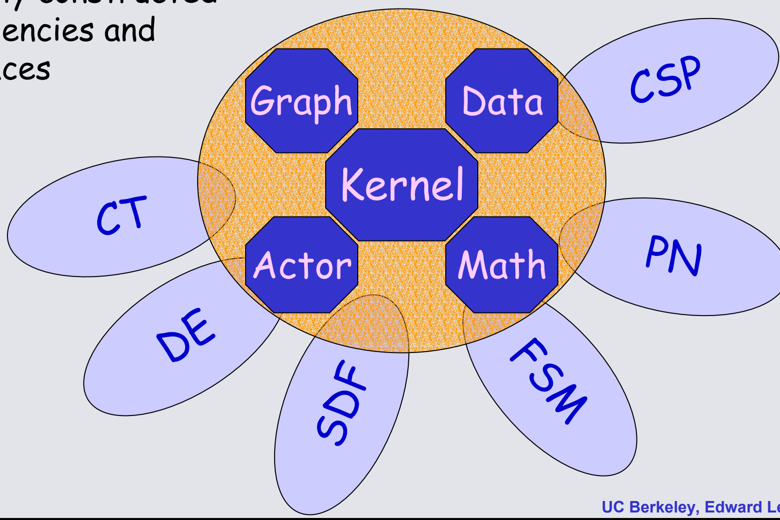
Each of these served us, first-and-foremost, as a laboratory for investigating design.

- 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).

Layered Software Architecture

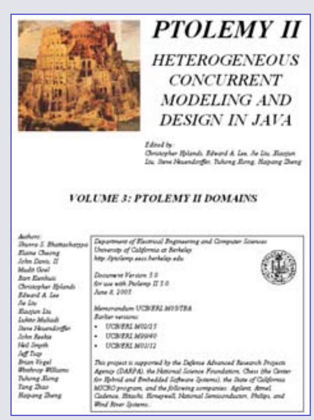
Ptolemy II packages have carefully constructed dependencies and interfaces



Design Document


PTOLEMY II
HETEROGENEOUS CONCURRENT MODELING AND DESIGN IN JAVA
VOLUME 1: INTRODUCTION TO PTOLEMY II


PTOLEMY II
HETEROGENEOUS CONCURRENT MODELING AND DESIGN IN JAVA
VOLUME 2: PTOLEMY II SOFTWARE ARCHITECTURE

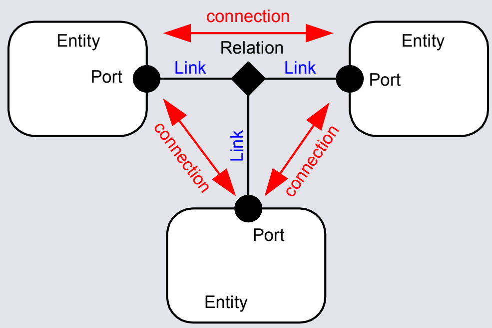

PTOLEMY II
HETEROGENEOUS CONCURRENT MODELING AND DESIGN IN JAVA
VOLUME 3: PTOLEMY II DOMAINS

Volume 1:
User-Oriented

Volume 2:
Developer-Oriented

Volume 3:
Researcher-Oriented

Kernel – Abstract Syntax
Entities, Ports, Relations and Attributes



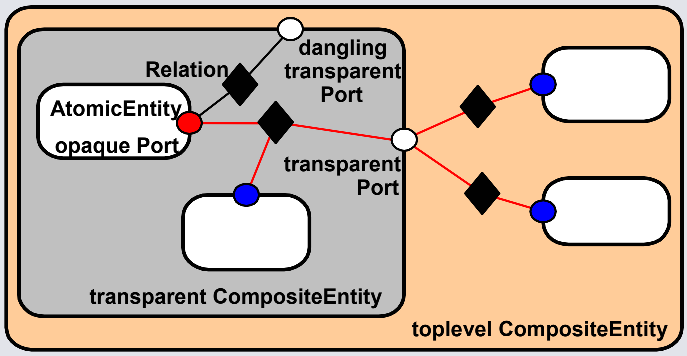
The Ptolemy II kernel provides an *abstract syntax* - clustered graphs - that is well suited to a wide variety of component-based modeling strategies, ranging from state machines to process networks.

MoML
XML Schema for this Abstract Syntax

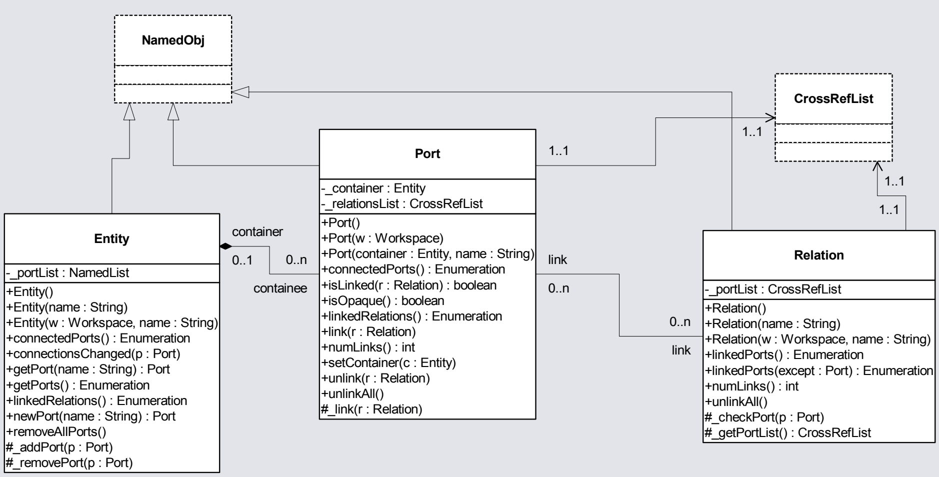
Ptolemy II designs are represented in XML:

```
...
<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"/>
...
```

Hierarchy
Composite Components

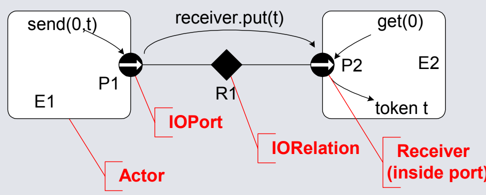


Kernel Classes
Support the Abstract Syntax



Actor Package
Supports Producer/Consumer Components

Basic Transport:

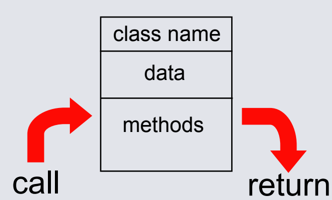


Services in the Infrastructure:

- **broadcast**
- **multicast**
- **busses**
- **mutations**
- **clustering**
- **parameterization**
- **typing**
- **polymorphism**

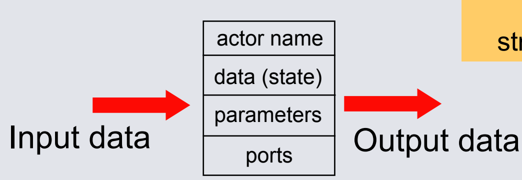
Focus on Actor-Oriented Design

- Object orientation:



What flows through an object is sequential control

- Actor orientation:

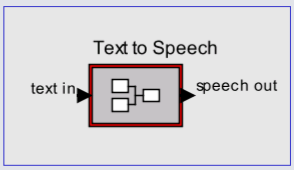


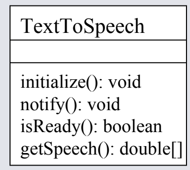
What flows through an object is streams of data

Actor-Oriented vs. Object-Oriented Interface Definitions

Actor Oriented

Object Oriented





actor-oriented interface definition says "Give me text and I'll give you speech"

OO interface definition gives procedures that have to be invoked in an order not specified as part of the interface definition.

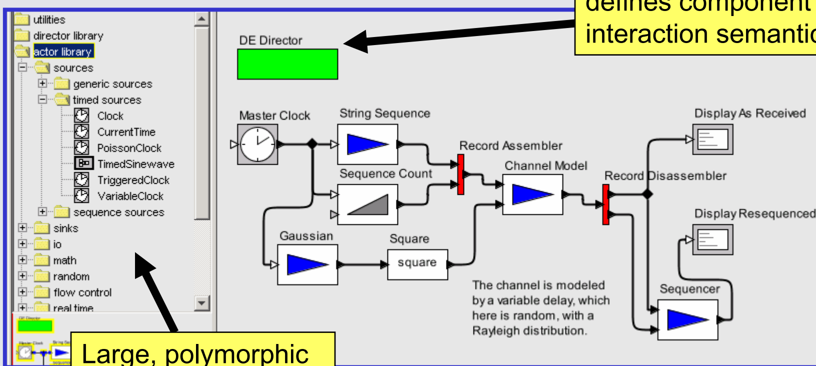
Examples of Actor-Oriented Component Frameworks

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

Ptolemy Project Principles

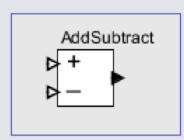
Basic Ptolemy II infrastructure:

Director from a library defines component interaction semantics



Large, polymorphic component library.

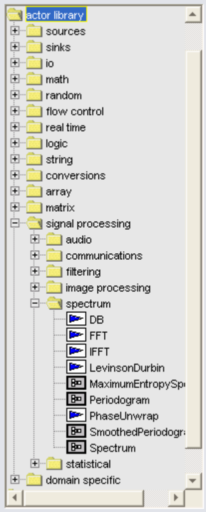
Polymorphic Components - Component Library Works Across Data Types and Domains

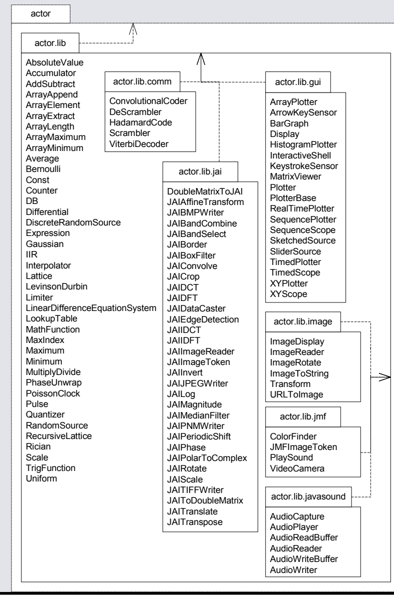


- Data polymorphism:
  - Add numbers (int, float, double, Complex)
  - Add strings (concatenation)
  - Add composite types (arrays, records, matrices)
  - Add user-defined types
- Behavioral polymorphism:
  - In dataflow, add when all connected inputs have data
  - In a time-triggered model, add when 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?

Ptolemy II Component Library





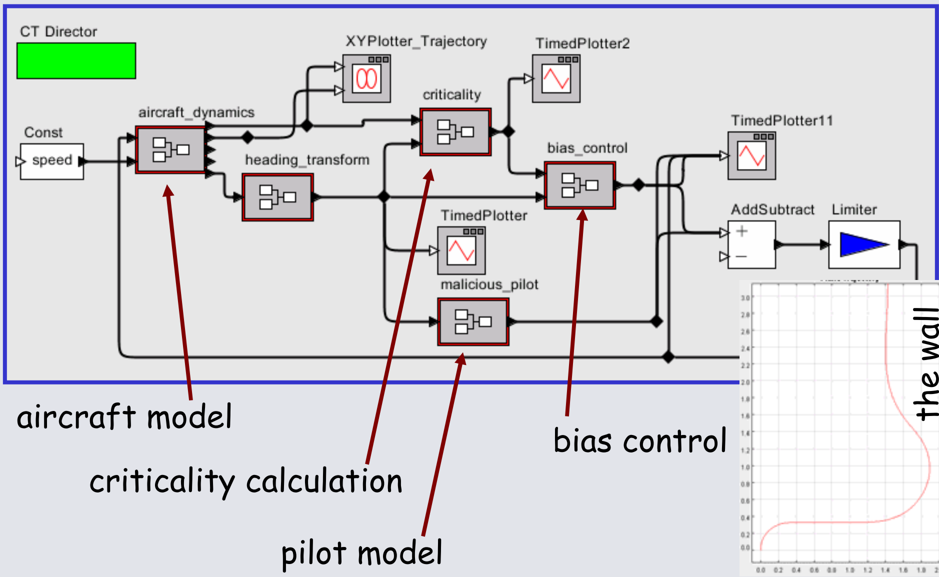
UML package diagram of key actor libraries included with Ptolemy II.

- **Data polymorphic components**
- **Domain polymorphic components**

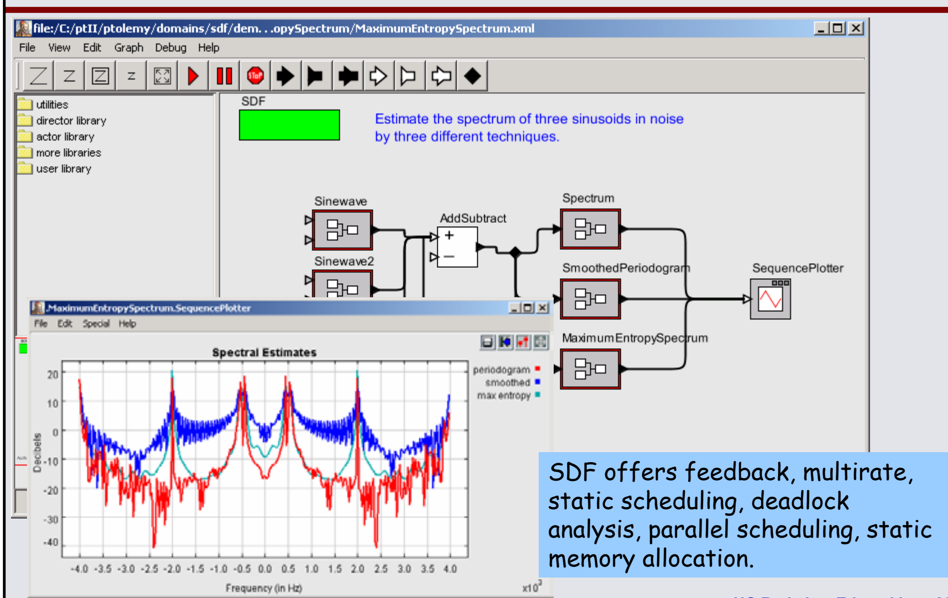
Domains – Provide semantic models for component interactions

- CI – Push/pull component interaction
- CSP – concurrent threads with rendezvous
- CT – continuous-time modeling
- DE – discrete-event systems
- DDE – distributed discrete events
- 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

Continuous-Time Models
Soft Walls Avionics System



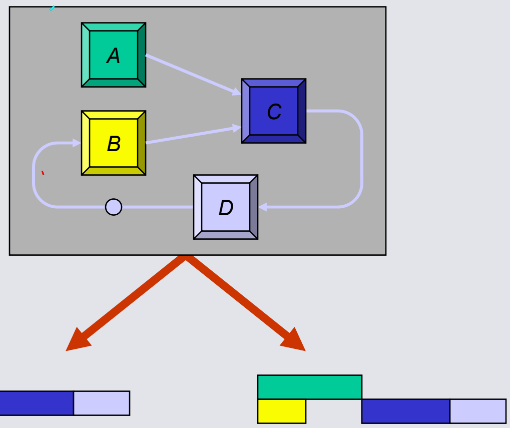
Synchronous Dataflow (SDF)



SDF offers feedback, multirate, static scheduling, deadlock analysis, parallel scheduling, static memory allocation.

Parallel Scheduling of SDF Models

SDF is suitable for automated mapping onto parallel processors

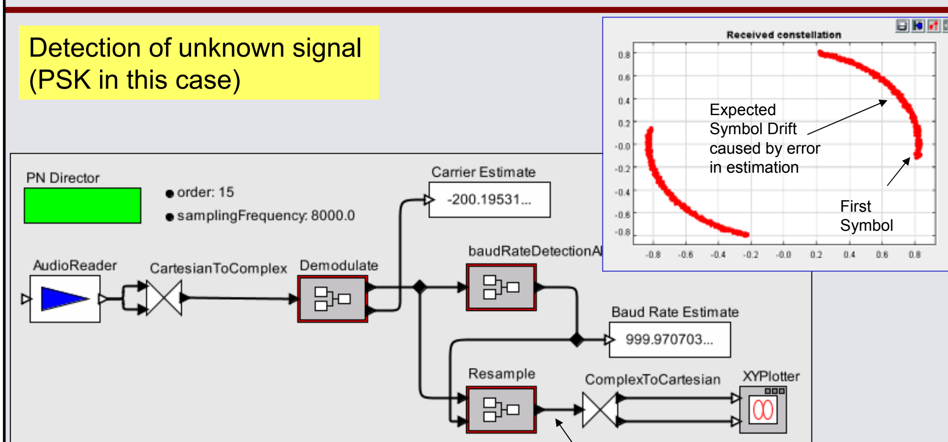


Sequential (software)

Parallel (hardware)

Other Dataflow Models
Process Networks

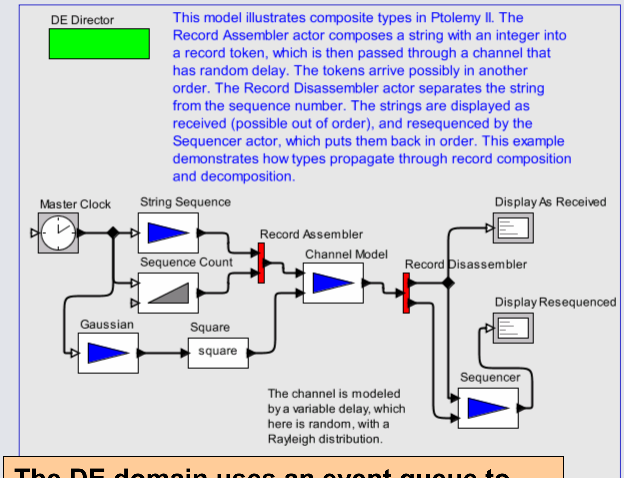
Detection of unknown signal (PSK in this case)



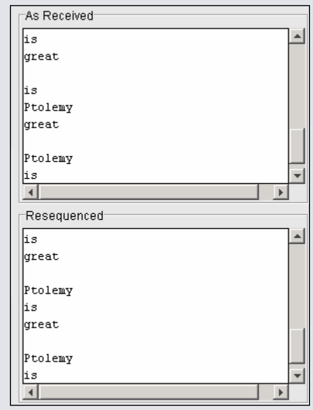
Output data sequence, at detected baud rate. (not known *apriori*)

Challenge problem under DARPA Mobies (Model-based design of embedded software),

Discrete Event Models

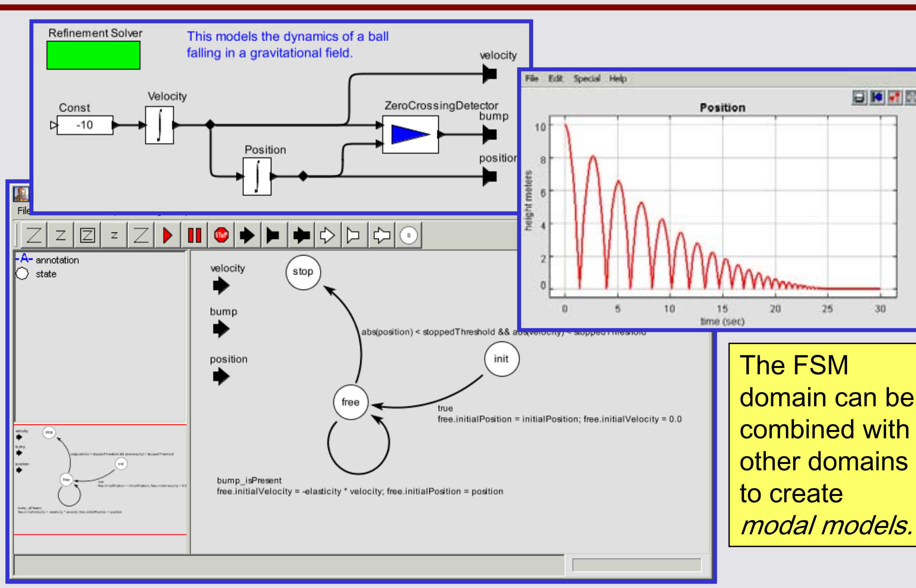


DE domain authors:
Lukito Muliadi
Jie Liu



The DE domain uses an event queue to process events in chronological order, as in VHDL, Verilog, and a number of network simulation languages (e.g. NS).

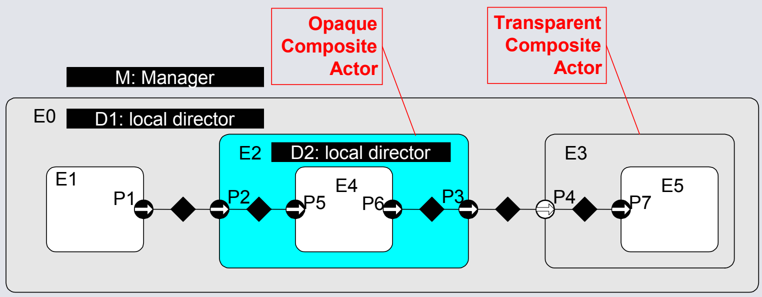
Heterogeneous Models
Hybrid Systems



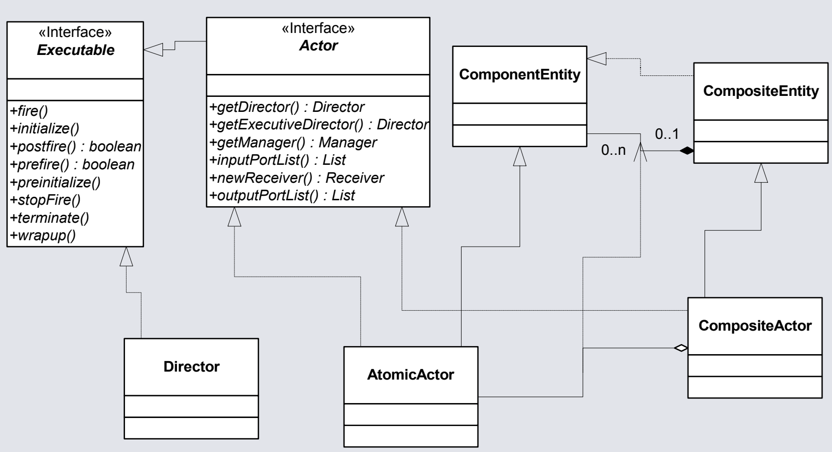
The FSM domain can be combined with other domains to create *modal models.*

Hierarchical Heterogeneity

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

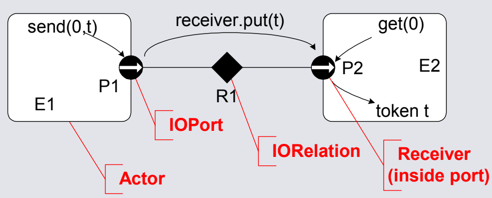


Object Model for Executable Components



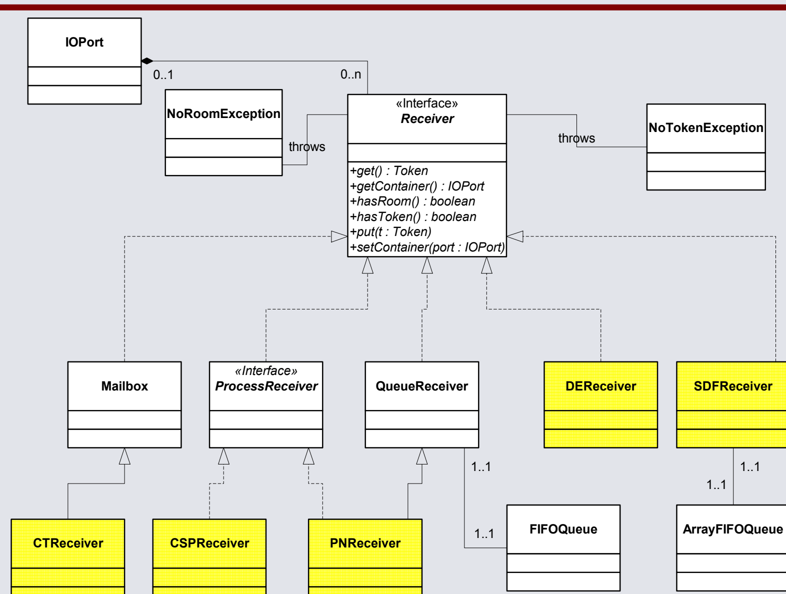
Actor View of Producer/Consumer Components

Basic Transport:



The send() and get() methods on ports are polymorphic. Their implementation is provided by an object implementing the Receiver interface. The Receiver is supplied by the director and implements the communication semantics of a model of computation.

Object Model for Communication Infrastructure

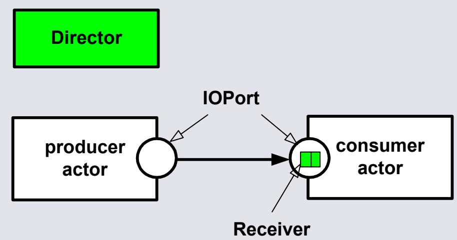


Object-Oriented Approach to Achieving Behavioral Polymorphism

«Interface»
Receiver

+get() : Token
+getContainer() : IOPort
+hasRoom() : boolean
+hasToken() : boolean
+put(t : Token)
+setContainer(port : IOPort)

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.

Extension Exercise

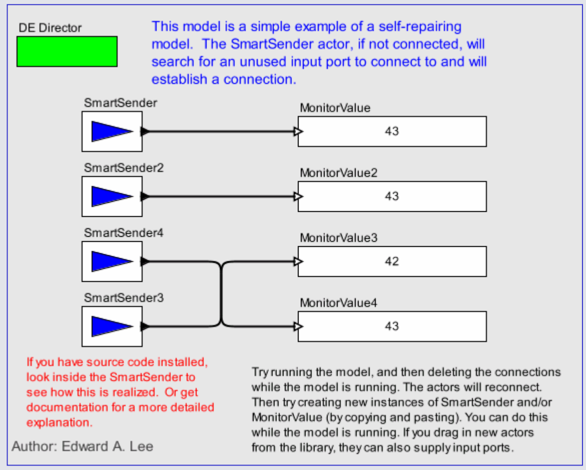
Exercise

Build a director that subclasses SDFDirector to allow substitution of receiver classes in place of the default SDFReceiver. Such substitutions are to be specified by attaching a parameter named "receiverClass" to an input port whose (string) value is the class name of a receiver.

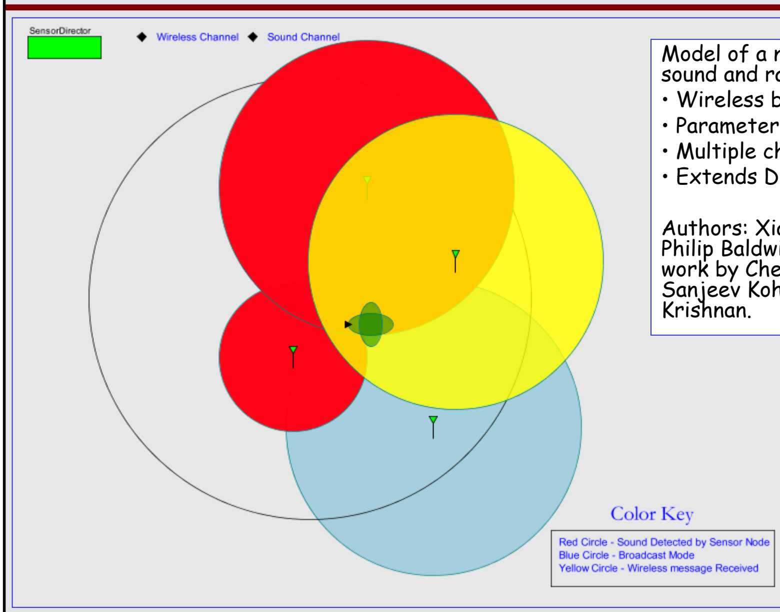
This illustrates a simple mechanism that could be used to support communication refinement.

Examples of 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.



Examples of Extensions
Sensor Nets



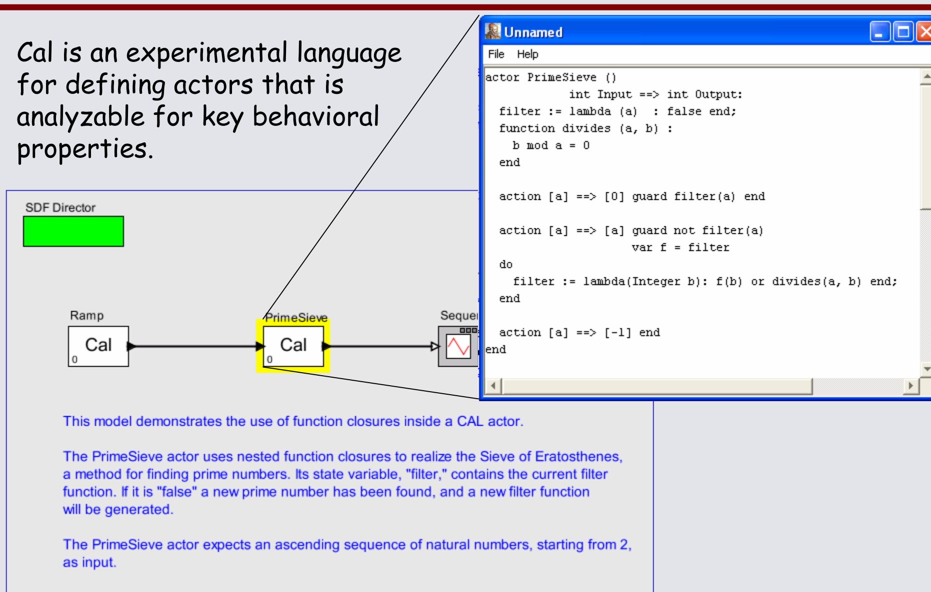
Model of a network with a sound and radio channel.

- Wireless block diagram
- Parameterized icons
- Multiple channels
- Extends DE domain

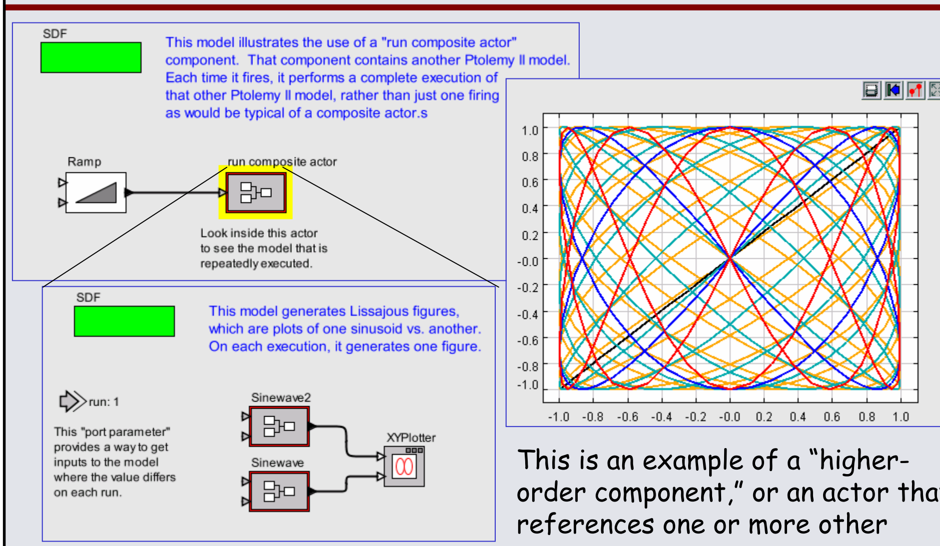
Authors: Xiaojun Liu and Philip Baldwin, based on work by Cheng Tien Ee, Sanjeev Kohli, and Vinay Krishnan.

Example Extensions
Python Actors and Cal Actors

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



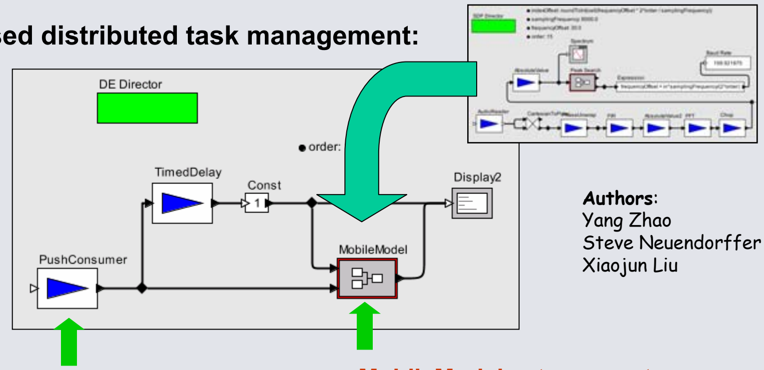
Example Extensions
Using Models to Control Models



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

Examples of Extensions
Mobile Models

Model-based distributed task management:



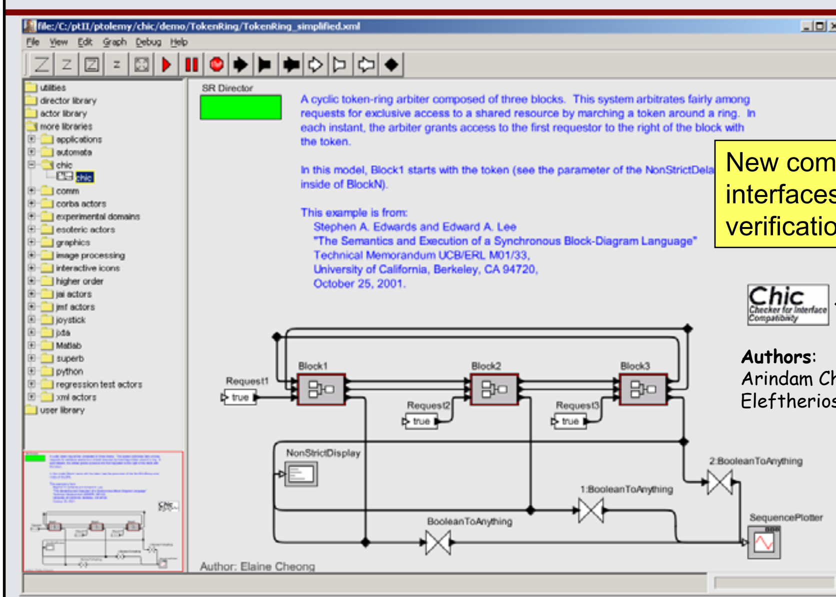
Authors:
Yang Zhao
Steve Neuendorffer
Xiaojun Liu

PushConsumer actor receives pushed data provided via CORBA, where the data is an XML model of a signal analysis algorithm.

MobileModel actor accepts a StringToken containing an XML description of a model. It then executes that model on a stream of input data.

Data *and* domain type safety will help make such models secure

Examples of Extensions
Hooks to Verification Tools

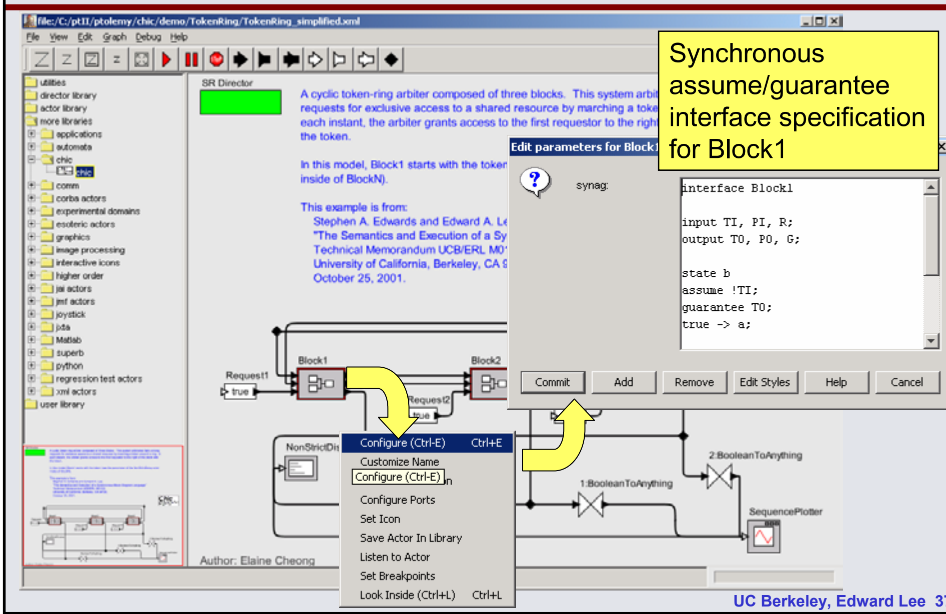


New component interfaces to Chic verification tool

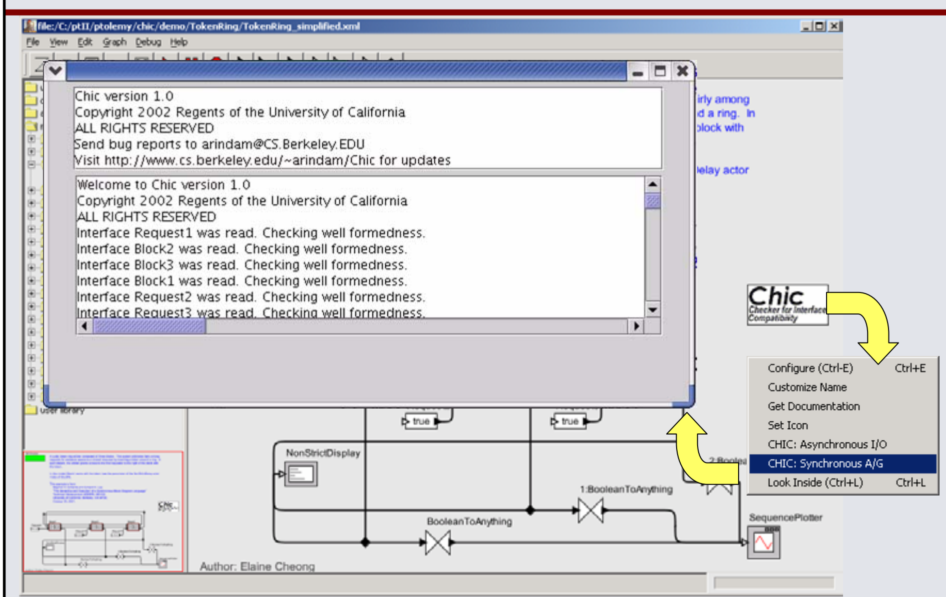
Chic
Checker for Interface Compatibility

Authors:
Arindam Chakrabarti
Eleftherios Matsikoudis

Examples of Extensions
Hooks to Verification Tools



Examples of Extensions
Hooks to Verification Tools

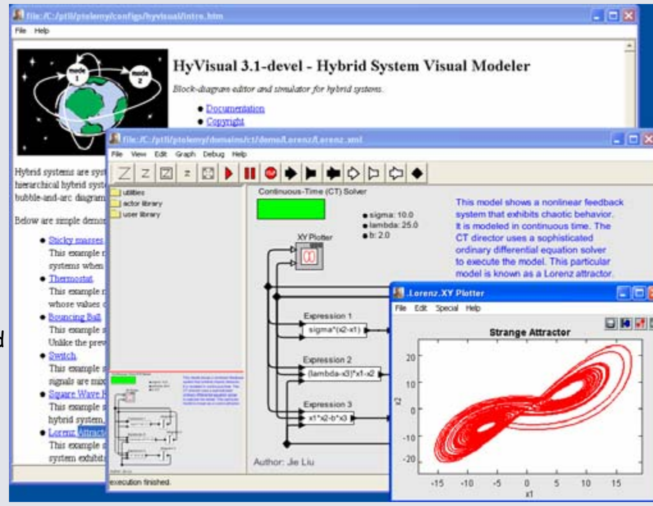


Branding

Ptolemy II configurations are Ptolemy II models that specify
- welcome window contents
- 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.



Configurable Tool Architecture

Ptolemy II configurations enable the construction of specialized tools with customized interfaces and selected portions of the infrastructure.