Process-Based Software Components

Mobies Phase 1, UC Berkeley

Edward A. Lee and Tom Henzinger

(with contributions from Steve Neuendorffer, Christopher Hylands, Jie Liu, Xiaojun Liu, Yang Zhao, and Haiyang Zheng)

PI Meeting, Seattle, WA

July 29-31, 2003

PI: Edward A. Lee, 510-642-0455, eal@eecs.berkeley.edu
Co-PI: Tom Henzinger, 510-643-2430, tah@eecs.berkeley.edu

PM: John Bay
Agent: Dale Vancleave, dale.vancleave@wpafb.af.mil
Award end date: December, 2003
Contract number: F33615-00-C-1703
AO #: J655

Subcontractors and Collaborators

- Subcontractor
  - Univ. of Maryland (C code generation)
- Collaborators
  - Ford/GM/Berkeley (HSIF + automotive OEP)
  - Southwest Research Institute (SW radio OEP)
  - UCB Chess (center for hybrid and embedded software systems)
  - Caltech SEC (fan-driven platform)
  - UCB SEC (helicopters)
  - Kestrel (code generation technology)
  - Vanderbilt (HSIF)
  - Penn (HSIF)
  - CMU (HSIF)
  - Research in Motion Limited
  - Brigham Young University (hardware generation)

Project Goals and Problem Description

Our focus is on component-based design using principled *models of computation* and their *runtime environments* for embedded systems. The emphasis of this project is on the dynamics of the components, including the communication protocols that they use to interface with other components, the modeling of their state, and their flow of control. The purpose of the mechanisms we develop is to improve robustness and safety while promoting component-based design.

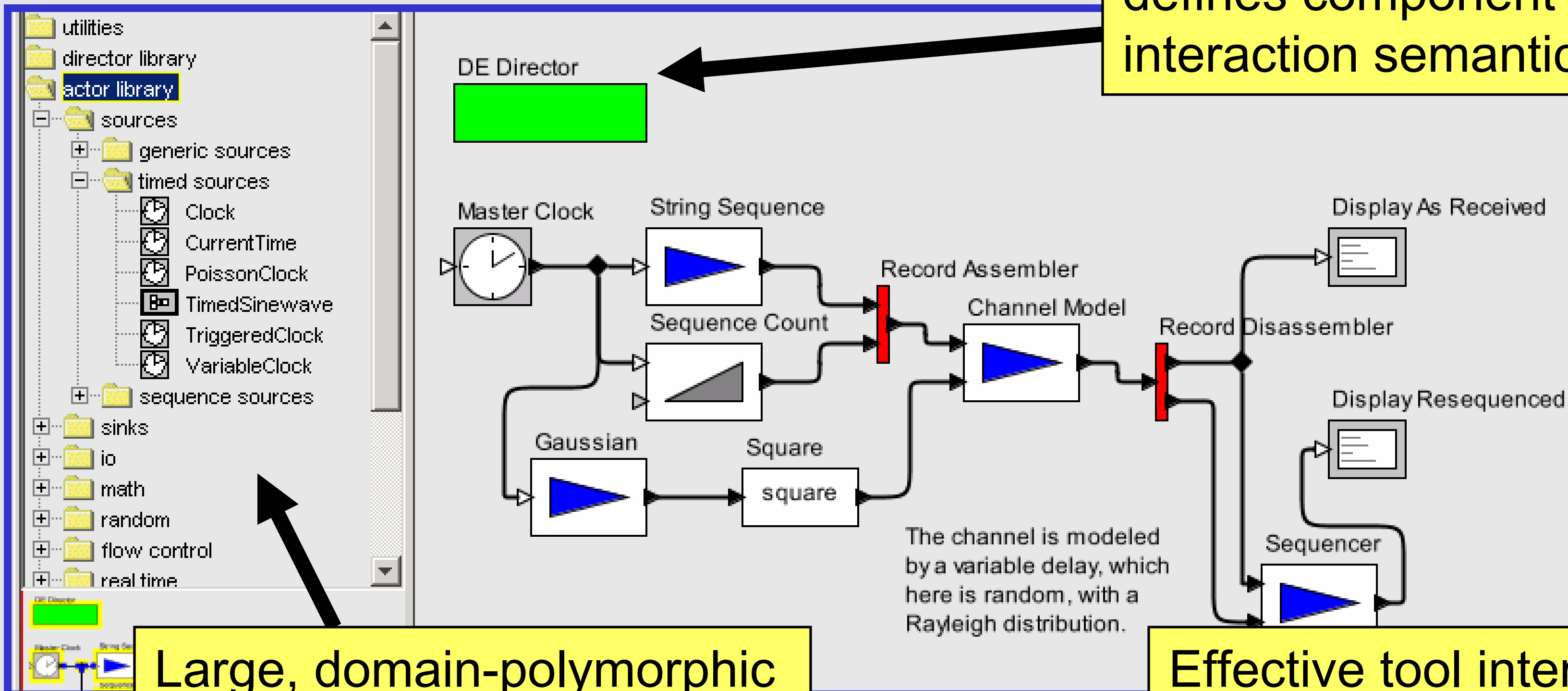
Technical Approach Summary

- Models of computation
  - supporting heterogeneity
  - supporting real-time computation
  - codifications of design patterns
  - definition as *behavioral types*
- Co-compilation and Component Specialization
  - specialization of components for a particular use
  - vs. code generation
  - supporting heterogeneity
- Ptolemy II ← our tool
  - our open-architecture software laboratory
  - shed light on models of computation & co-compilation
  - by prototyping modeling frameworks and techniques

Review Of Ptolemy Project Principles

Basic Ptolemy II infrastructure:

Director from a library defines component interaction semantics



Large, domain-polymorphic component library.

Effective tool interchange requires more narrowly defined semantics and more targeted component libraries.

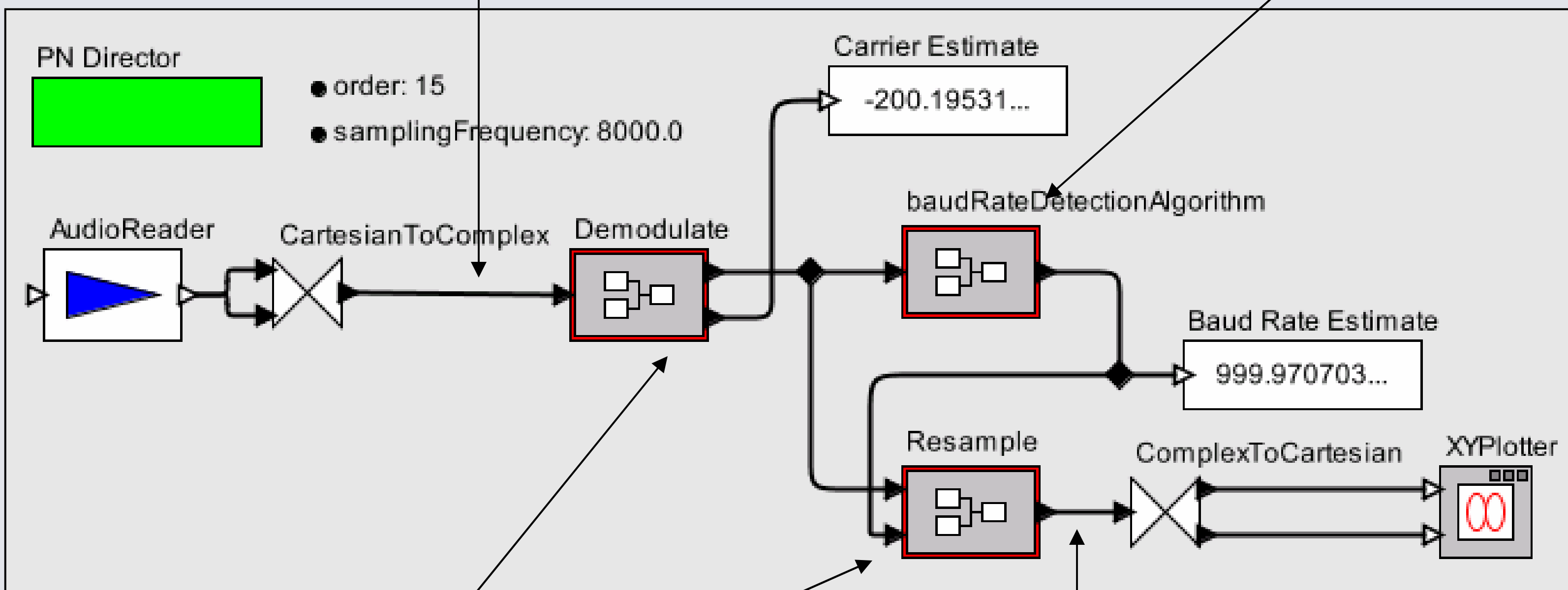
Accomplishments Highlighted Here

- Software Radio OEP
  - Created E0 challenge problem in Ptolemy II
  - Demonstrated model-based task management
  - Created E1 challenge problem in Ptolemy II
  - Identified dataflow variants required
- Hybrid Systems Semantics
  - Guards must be triggers, not enablers
  - Discontinuous signals must have zero transition times.
  - Discrete signals should have values only at discrete times
  - Transient states must be active for zero time.
  - Sampling of discontinuous signals must be well-defined.
  - Transient states must be active for zero time.
- Spinoff Projects
  - Verification by interface checking
  - Sensor nets
  - Network integrated modeling

Ethereal Sting OEP
E1 Challenge Problem

Input data sequence, at *samplingFrequency* Hz

Solution to E0 Challenge Problem



E1 Challenge Problem Components

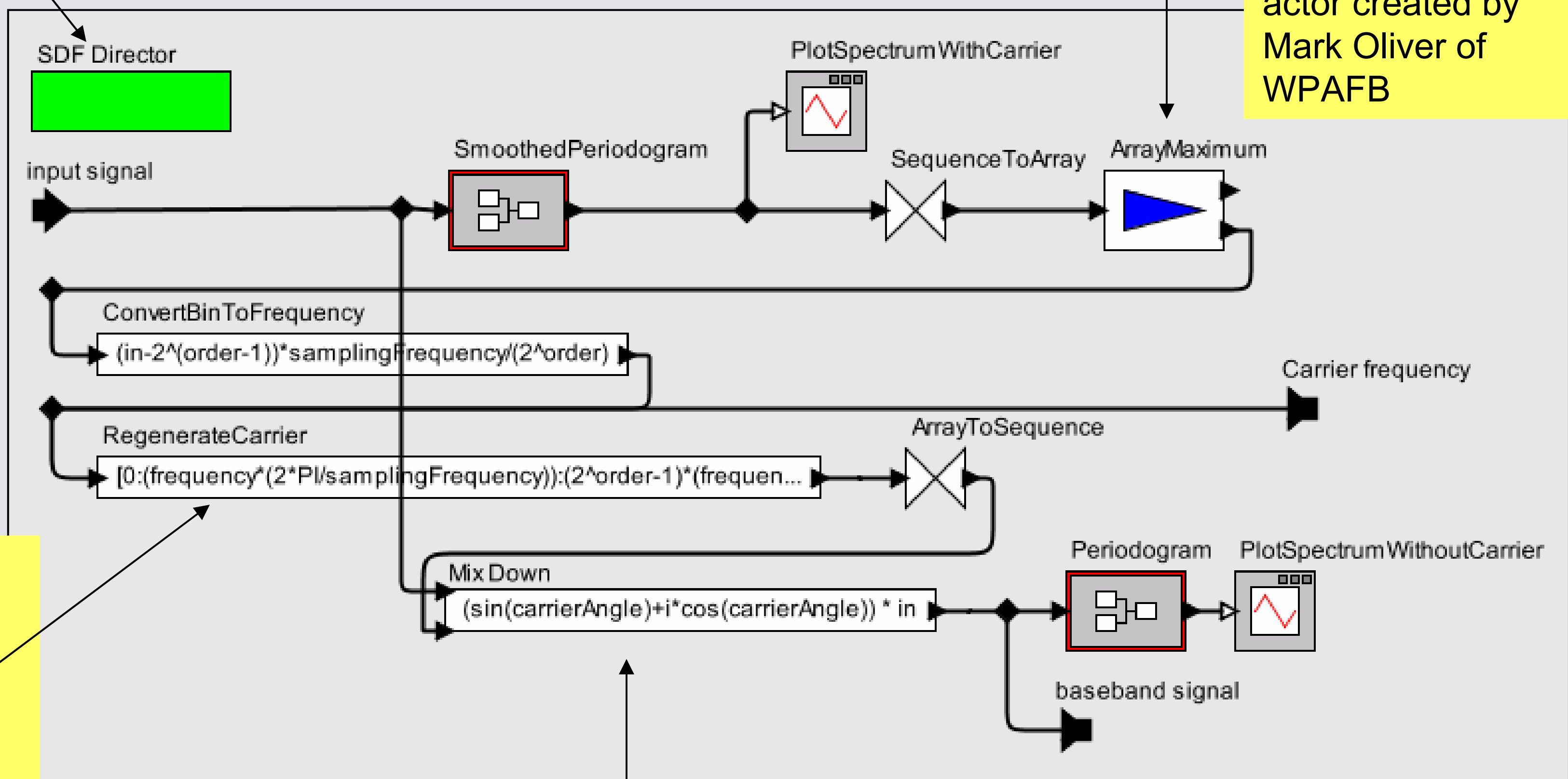
Output data sequence, at detected baud rate. (not known *apriori*)

Carrier Detection and Demodulation

Synchronous Dataflow: Allows for static scheduling and code generation

Executes only once for each complete signal

actor created by Mark Oliver of WPAFB



Array expression creates carrier signal in one line

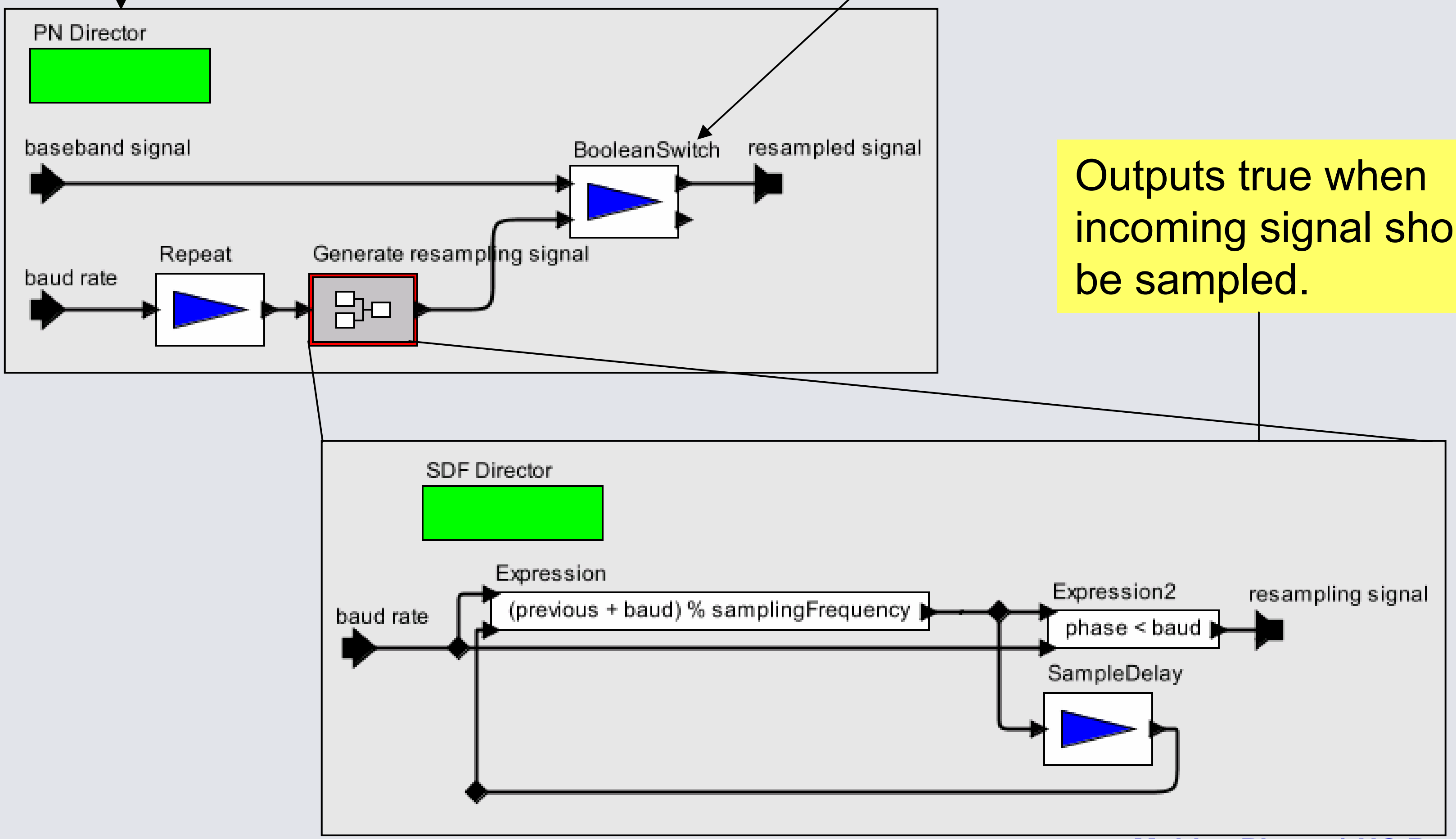
Executes once for every input sample

Resampling

Process Network model: NOT statically scheduled, since output rate not known.

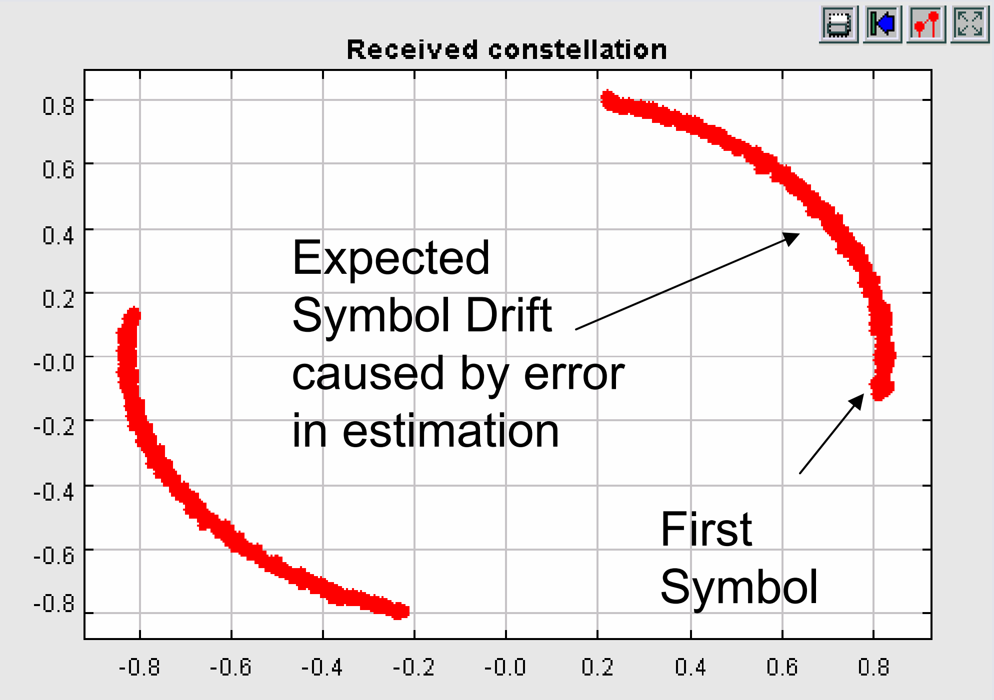
Switch drops input tokens when control token is false.

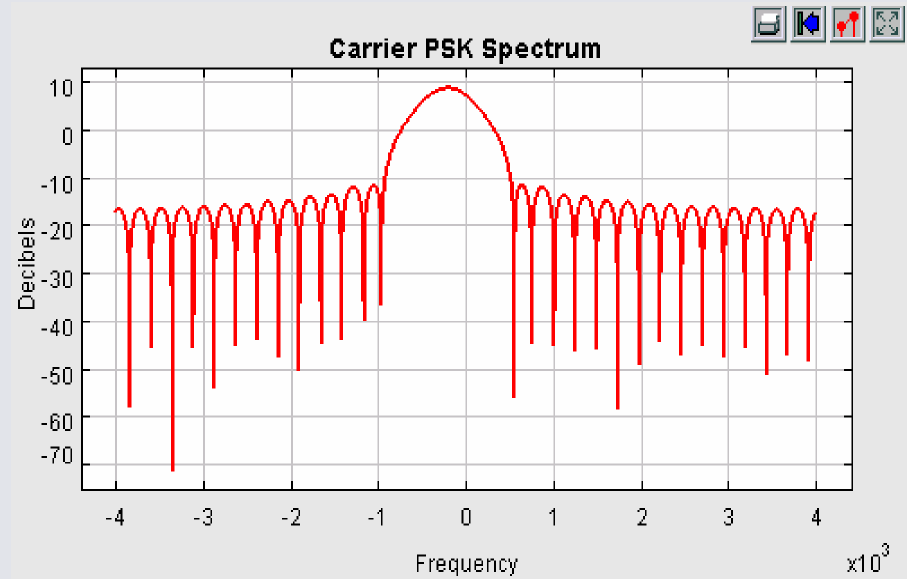
Outputs true when incoming signal should be sampled.

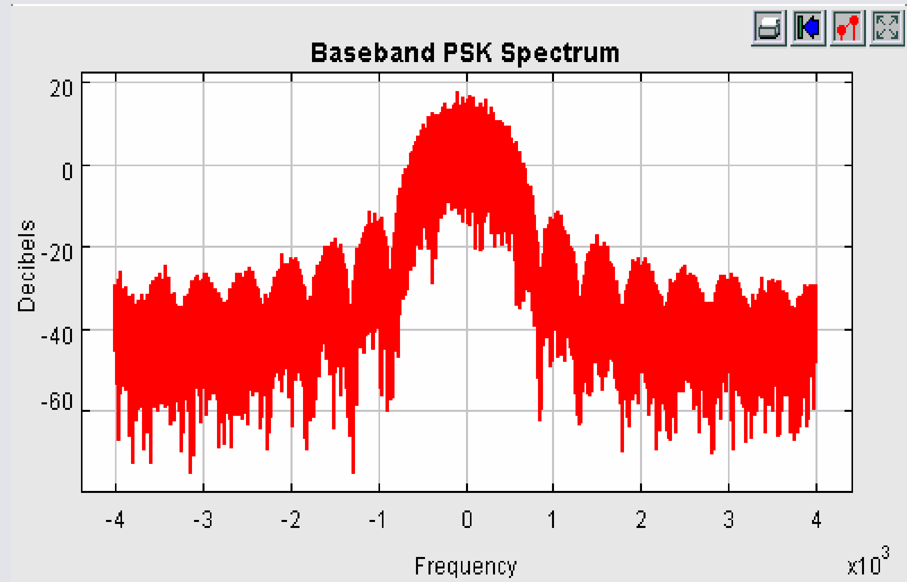


Execution

PSK, 200 Hz carrier, 1KHz symbol rate







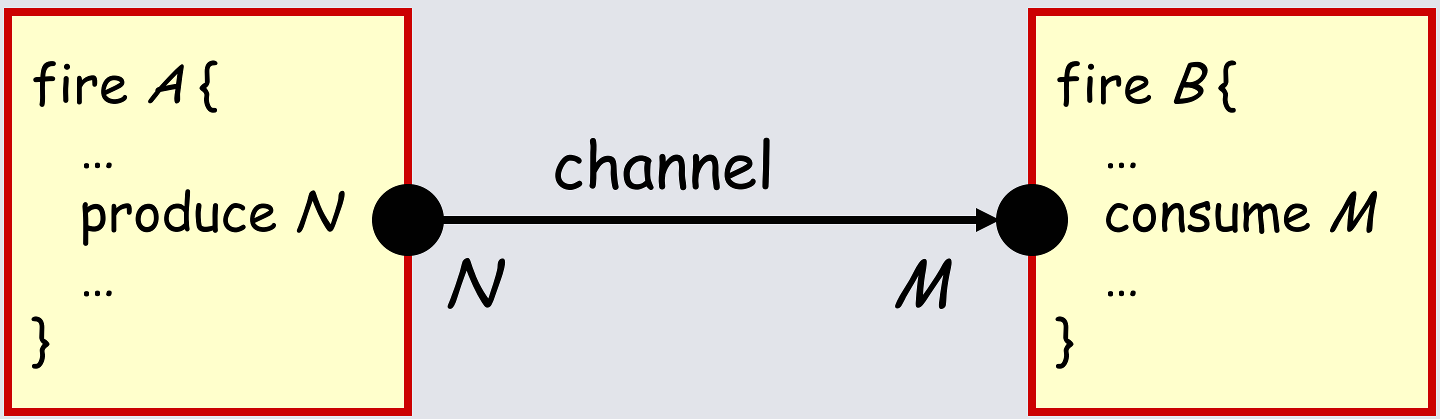
- Solved E1 Challenge Problem
- Working on code generation support for components with unknown data rates.
- Model completed in a short afternoon.

Synchronous Dataflow (SDF)

- Balance equations (one for each channel):

  $$F_A N = F_B M$$

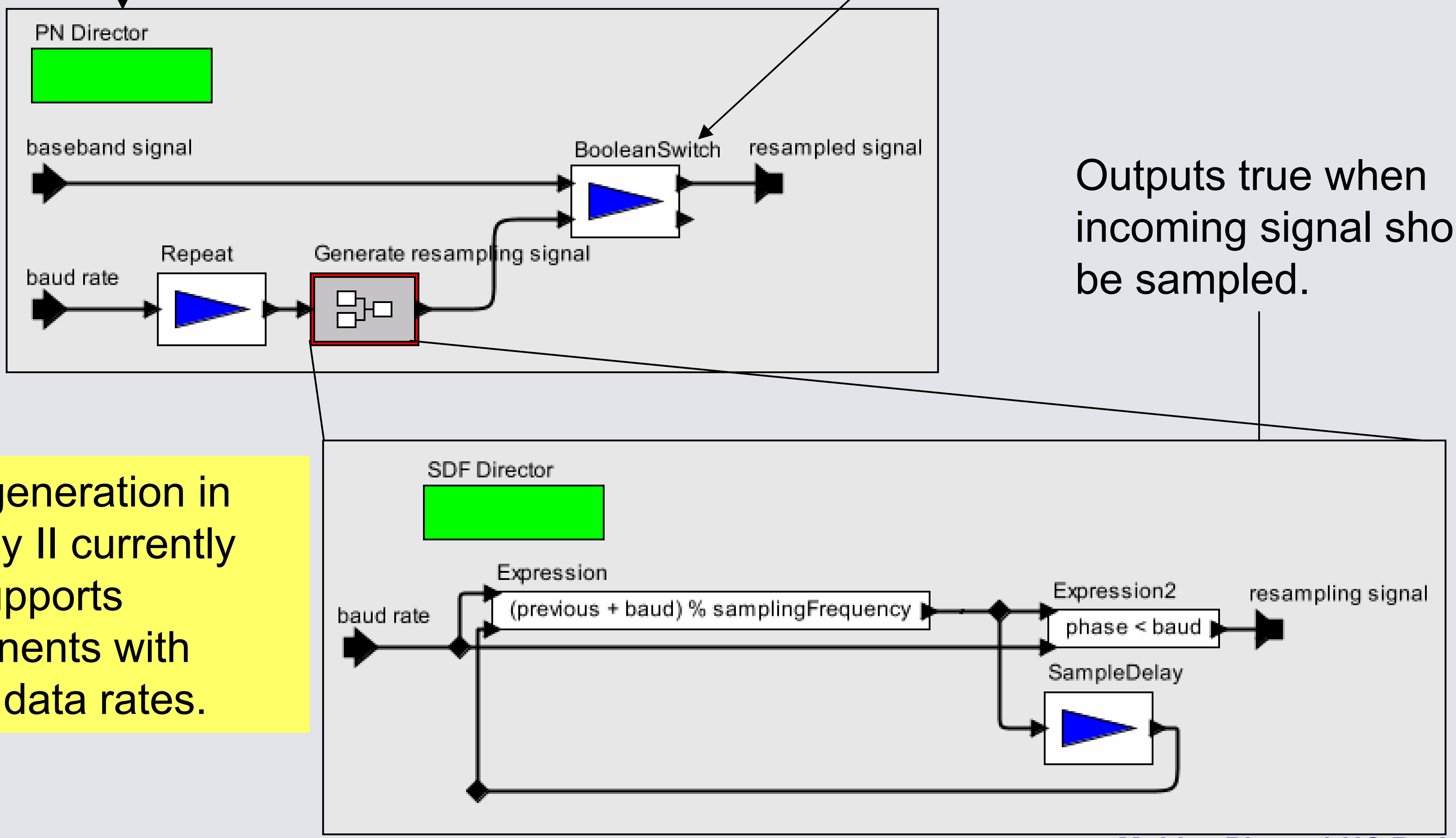
- Schedulable statically (parallel or sequential)
- Decidable resource requirements



Resampling Violates SDF

Process Network model: NOT statically scheduled, since output rate not known.

Switch drops input tokens when control token is false.



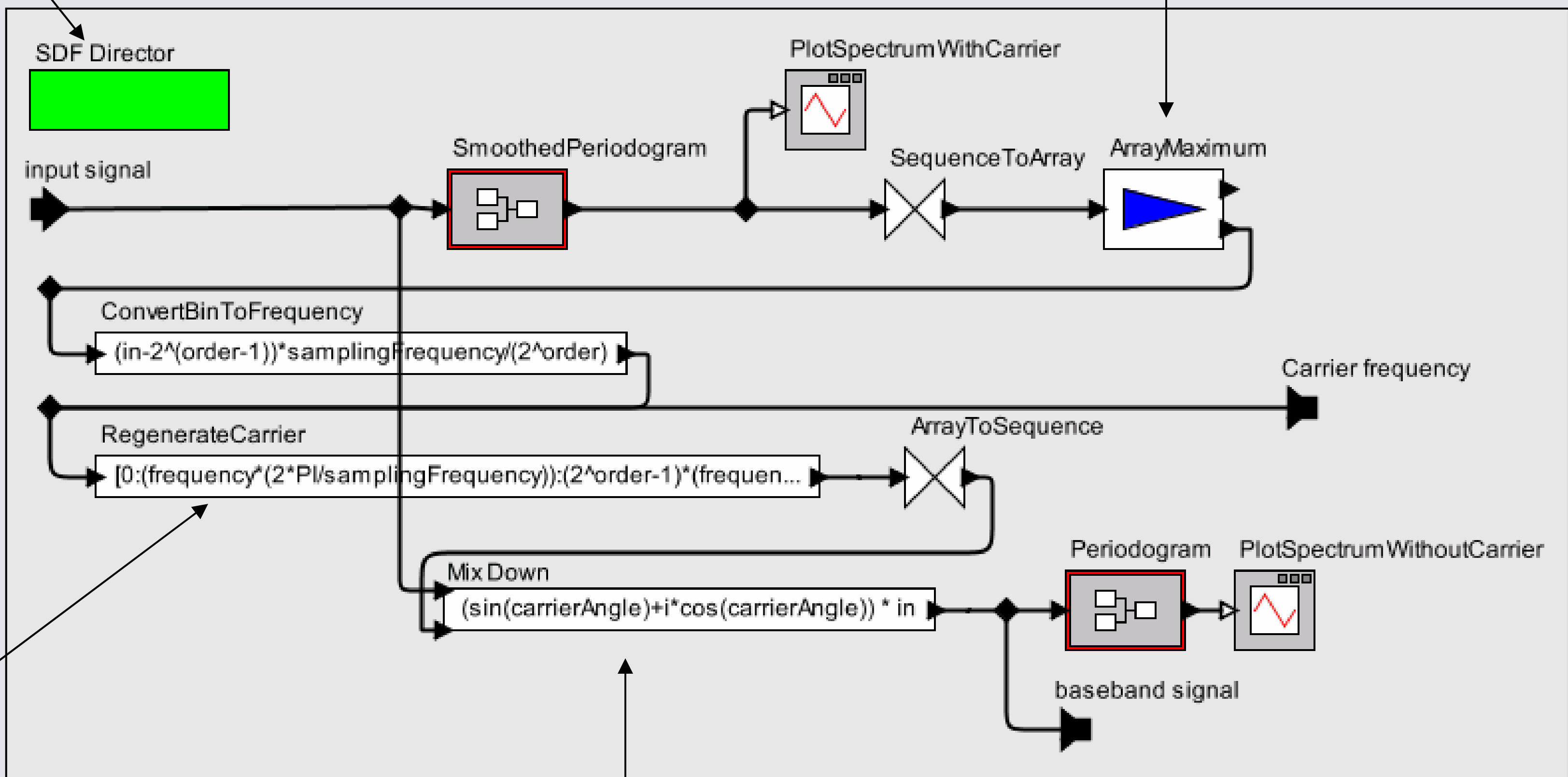
Outputs true when incoming signal should be sampled.

Code generation in Ptolemy II currently only supports components with known data rates.

Carrier Detection and Demodulation Obeys SDF

Synchronous Dataflow: Allows for static scheduling and code generation

Executes only once for each complete signal



Array expression creates carrier signal in one line

Executes once for every input sample

Dataflow Taxonomy

- Synchronous dataflow (SDF)
  - Balance equation formalism
  - Statically schedulable
  - Decidable resource requirements
- Heterochronous Dataflow (HDF)
  - Combines state machines with SDF graphs
  - Very expressive, yet decidable
  - Scheduling complexity can be high
- Boolean & Integer Dataflow (BDF, IDF)
  - Balance equations are solved symbolically
  - Turing-complete expressiveness
  - Undecidable, yet often statically schedulable
- Process Networks (PN)
  - Turing-complete expressiveness
  - Undecidable (schedule and resource requirements)
  - Thread scheduling with interlocks provably executes with bounded resources when that is possible.

Our solution uses these two

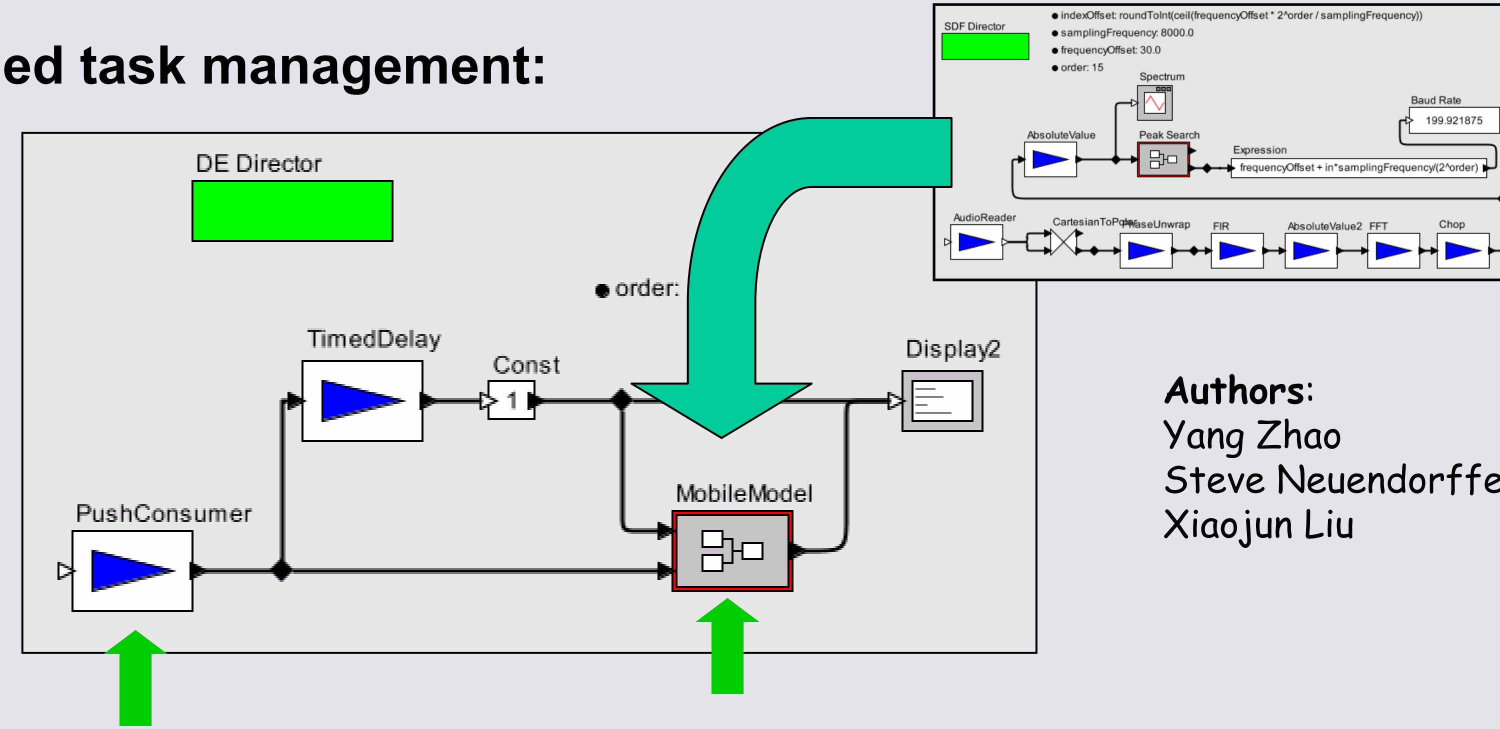
We are implementing this, which works for this problem.

This has been implemented in Ptolemy Classic

Supervisory Structure
Mobile Models Transported via CORBA

Model-based task management:

We described this at the Ethereal Sting Working Group meeting in June.



Authors:
Yang Zhao
Steve Neuendorffer
Xiaojun Liu

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

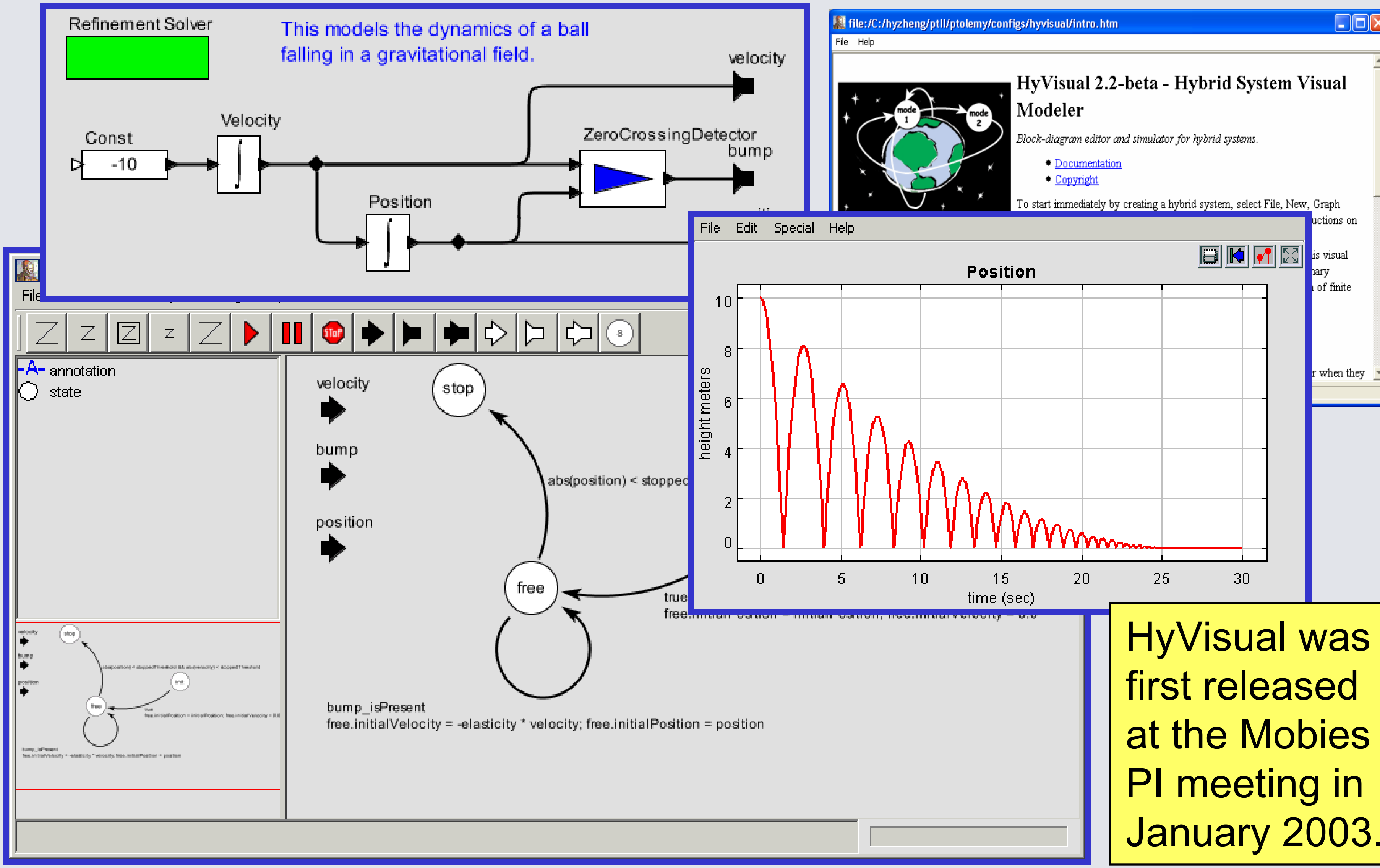
MobileModel actor accepts a StringToken containing an XML description of a model. It then executes that model on a stream of input data.

A significant challenge here is achieving type safety and security.

Accomplishments Highlighted Here

- Software Radio OEP
  - Created E0 challenge problem in Ptolemy II
  - Demonstrated model-based task management
  - Created E1 challenge problem in Ptolemy II
  - Identified dataflow variants required
- Hybrid Systems Semantics
  - Guards must be triggers, not enablers
  - Discontinuous signals must have zero transition times.
  - Discrete signals should have values only at discrete times
  - Transient states must be active for zero time.
  - Sampling of discontinuous signals must be well-defined.
  - Transient states must be active for zero time.
- Spinoff Projects
  - Verification by interface checking
  - Sensor nets
  - Network integrated modeling

HyVisual – Hybrid System Modeling Tool Based on Ptolemy II



HyVisual was first released at the Mobies PI meeting in January 2003.

Operational Semantics of Hybrid Systems
(How to Build Simulators)

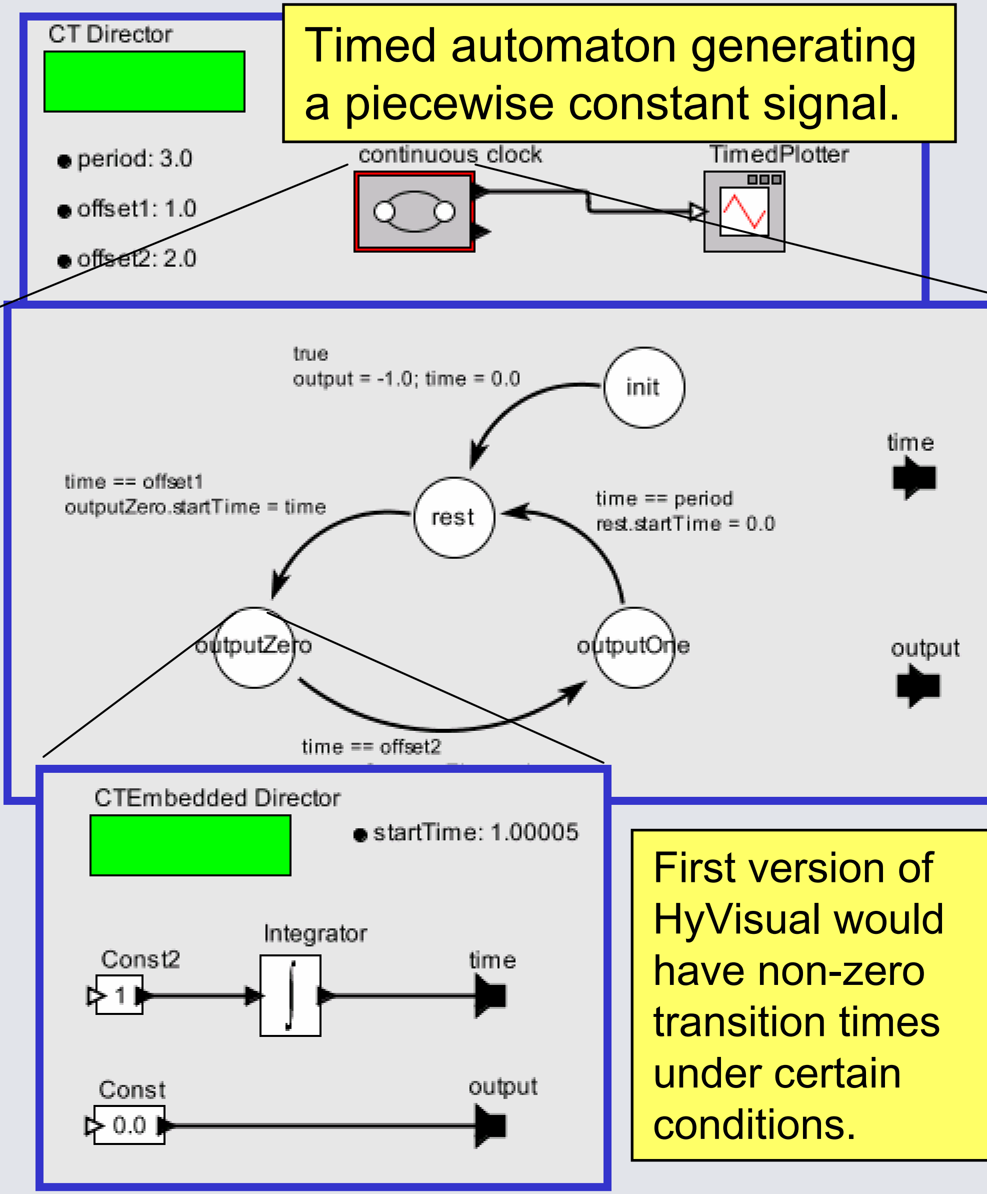
- If you are going to rely on simulation results, then you need an operational semantics.
  - Hybrid system semantics tend to be denotational.
- A simulator cannot ignore nondeterminism.
  - It is incorrect to choose one trajectory.
  - Creating deterministic models must be easy.
  - Nondeterministic models must be explored either exhaustively or using Monte Carlo methods.
  - *Must* avoid unnecessary nondeterminism.
- Should not use continuous-time models to represent discrete behaviors.
  - Inaccurate for software.
  - Heterogeneous models are better.

What we Have Learned from HyVisual

- Guards must be triggers, not enablers
  - Avoid unnecessary nondeterminism.
- Discontinuous signals must have zero transition times.
  - Precise transition times.
  - Accurate model of Zeno conditions.
  - Avoid unnecessary nondeterminism.
- Discrete signals should have values only at discrete times
  - Accurately heterogeneous model (vs. continuous approximation)
- Sampling of discontinuous signals must be well-defined.
  - Avoid unnecessary nondeterminism.
- Transient states must be active for zero time.
  - Properly represent glitches.

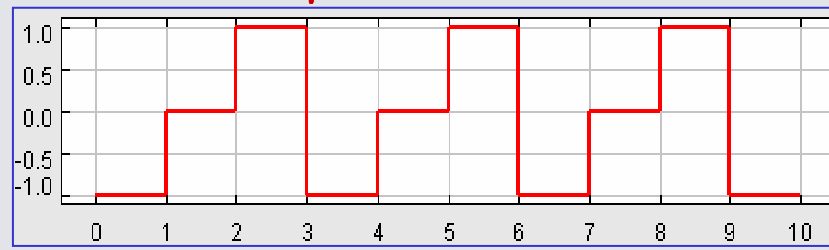
Discontinuous Signals

Timed automaton generating a piecewise constant signal.

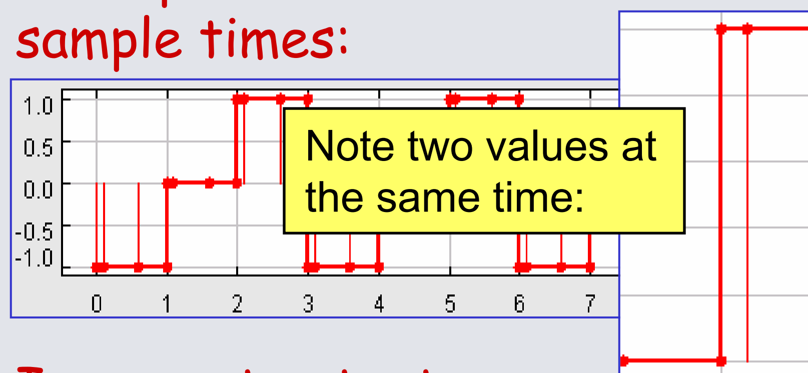


First version of HyVisual would have non-zero transition times under certain conditions.

Correct output:

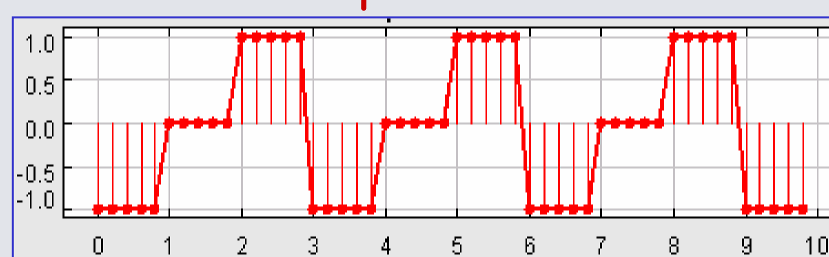


RK 2-3 variable-step solver and breakpoint solver determine sample times:

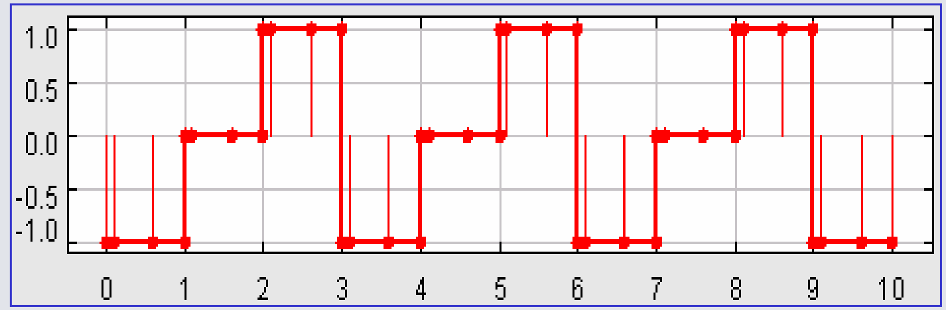


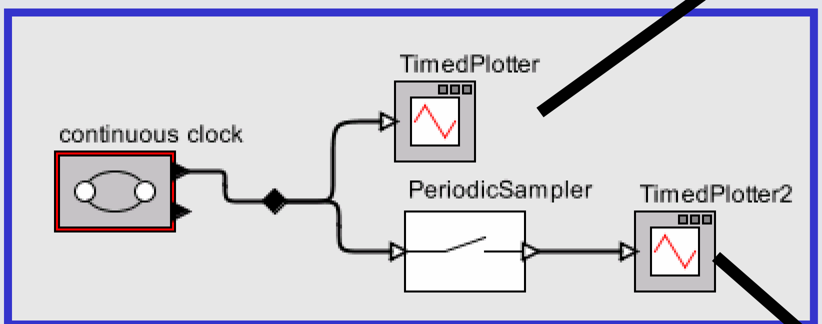
Note two values at the same time:

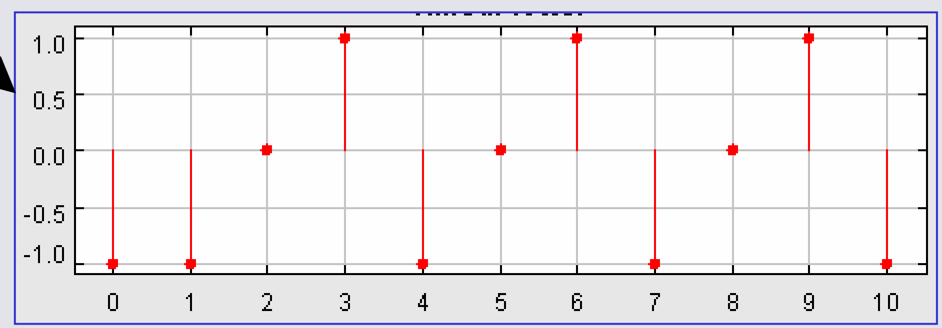
Incorrect output:



Sampling Discontinuous Signals



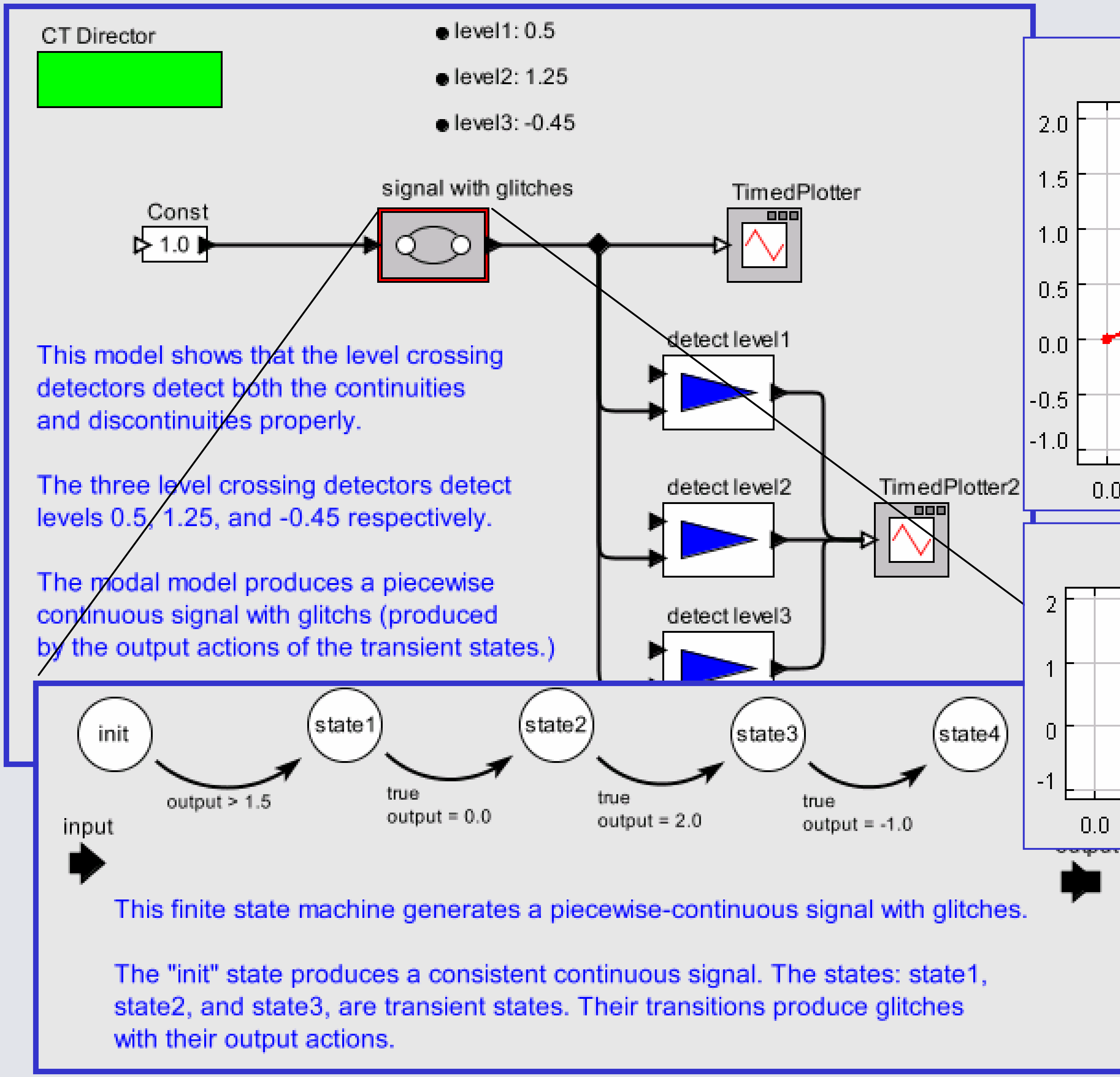


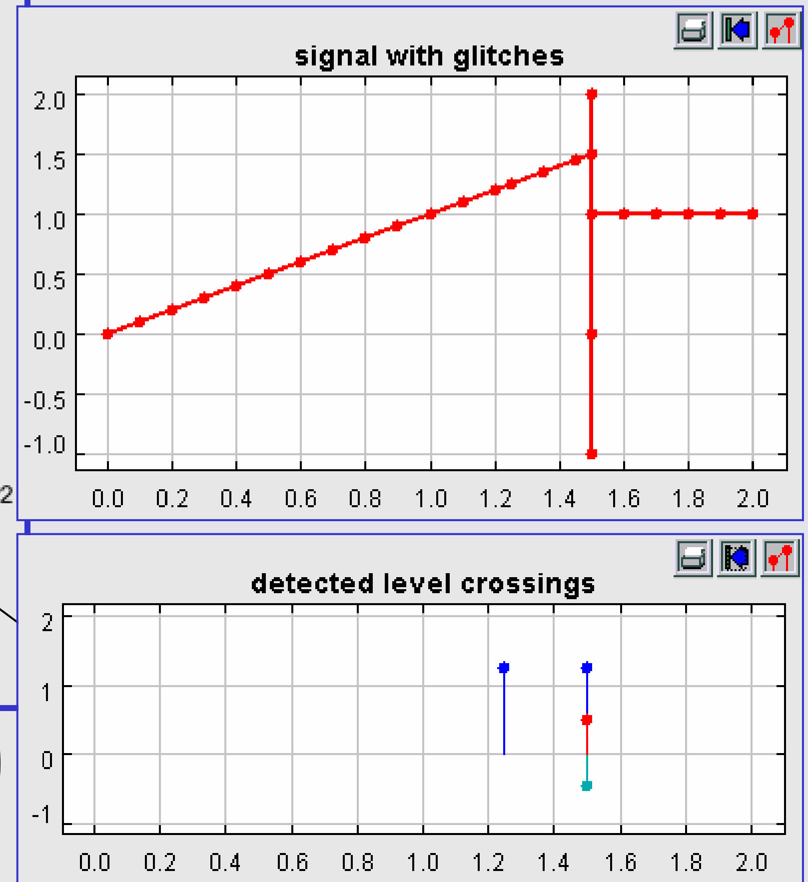


Samples must be deterministically taken at t- or t+. Our choice is t-, inspired by hardware setup times.

Note that in HyVisual, unlike Simulink, discrete signals have no value except at discrete points.

Transient States and Glitches



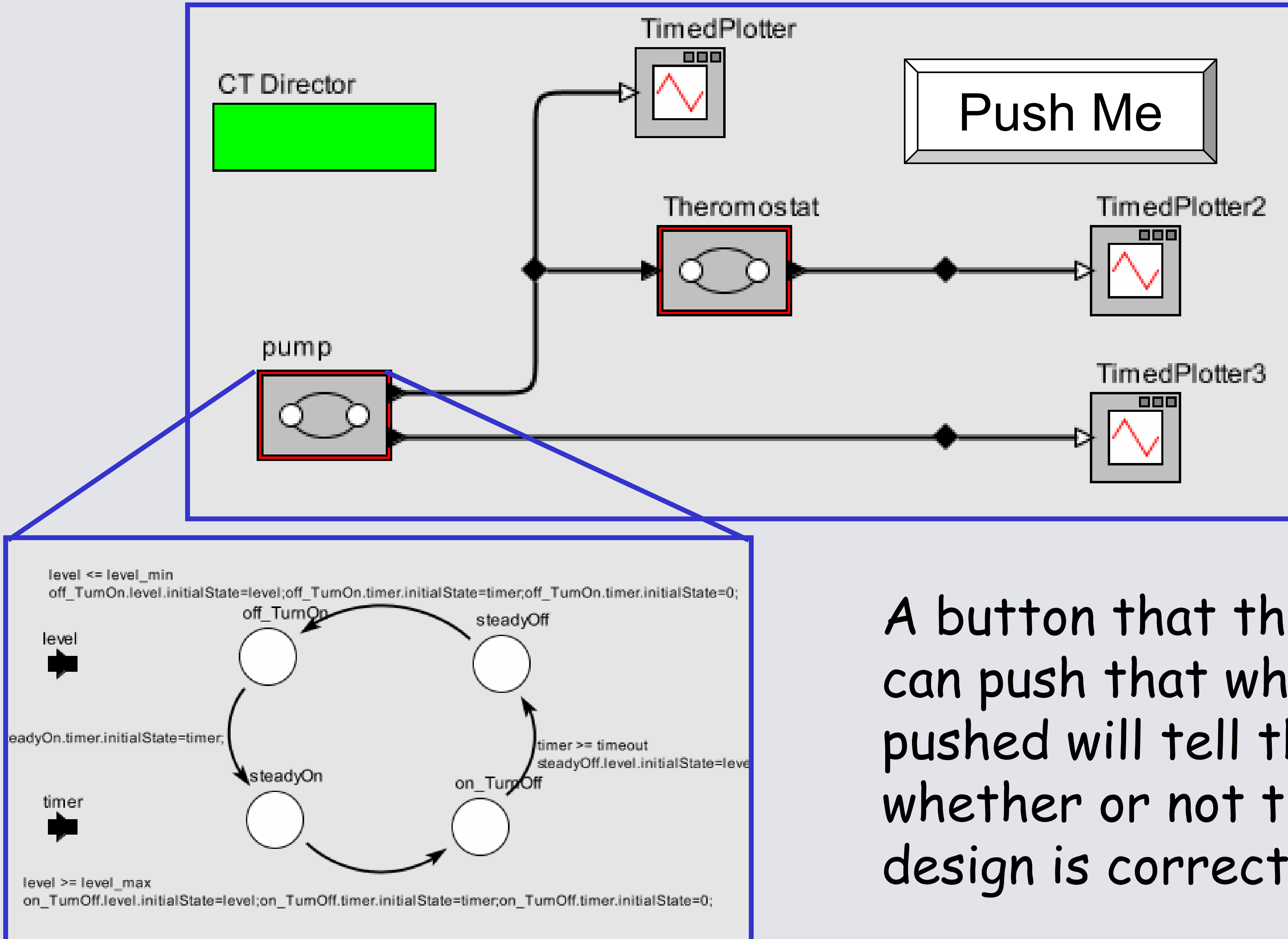


If an outgoing guard is true upon entering a state, then the time spent in that state is identically zero. This can create glitches.

Accomplishments Highlighted Here

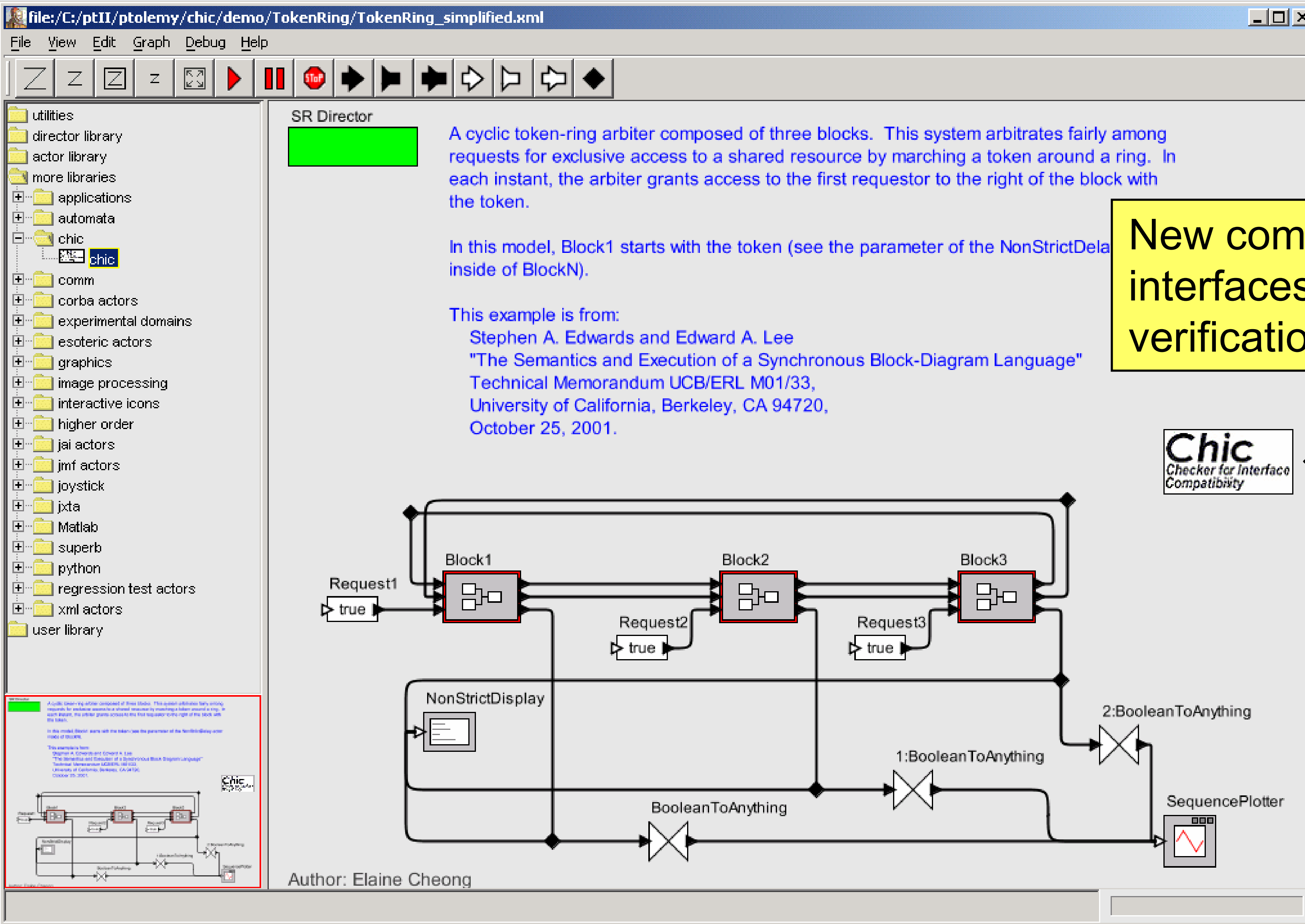
- Software Radio OEP
  - Created E0 challenge problem in Ptolemy II
  - Demonstrated model-based task management
  - Created E1 challenge problem in Ptolemy II
  - Identified dataflow variants required
- Hybrid Systems Semantics
  - Guards must be triggers, not enablers
  - Discontinuous signals must have zero transition times.
  - Discrete signals should have values only at discrete times
  - Transient states must be active for zero time.
  - Sampling of discontinuous signals must be well-defined.
  - Transient states must be active for zero time.
- Spinoff Projects
  - Verification by interface checking
  - Sensor nets
  - Network integrated modeling

Verification & Validation
What Many People Say They Want



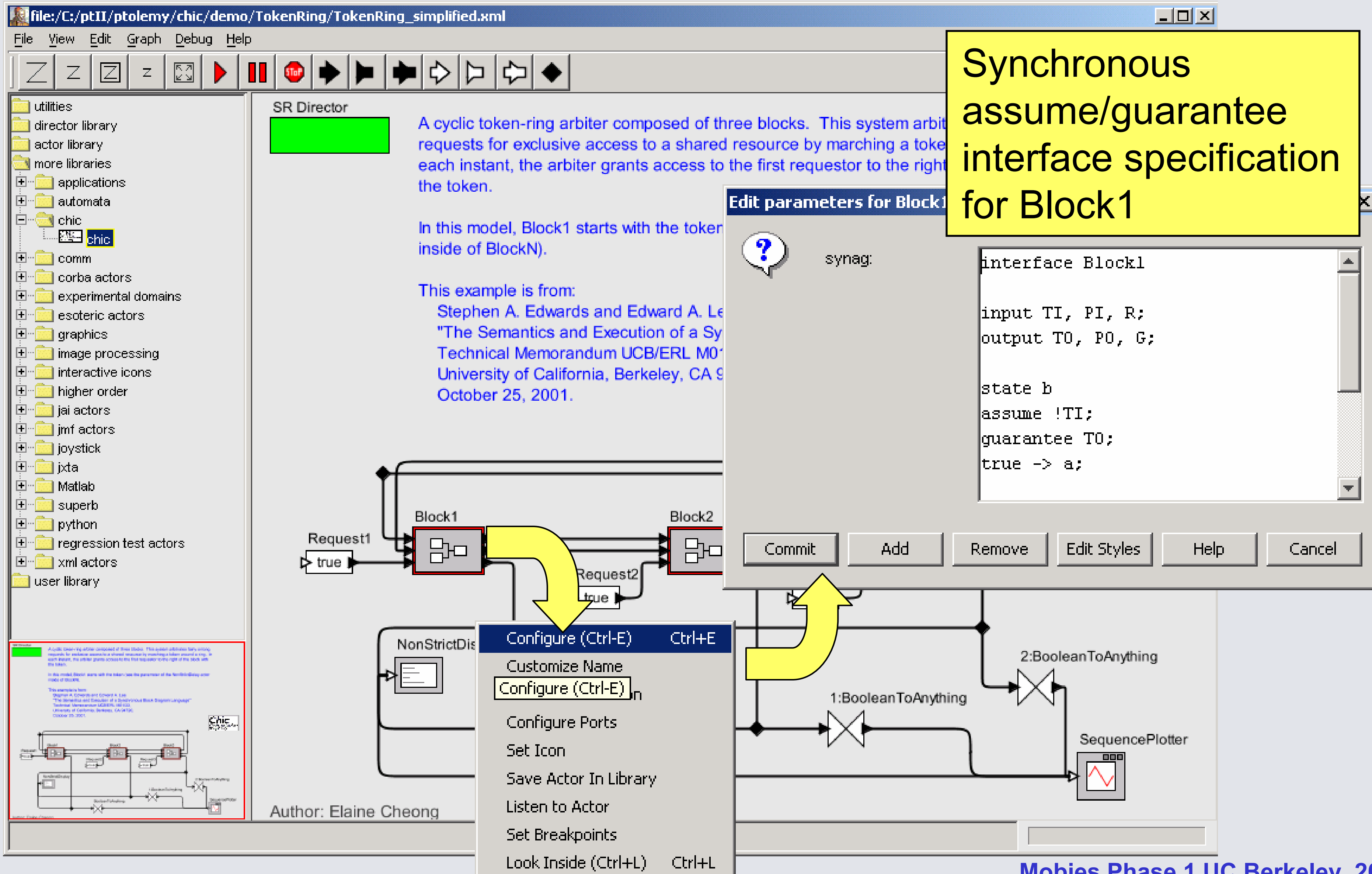
A button that they can push that when pushed will tell them whether or not the design is correct.

Spinoff to Chess NSF/ITR (& Escher?)
Verification by Interface Checking

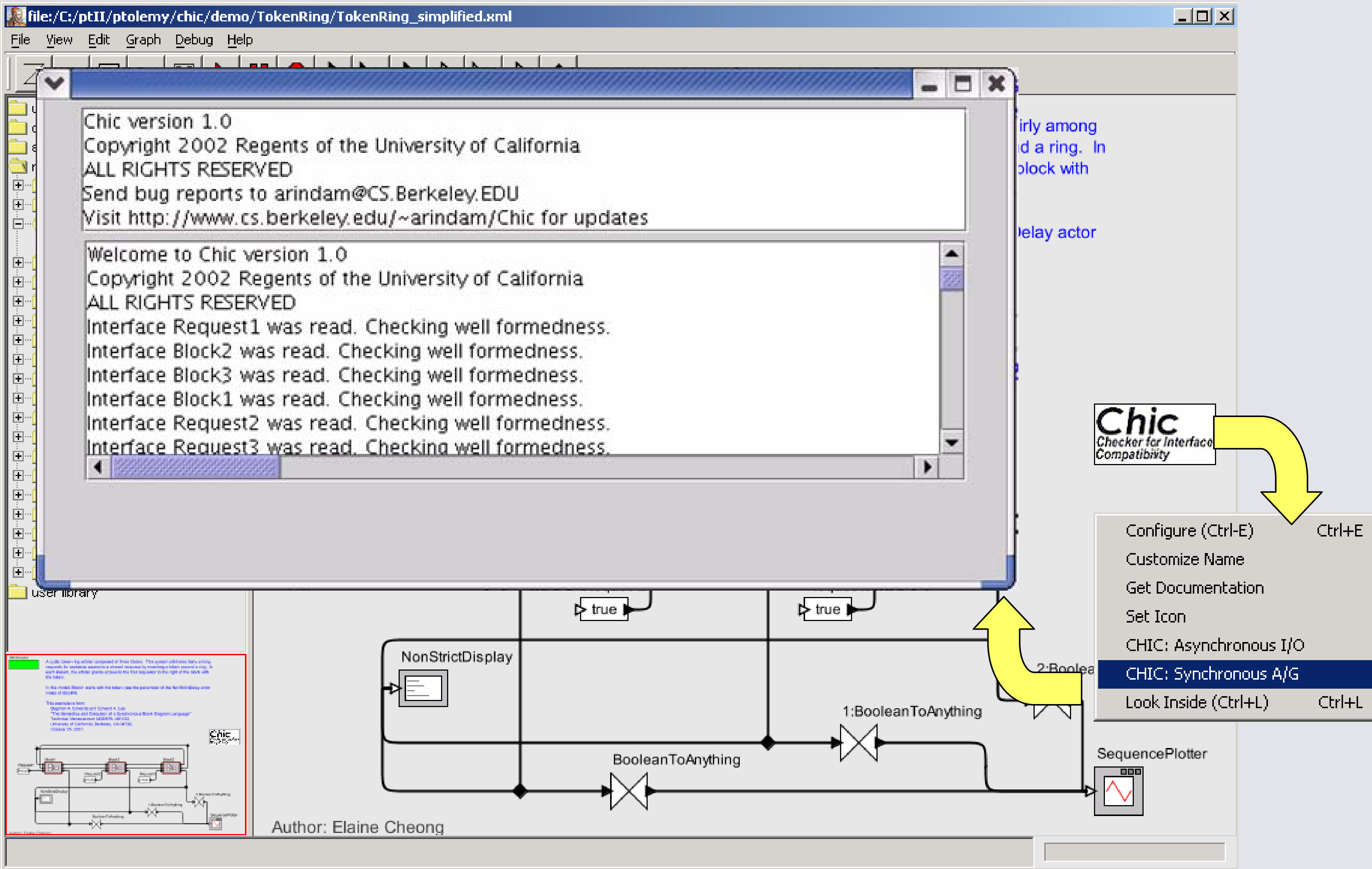


New component interfaces to Chic verification tool

Interfaces Specified as Attributes
Multiple Interface Theories can Co-Exist



Compatibility Checking Via the Chic Component

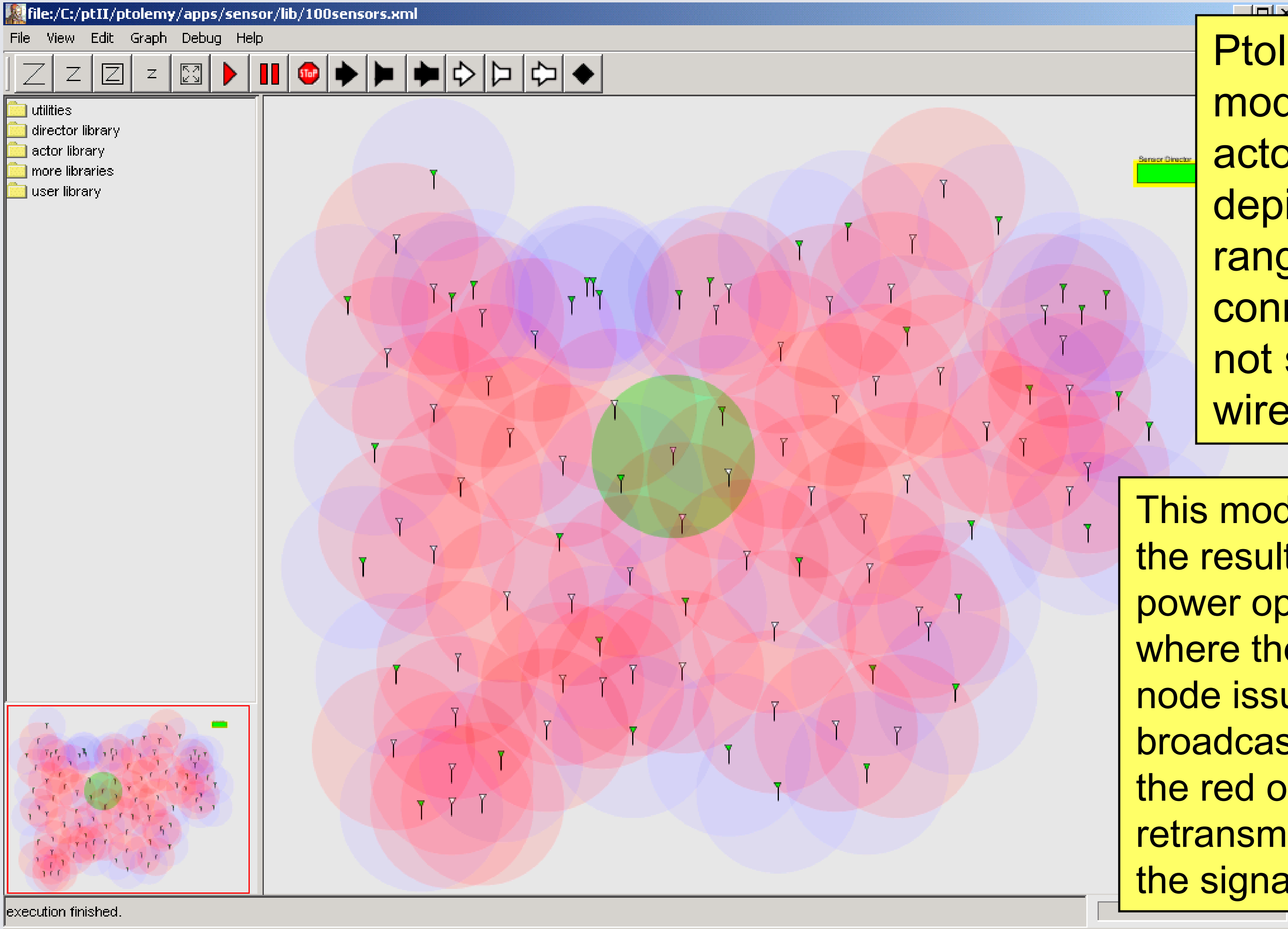


Limitations of Interface Checking Further Work under NSF/ITR (& Escher?)

- Chic currently only runs on Linux
  - Uses a non-portable BDD package
- "Synchronous" semantics doesn't match SR
  - State-dependent output only
  - Does match Giotto
- Interface automata composition not yet there
  - Implemented in Ptolemy II, but not as part of a general interface checker like Chic
  - Prefer visual specification of behavioral types

- We don't yet know what sorts of interface theories and interfaces will be useful
  - But we now have the experimental framework to try things out.

Spinoff to Chess (NSF/ITR) Sensor Nets Modeling

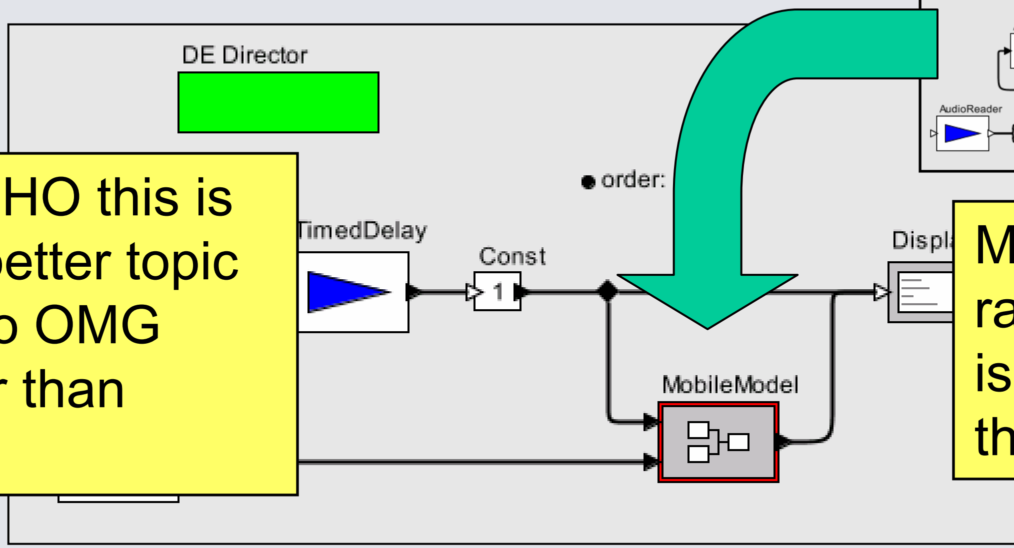


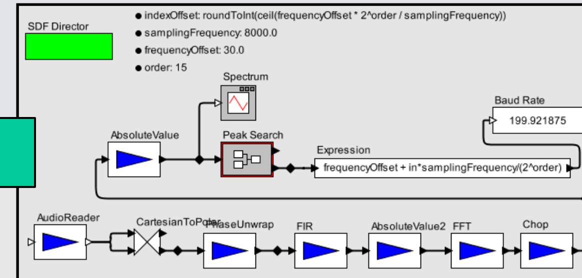
Ptolemy II model where actor icons depict sensor range and connectivity is not shown with wires

This model shows the results of a power optimization where the green node issues a broadcast signal and the red ones retransmit to relay the signal.

Spinoff to ??? (Unfunded)
Distributed Actor-Oriented Middleware

Model-based distributed computing and middleware:





Note that IMHO this is probably a better topic for Mobies to OMG tech transfer than HSIF.

Mobile model for SW radio task management is an early instance of this technology

- Domain-specific middleware (distributed Ptolemy domains)
- Actor-oriented middleware (concurrent models of computation)
- Behavioral interface definitions for components & middleware
- Middleware-polymorphic components
- Failure management
- Security (authentication, encryption, capability management)

Plans
(Most of the Tech Team Moves on in August)

- Software radio OEP target: E2 and model-based task management
  - Handle failures of mobile model
  - Secure execution of mobile model
  - Encrypted communication of models & data
  - Authenticated access to MobileModels
- Transition interface definition/checking work to Chess
  - Generalize Chic methods to true synchronous models
  - Integrate interface automata composition framework
  - Identify useful interface theories
- Wrap up code generation
  - Support PN code generation
- Hybrid system semantics and HSIF
  - Resolve remaining semantics questions
  - Ensure HyVisual compliance with HSIF
  - Transition this work to Chess - Toyota, Daimler-Chrysler (and Escher?)

Technology Transition
Key Activities Since January

- Ptolemy II chosen as workflow design/execution environment for
  - NSF SEEK project (http://seek.ecoinformatics.org)
  - DOE SDM project (http://sdm.lbl.gov/sdmcenter/)
- Ptolemy II version 3.0-beta and 3.0.1 (first released here)
  - Major release with many enhancements
- HyVisual version 3.0.1 (first released here)
  - Branded, domain-specific tool based on Ptolemy II
- Ptolemy Miniconference, May 9, 2003
  - 90 attendees from 43 organizations worldwide
- Commercializations and commercial applications
  - RIM: Blackberry handheld
  - MLDesign Technologies: MLDesigner software modeling
  - Thales: Distributed process networks and native C components
  - RSoft: LambdaSIM, optical network modeling
  - Comenius University, Slovakia: VT11: microcomputer interfacing education
- Many third party contributions to software