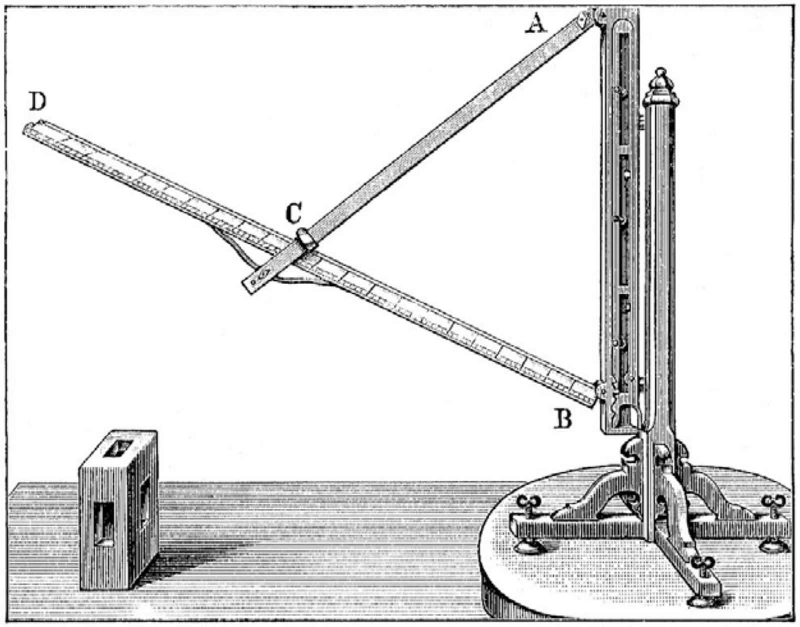
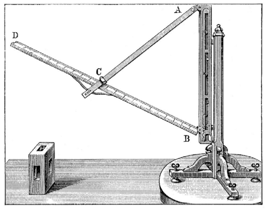
Triquetrum: Models of Computation for Workflows

Christopher Brooks,
University of California, Berkeley
Erwin De Ley,
iSencia, Belgium



EclipseCon NA 2016
Reston, VA
March 8, 2016



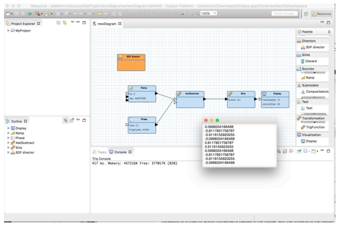
Triquetrum: *Models of Concurrent Computation for Workflow Management and Execution*

Definition:

Triquetrum is an Eclipse project that uses the Ptolemy II actor-oriented execution engine to provide run time semantics for use in workflows.

The Goal

There are already several Eclipse-based scientific workflow systems available, but many are specific to particular research domains. The combination of Eclipse/OSGi with Ptolemy's architecture for hierarchical and heterogeneous actor-based modeling, delivers a solid platform for a wide range of workflow applications.



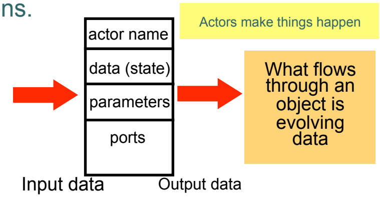
Triquetrum Screen Shot

Actors-Oriented Execution

In an actor system, data flows through actors.

Actors are inherently concurrent.

Actor oriented:


Actors make things happen
actor name
data (state)
parameters
ports
What flows through an object is evolving data
Input data
Output data

Models of Computation (MOCs)

A model of computation governs the semantics of the interaction, and thus imposes an execution-time discipline. Ptolemy II has implementations of many models of computation including Synchronous Data Flow, Kahn Process Networks, Discrete Event, Continuous Time, Synchronous/Reactive and Modal Models. Composing these can be very powerful.

Deliverables/Products:

1. A Ptolemy **II RCP model editor and execution runtime**, taking advantage of Ptolemy's features for heterogeneous and hierarchical models.
   a. The runtime must be easy to integrate in different environments, ranging from a personal RCP workbench to large-scale distributed systems.
   b. To that end we will deliver supporting APIs for local & remote executions, including support for debugging/breakpoints etc.
   c. The platform and RCP editor must be extensible with domain-specific components and modules.
   d. We will also deliver APIs to facilitate development of extensions, building on the features provided by Ptolemy and OSGi.

2. APIs and OSGi service implementations for **Task-based processing**. This would be a "layer" that can be used independently of Ptolemy, e.g. by other workflow/orchestration/sequencing software or even ad-hoc systems, interactive UIs etc.

3. **Supporting APIs and tools**, e.g. integration adapters to all kinds of things like external software packages, resource managers, data sources etc.

The problems being solved:

1. Promote integration of a workflow system in scientific software
2. Provide a correct-by-construction framework for workflow systems with useful features such as determinism.
3. Ptolemy is exploring IoT by combining Asynchronous Atomic Callbacks (AAC) with Actors. Triquetrum will make this work more reusable.
4. Ptolemy is used by Passerelle, which is used by the Eclipse DAWNScience project. However, Eclipselabs@google closing down and Passerelle needs a new home.
5. The Ptolemy II code base started in 1996, now is a good time to extract the core and make it reusable via OSGi.

Claudius Ptolemaus shown holding a triquetrum

Christopher Brooks <cxh@eecs.berkeley.edu> March 1, 2016

Applications of the Technology

Passerelle

- Workflows for control & data acquisition
- Automated telecom diagnosis and repair
- Start in Early 2000's
- Used in Synchrotron Soleil, Diamond Light Source, ESRF Proximus

Dawb -> DAWNScience

- 2009: start. 2010-2012: Using Passerelle 2014: an Eclipse Project
- Scientific Data Analysis, Visualization, Workflows

ICE (Integrated Computing Environment)

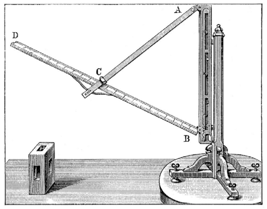
- Support for model setup, launching, analyzing, managing I/O data
- 2009: start. 2014: an Eclipse Project

Outcomes (so far)

- Eclipse project started in December 2015
- https://github.com/eclipse/triquetrum
- https://wiki.eclipse.org/Triquetrum (Downloads!)
- Triquetrum-dev mailing list (7! Users)

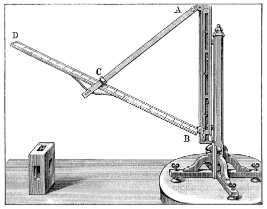
Who

- Erwin de Ley (iSencia)
  Project Lead, primary committer
- Christopher Brooks (UC Berkeley)
  Project lead, committer
- Jay Jay Billing (Oak Ridge Nat. Labs)
  Committer, Informal Mentor
- Alex McCaskey, Matt Gerring: Committers
- Jonas Helming, Wayne Beaton: Mentors



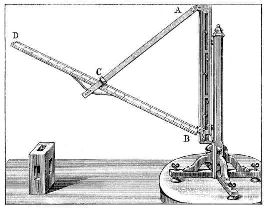
What is Triquetrum?

- Triquetrum is an Eclipse project that uses the Ptolemy II actor-oriented execution engine to provide **run time semantics for use in workflows**.
- The project started in 2015 as a project in the Eclipse Science Working Group.
- Triquetrum uses Ptolemy II as its execution engine.
- Triquetrum is named for the three sided astronomical instrument that Mr. Ptolemy is holding.
- **Tri**quetrum evokes Model-View-Controller.
- Pronounced tri-QUET-rum not ~~tri-QUEET-rum~~



Triquetrum Goals

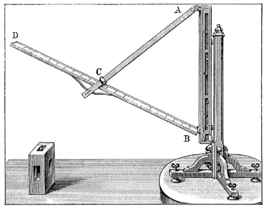
- Deliver an open platform for managing and deterministically executing scientific workflows
- Support a wide range of use cases:
  - Automated processes based on predefined models
  - Replaying ad-hoc research workflows based on a recording of user interactions
  - Allow users to define and execute small and large models
- Provide extension APIs and services with a focus on scientific workflows.
  - Currently interested organizations are big research institutions in materials research (synchrotrons), physics and engineering.



What can Workflow Systems do for Scientific Software Systems?

Workflow Systems benefit Scientific Software Systems as follows:

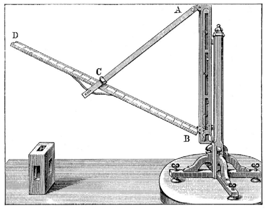
1. Make the steps in scientific processes visible
2. Models can be used for presentation and discussion.
3. Different roles with a common toolset: software engineers, model builders, model users etc.
4. Reuse!
5. Automating complex processes.
6. Crucial tool for advanced analytics on huge datasets
7. Integrates execution tracing, provenance data, etc.



Triquetrum workflows? How?

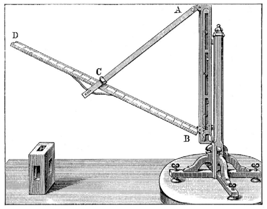
- The core of Triquetrum is an integration of Ptolemy II in an Eclipse and OSGi technology stack.
  - Ptolemy II (Berkeley, BSD License):
    "Ptolemy II is an open-source software framework supporting experimentation with actor-oriented design." (source: http://ptolemy.org)

- Triquetrum adds:
  a Rich Client Platform (RCP) editor
  + modularity & service-based design
  + possible integration of many interesting Eclipse frameworks and technologies.

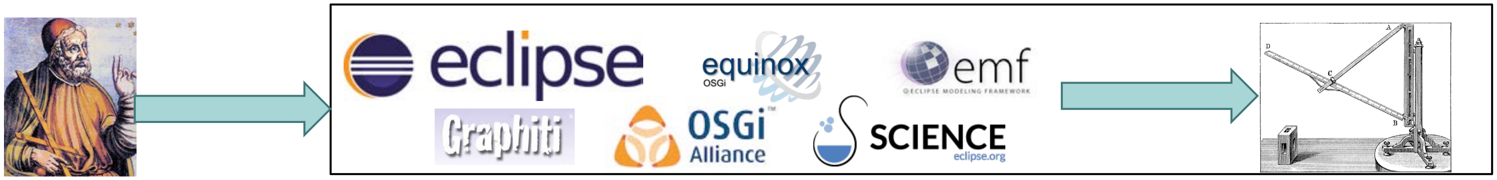


The results

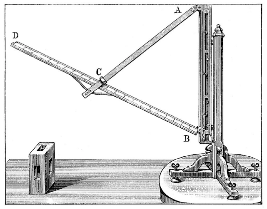
- The combination of Eclipse/OSGi with Ptolemy II delivers a solid platform for a wide range of workflow applications, especially scientific workflows.
- A powerful ecosystem for projects like Triquetrum comes from:
  - The modularity and dynamism offered by OSGi
  - The rich set of frameworks and technologies offered through the Eclipse Foundation,
  - and the community of the Eclipse Science Working Group



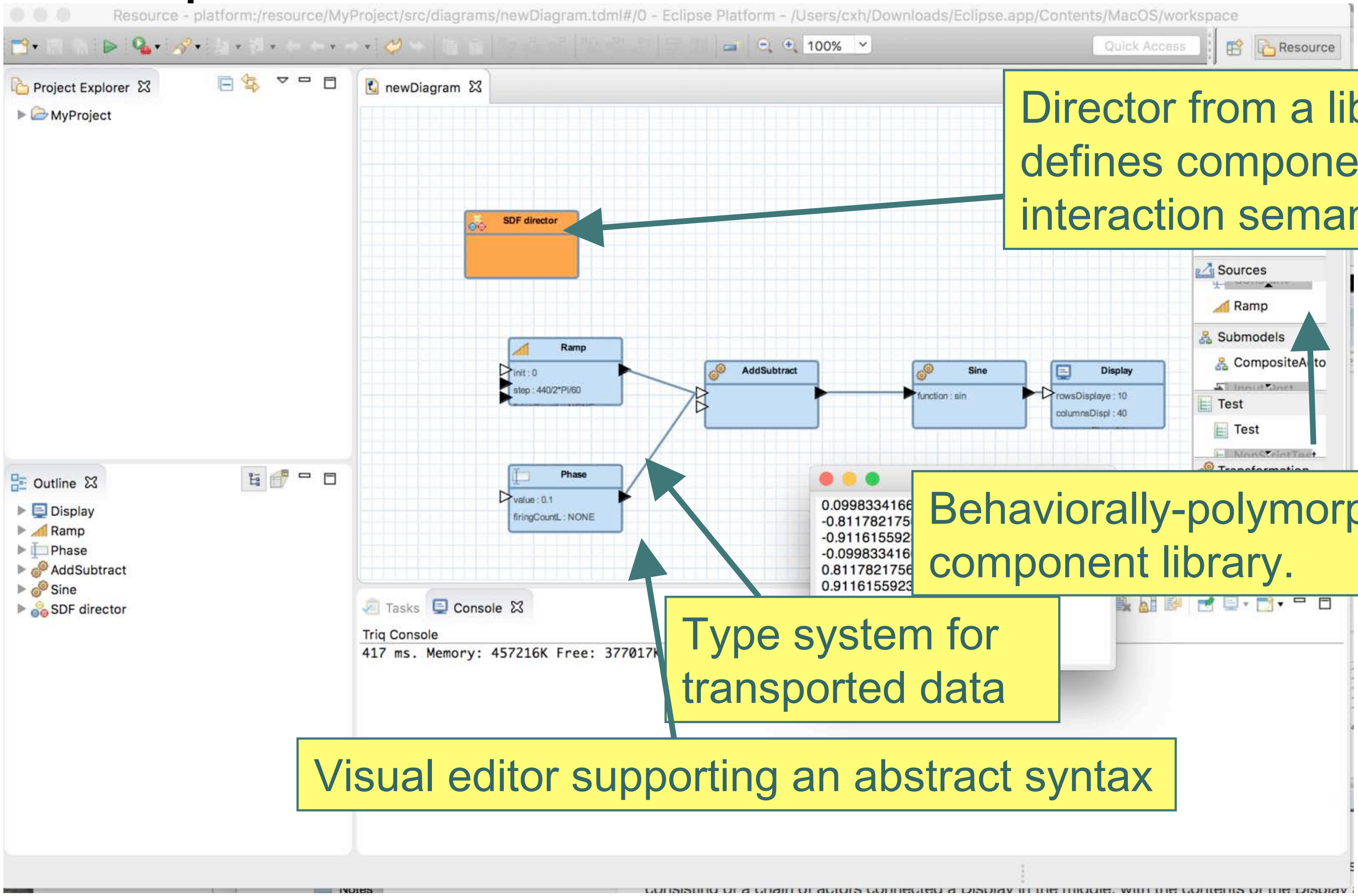
Triquetrum is standing on the shoulders of giants



- Ptolemy II (Prof. Edward A. Lee and many others)
- The main Eclipse frameworks that are used for the workflow editor are:
  - Equinox, Rich Client Platform (RCP),… : the traditional stuff for RCP apps.
  - Graphiti: for the graphical workflow editor
  - Eclipse Modeling Framework (EMF): to define a meta-model for Ptolemy II's model elements like Actors, CompositeActors, Parameters, Directors etc., for use by the Graphiti editor.
  - EMF Forms: to define Actor configuration forms during the workflow design



Triquetrum: A laboratory for experimenting with actor-oriented modeling



```
0.0998334166
-0.811782175
-0.911615592
-0.099833416
0.8117821756
0.9116155923
```

Triq Console

```
417 ms. Memory: 457216K Free: 377017K
```

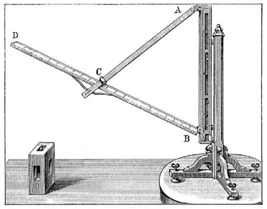
Director from a library defines component interaction semantics

Behaviorally-polymorphic component library.

Type system for transported data

Visual editor supporting an abstract syntax

(Based on a Ptolemy Slide by Edward A. Lee)

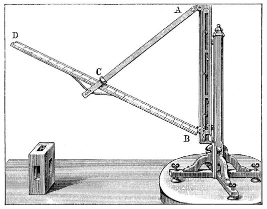


Actor Model

Carl Hewitt

(source: Wikipedia, Jean-Baptist LABRUNE CC-BY-2.0)

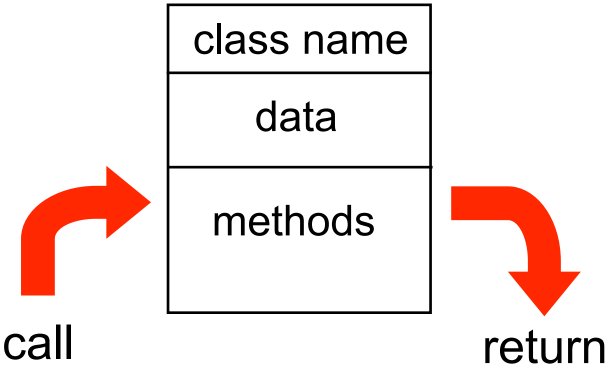
- “The **actor model** in computer science is a mathematical model of concurrent computation that treats "actors" as the universal primitives of concurrent computation: in **response** to a message that it receives, an actor can **make local decisions**, **create** more actors, **send** more messages, and determine how to respond to the next message received.” (Wikipedia)
- “The actor model originated in 1973” (Wikipedia) and cites a paper by Carl Hewitt (who sometimes attends EclipseCon) and Peter Bishop.



Object Oriented vs. Actor Oriented

Objects are not concurrent by design and need something like threads to be concurrent

The established: Object-oriented:

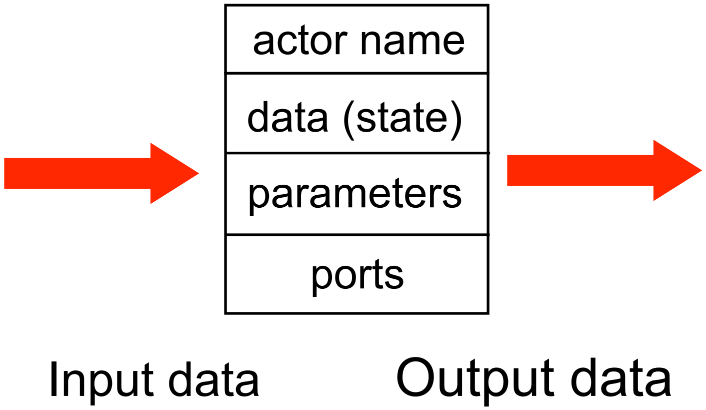


What flows through an object is sequential control

Things happen to objects

The alternative: Actor oriented:

Actors make things happen



What flows through an object is evolving data, which matches the nature of workflows

Actors are concurrent

(Based on a slide by: Edward A, Lee)

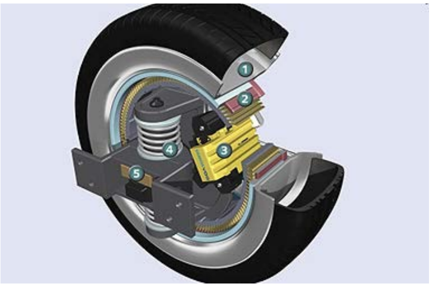
Who cares?

Cyber-Physical Systems (CPS): *Orchestrating networked computational resources with physical systems*

Internet of Important Things (IoIT)

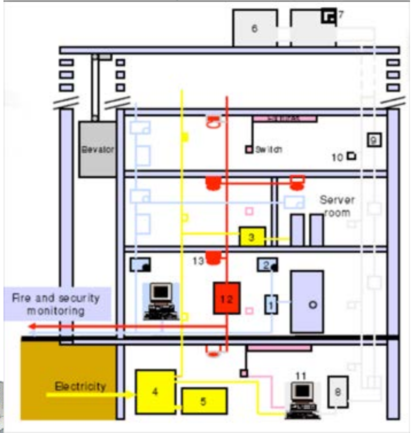
Scientific Workflows

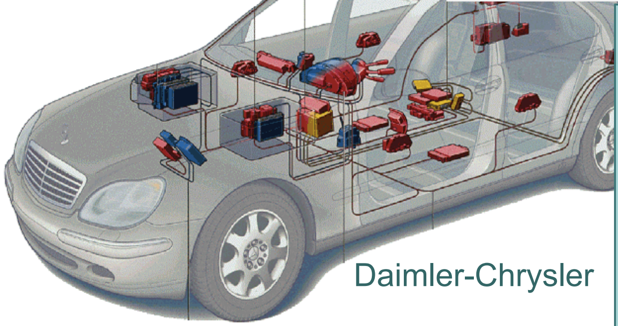
Automotive



E-Corner, Siemens

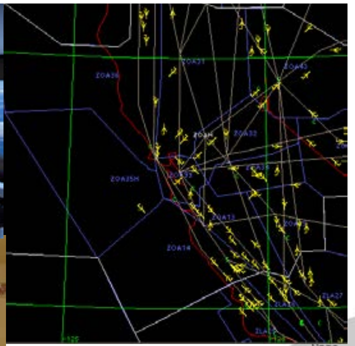
Building Systems





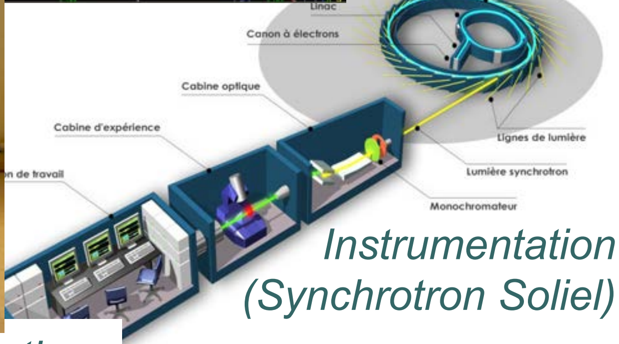
Daimler-Chrysler

Avionics



Transportation (Air traffic control at SFO)

Biomedical

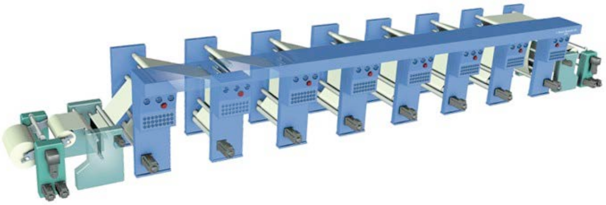


Instrumentation (Synchrotron Soliel)

Power generation and distribution

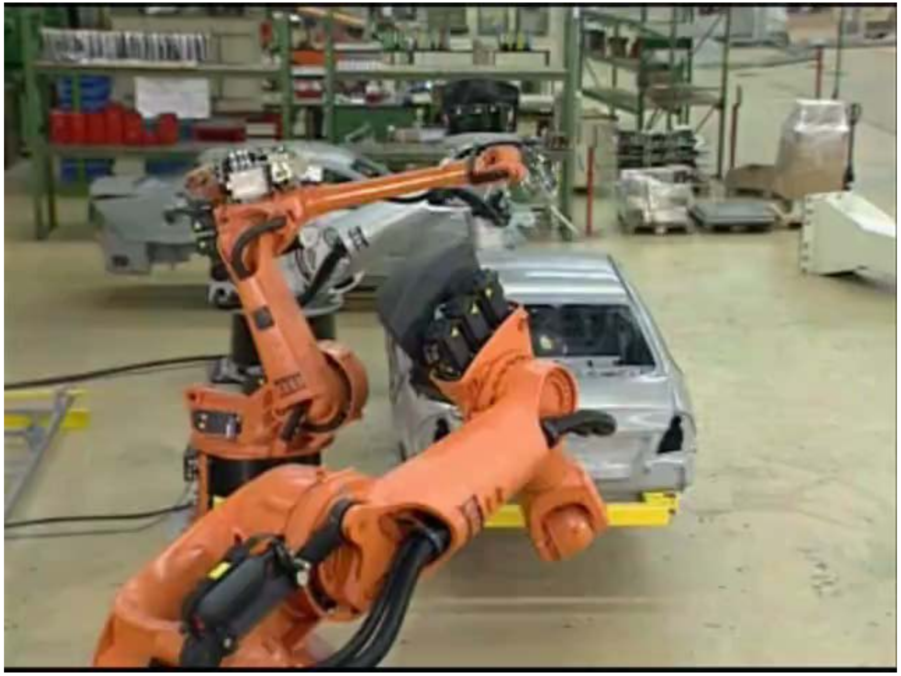
Courtesy of General Electric

High speed printing



Bosch Rexroth

Factory automation



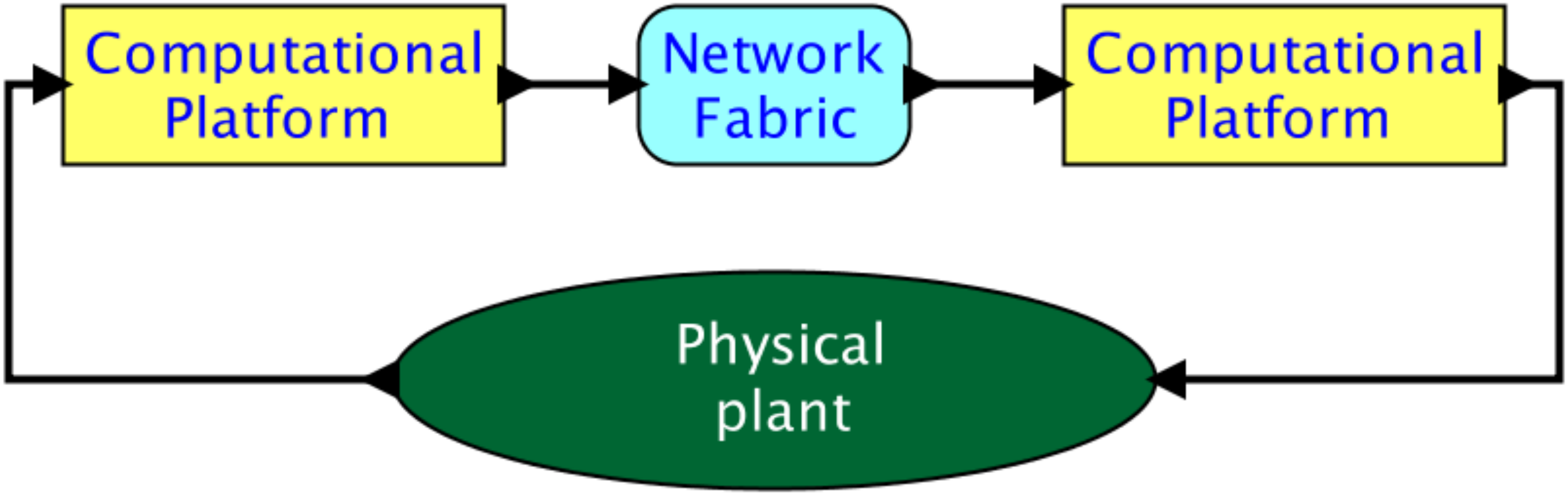
Courtesy of Kuka Robotics Corp.

(Source: Edward A. Lee)

Schematic of a simple Cyber-Physical System (CPS)

Internet of Things (IoT)

Scientific Workflows



(Source: Edward A. Lee)

Repeatability requires

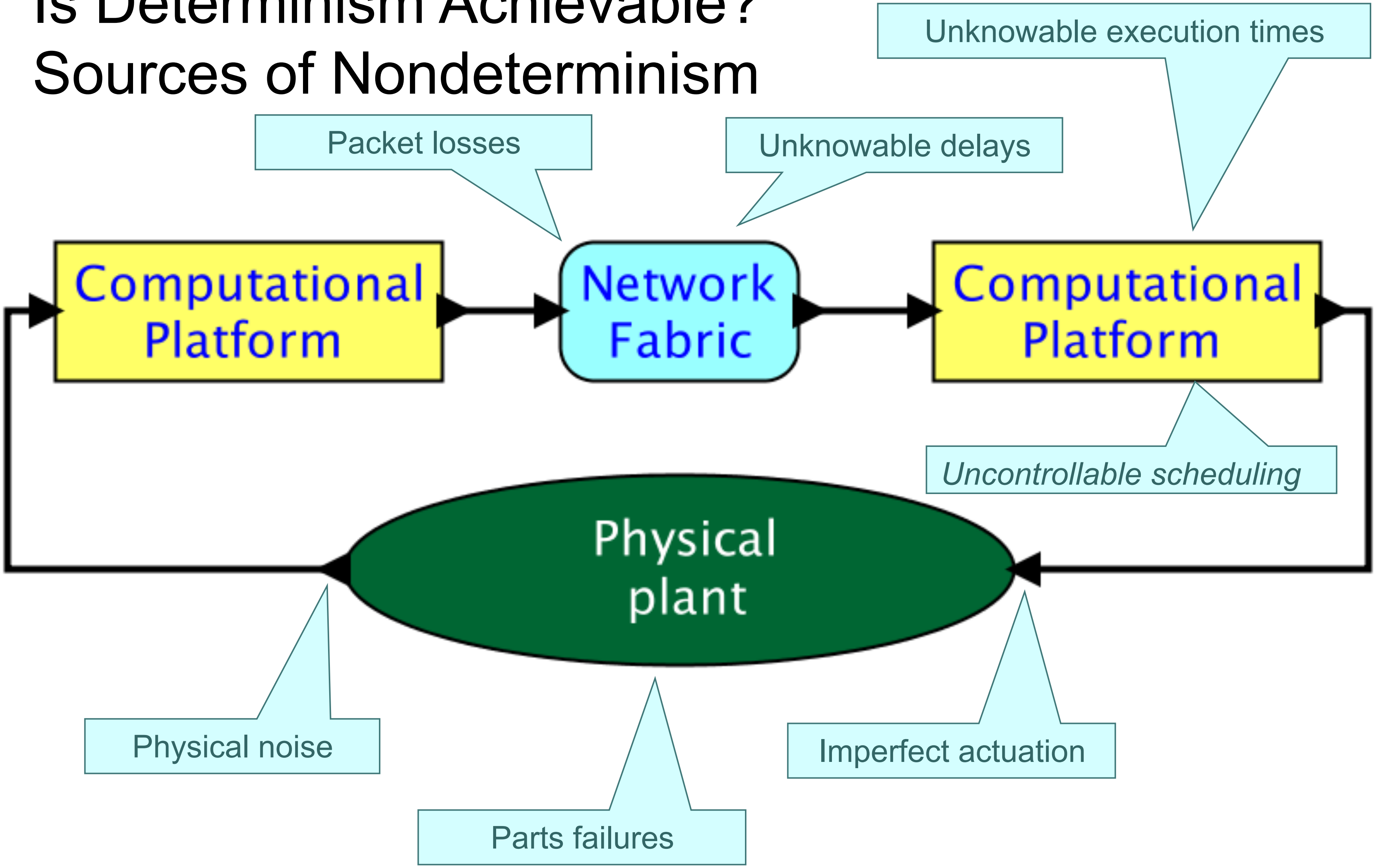
Determinacy

or

The same inputs yield the same outputs.

(Source: Edward A. Lee)

Is Determinism Achievable?
Sources of Nondeterminism



Computational Platform

Network Fabric

Computational Platform

Physical plant

Packet losses

Unknowable delays

Unknowable execution times

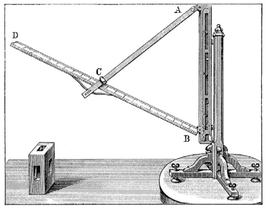
Uncontrollable scheduling

Physical noise

Parts failures

Imperfect actuation

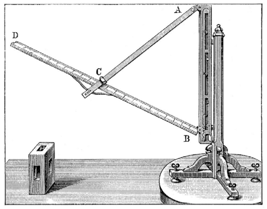
(Source: Edward A. Lee)



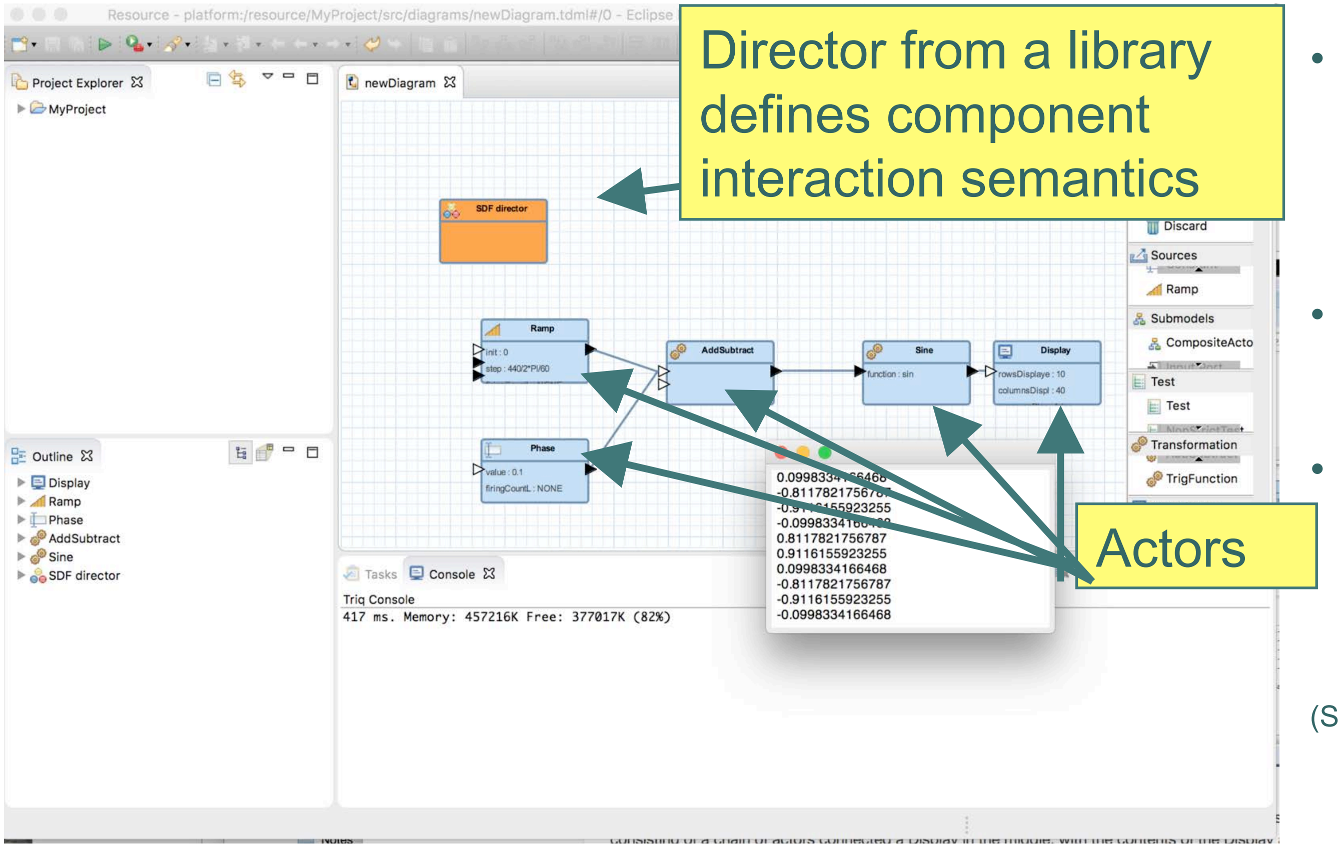
Ensure properties like determinancy with **Models of Computation**

A model of computation governs the semantics of the interaction, and thus imposes an execution-time discipline.

Composing Models of Computation can be very powerful.



To support **determinacy**, Triquetrum uses the **Actor Model** to implement **Models of Computation**


Resource - platform:/resource/MyProject/src/diagrams/newDiagram.tdml#/0 - Eclipse

```
0.0998334166468
-0.8117821756787
-0.9116155923255
-0.0998334166468
0.8117821756787
0.9116155923255
0.0998334166468
-0.8117821756787
-0.9116155923255
-0.0998334166468
```

Triq Console

```
417 ms. Memory: 457216K Free: 377017K (82%)
```

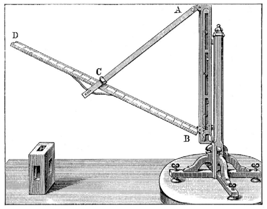
Director from a library defines component interaction semantics

Actors

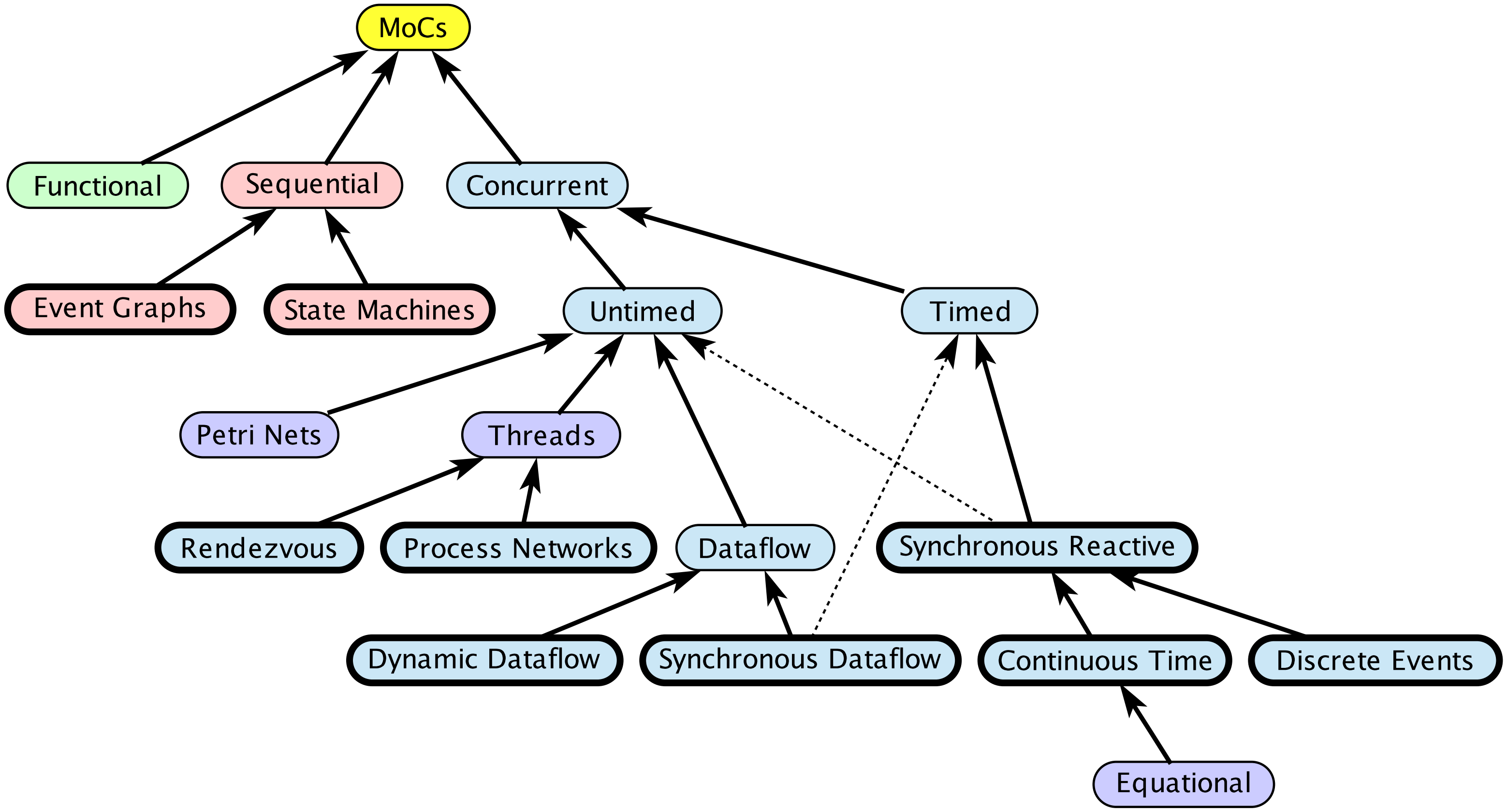
Actors are a good abstraction because:

- **Data flows through actors**, which matches workflows.
- The actor model features a **concurrency** model
- Actor abstract semantics allow the **combination** of different models of computation.

(Source: Marten Lohstroh)



A quick tour of Models of Computation



MoCs

Functional

Sequential

Concurrent

Event Graphs

State Machines

Untimed

Timed

Petri Nets

Threads

Rendezvous

Process Networks

Dataflow

Synchronous Reactive

Dynamic Dataflow

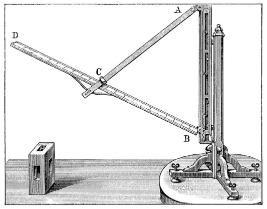
Synchronous Dataflow

Continuous Time

Discrete Events

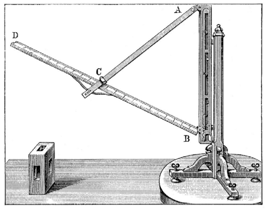
Equational

(Source: Ptolemy Book)

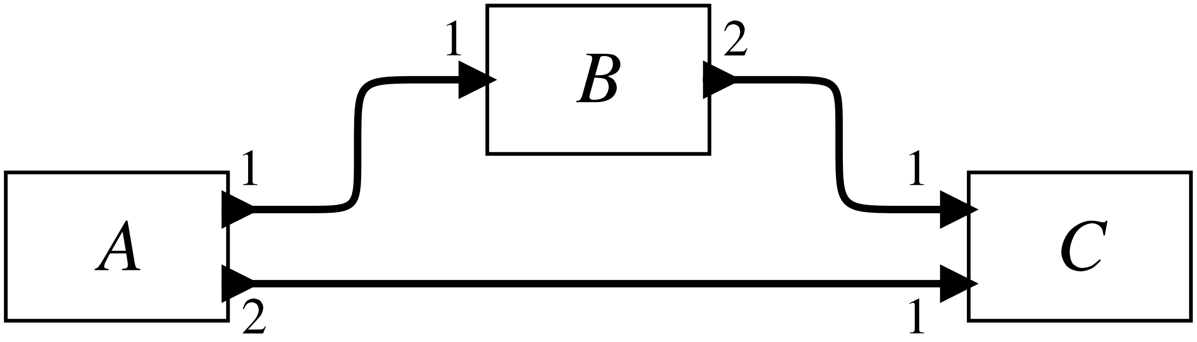


Synchronous Data Flow (SDF)

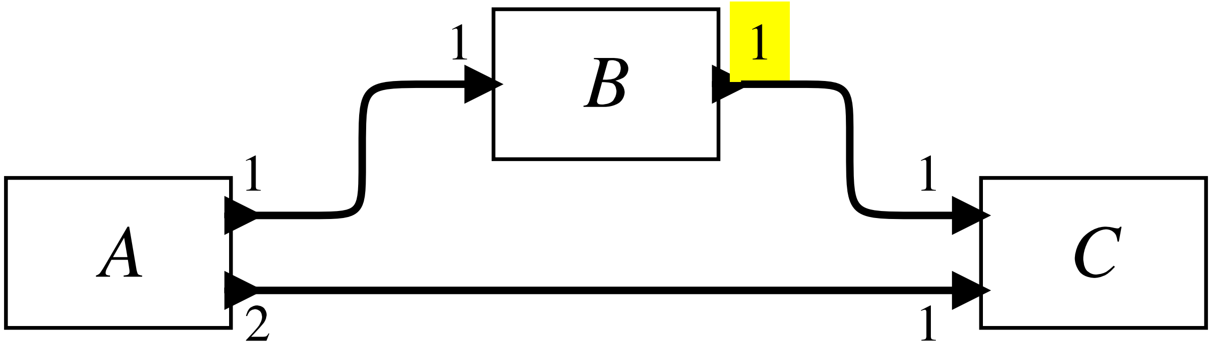
- “When an actor is executed in SDF, it consumes a fixed amount of data from each input port, and produces a fixed amount of data to each output port.” (Ptolemy 2014)
- SDF models are easy to check for deadlock and bounded buffer size.
- SDF schedules are easy to compute
- SDF is usually untimed, but can be timed.
- Dataflow is really good for streaming systems.



SDF Firing Schedules

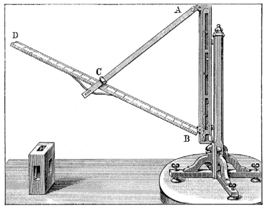


The actor firing schedule A;B;C;C can be repeated forever and with a bounded buffer size



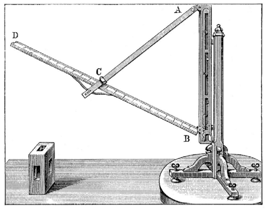
There is no actor firing schedule for this model that ensures a bounded buffer size

(Source: Ptolemy Book)



Process Networks (PN)

- PN is a superset of Synchronous Data Flow
- Each actor is a thread, however too much communication between actors can hurt performance.
- Writing to the queues always succeeds immediately, while reading from an empty queue blocks the reader process. (This ensures determinacy)
- PN is useful for modeling processes that communicate asynchronously



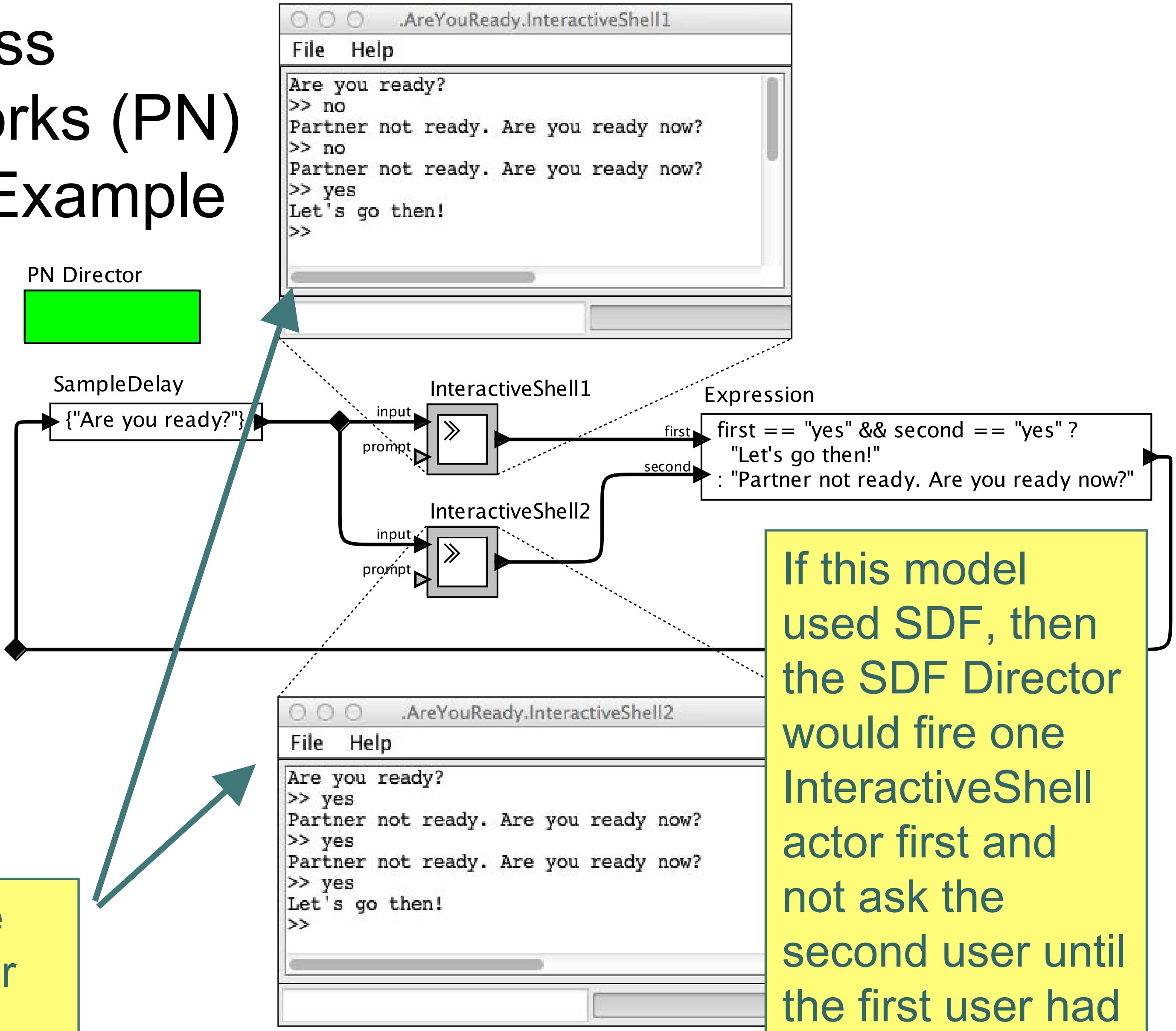
Process Networks (PN) Chat Example

PN is useful when actors do not return immediately

This model seeks concurrence from both users before proceeding

It asks each user if they are ready and after both users have responded "yes", the model responds "Let's go then!"

Each instance of the InteractiveShell actor is run in a separate thread

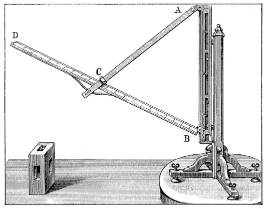


```
Are you ready?
>> no
Partner not ready. Are you ready now?
>> no
Partner not ready. Are you ready now?
>> yes
Let's go then!
>>
```

```
Are you ready?
>> yes
Partner not ready. Are you ready now?
>> yes
Partner not ready. Are you ready now?
>> yes
Let's go then!
>>
```

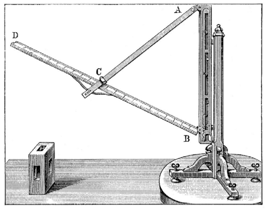
If this model used SDF, then the SDF Director would fire one InteractiveShell actor first and not ask the second user until the first user had responded.

(Source: Ptolemy Book)

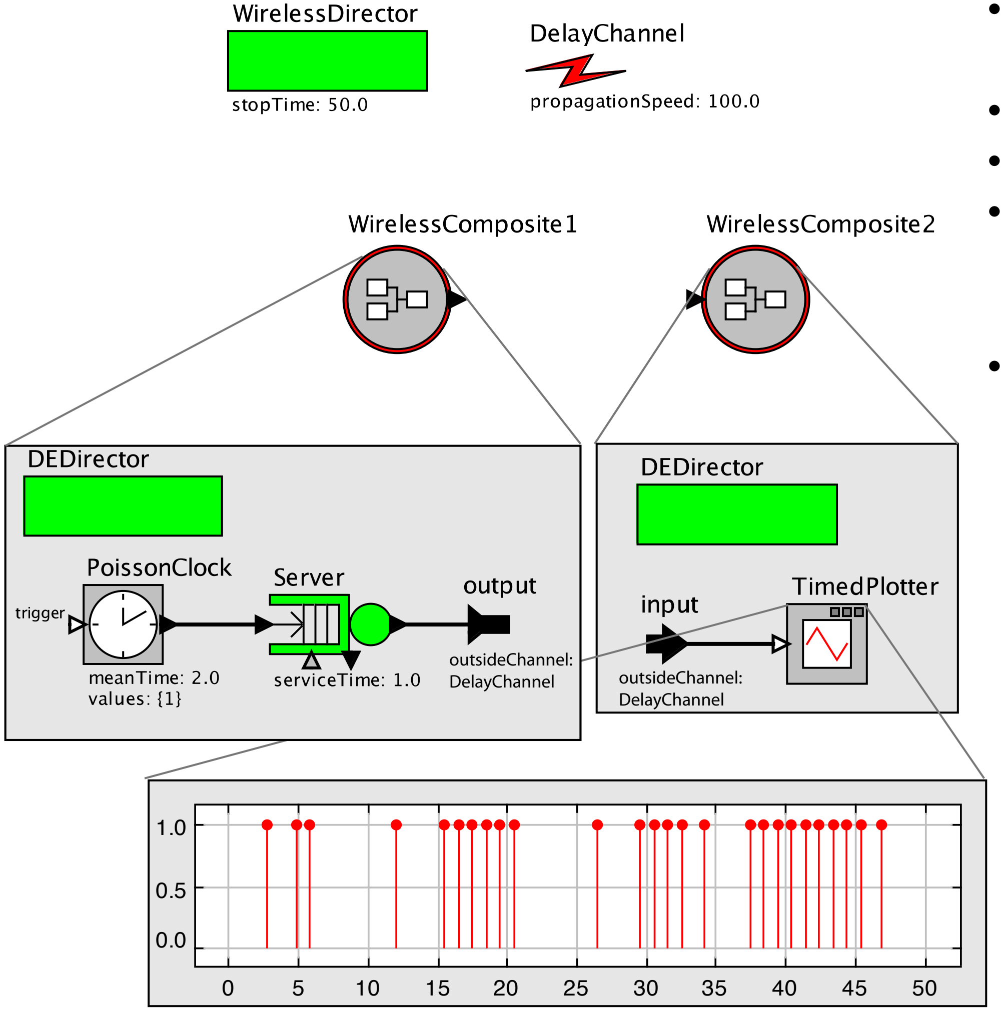


Discrete Event:
A timed model of computation

- Actors communicate through events placed on a time line.
- Events have a value and a time stamp.
- Useful for modeling complex systems over time like digital circuits and financial systems.

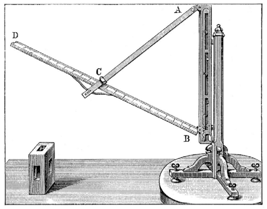


Wireless Example



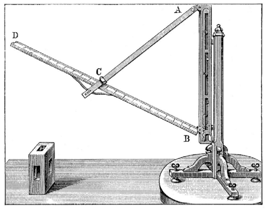
(Source: Ptolemy Book)

- This model contains **a WirelessDirector**, which extends the Discrete Event Director
- WirelessComposite1 has an output port
- WirelessComposite2 has an input port
- They are not directly connected, but they send values with timestamps via a Delay Channel
- The **position** of the composite icons on the screen **affects the communication** between them.

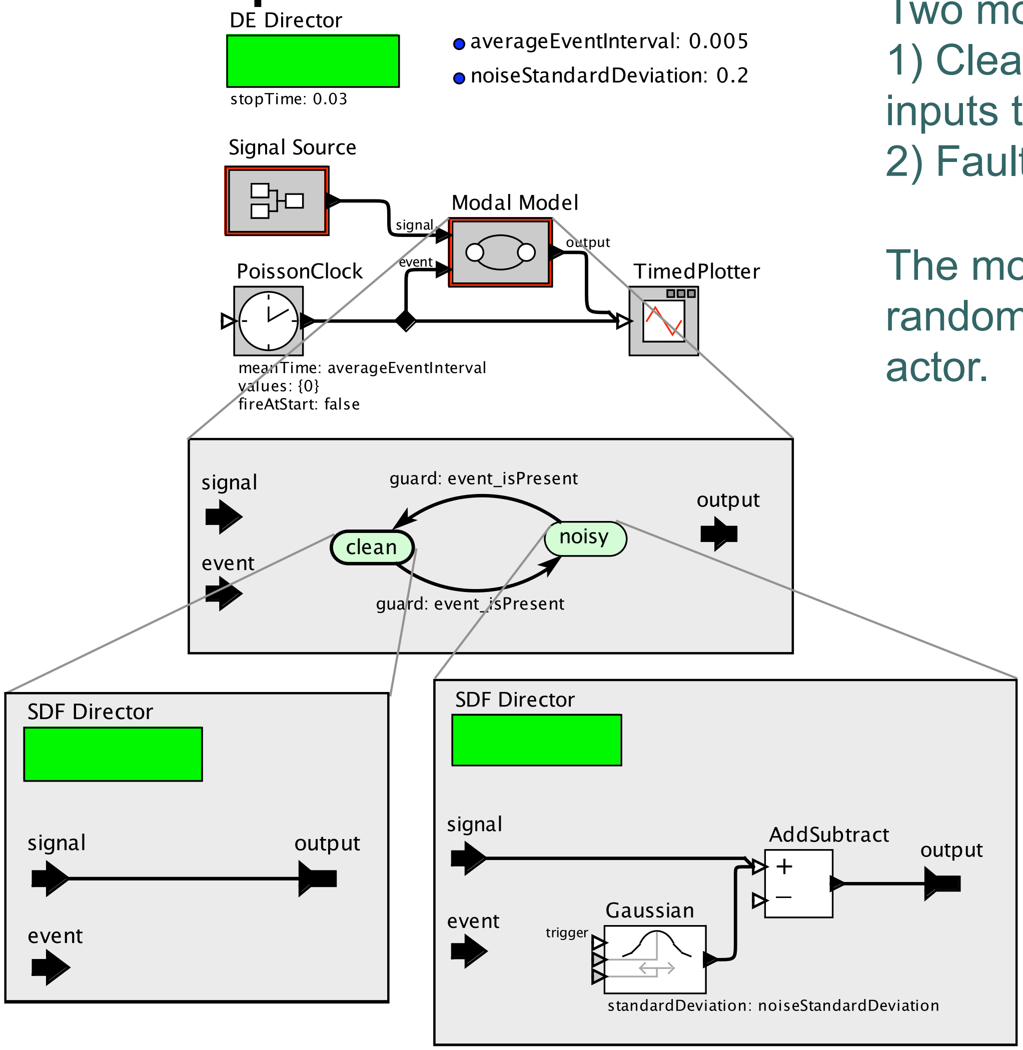


Modal Models

- "A modal model is an explicit representation of a finite set of behaviors (or modes) and the rules that govern transitions between them. The rules are captured by a finite state machine (FSM)." (Ptolemy 2014)

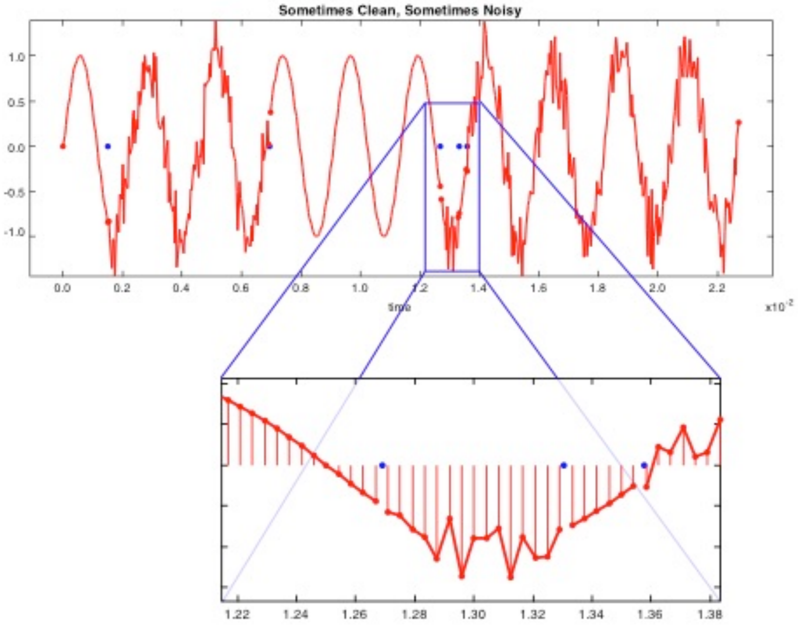


A Modal Model with two modes

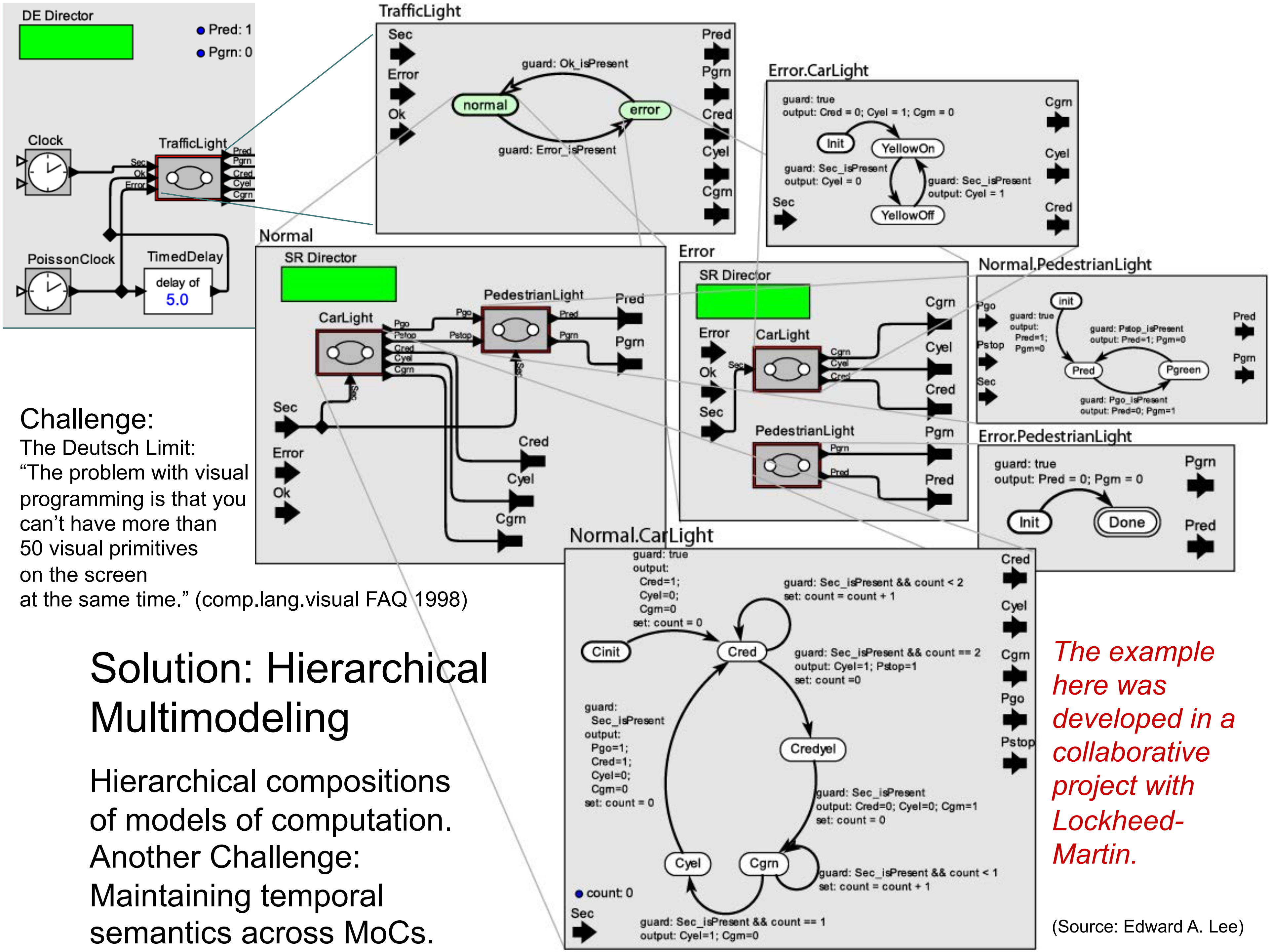


Two modes:
1) Clean operating mode, in which it passes inputs to the output unchanged, and
2) Faulty mode, in which it adds Gaussian noise.

The model switches between these modes at random times determined by the PoissonClock actor.



(Source: Ptolemy Book)



Challenge:

The Deutsch Limit:
"The problem with visual programming is that you can't have more than 50 visual primitives on the screen at the same time." (comp.lang.visual FAQ 1998)

Solution: Hierarchical Multimodeling

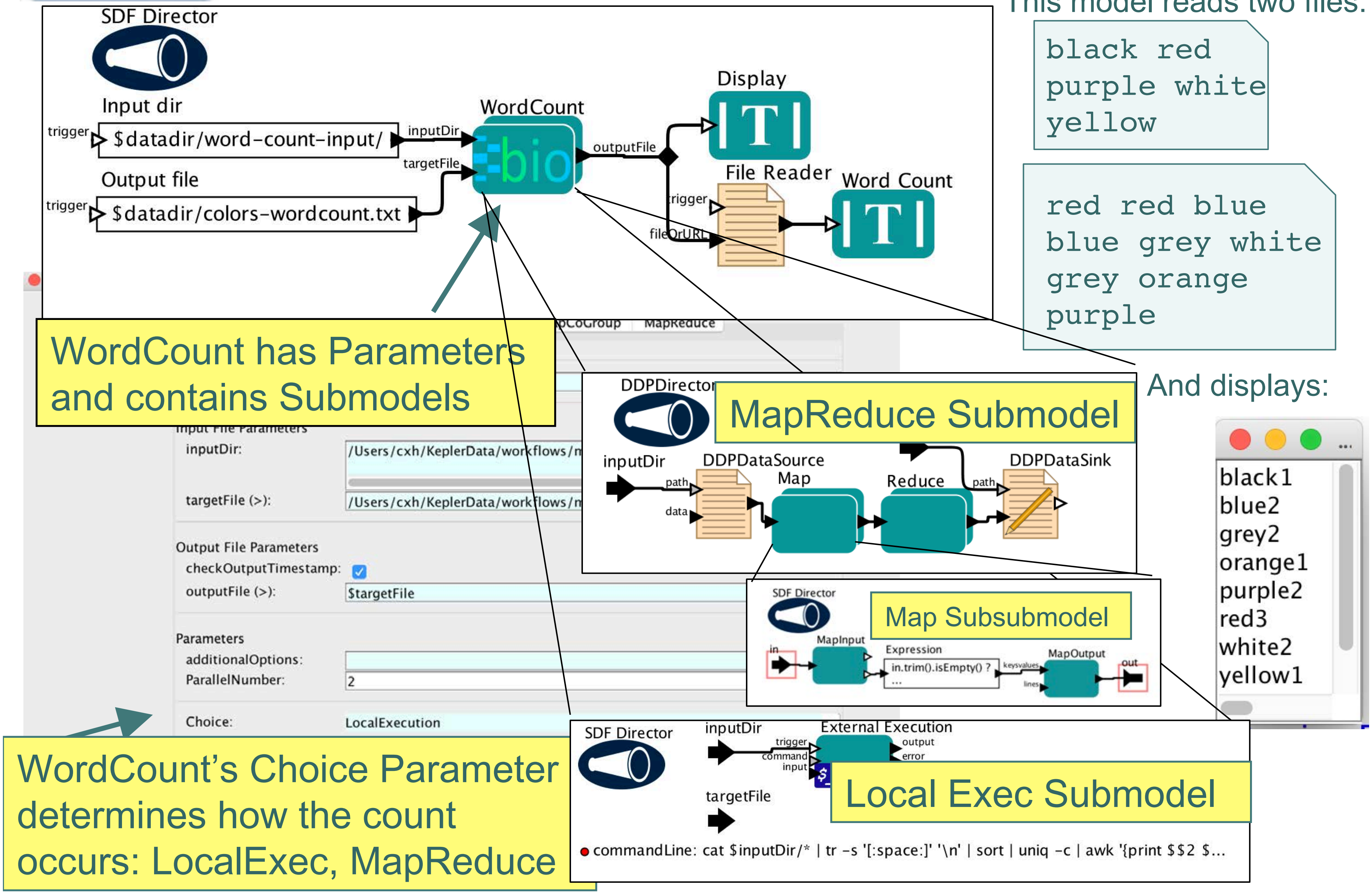
Hierarchical compositions of models of computation.

Another Challenge: Maintaining temporal semantics across MoCs.

The example here was developed in a collaborative project with Lockheed-Martin.

(Source: Edward A. Lee)

- The Kepler Scientific Workflow System Uses Ptolemy
- This model uses hierarchy to determine words are counted



This model reads two files:

```
black red
purple white
yellow
```

```
red red blue
blue grey white
grey orange
purple
```

And displays:

```
black1
blue2
grey2
orange1
purple2
red3
white2
yellow1
```

WordCount has Parameters and contains Submodels

MapReduce Submodel

Map Subsubmodel

Local Exec Submodel

WordCount's Choice Parameter determines how the count occurs: LocalExec, MapReduce

inputDir: /Users/cxh/KeplerData/workflows/m

targetFile (>): /Users/cxh/KeplerData/workflows/m

Output File Parameters

checkOutputTimestamp: ☑

outputFile (>): $targetFile

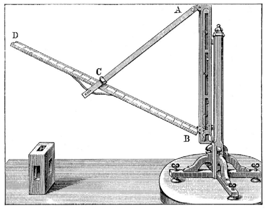
Parameters

additionalOptions:

ParallelNumber: 2

Choice: LocalExecution

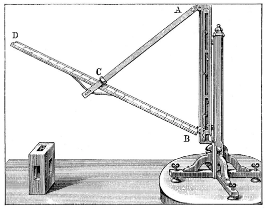
● commandLine: cat $inputDir/* | tr -s '[:space:]' '\n' | sort | uniq -c | awk '{print $$2 $...



Examples of Actor-Oriented “Languages”

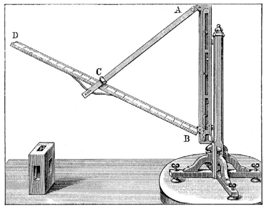
- Akka (message based and asynchronous, part of Scala)
- CAL (textual dataflow actors, used by a MPEG working group)
- **Calvin** (dataflow, Python**, Apache 2.0 License**, Ericcson)
- CORBA event service (distributed push-pull)
- ROOM and UML-2 (dataflow, Rational, IBM)
- VHDL, Verilog (discrete events, Cadence, Synopsys, ...)
- **LabVIEW** (structured dataflow, National Instruments)
- **Modelica** (continuous-time, constraint-based, Commercial and **Open Source** licenses, Linkoping)
- **Node-RED** (event-driven browser-based, Node.js, **Apache 2.0 License**, IBM)
- OPNET (discrete events, Opnet Technologies)
- SDL (process networks)
- Occam (rendezvous)
- **Simulink** (Continuous-time, The MathWorks)
- SPW (synchronous dataflow, Cadence, CoWare)
- Taverna (Lambda Calculus, LGLP -> **Apache 2.0 License**)
- …

(Source: Edward A, Lee)



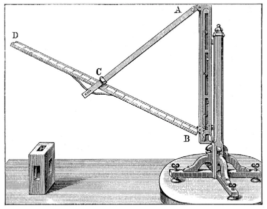
Triquetrum Scope

1. Rich Client Platform (RCP) **Editor**
   (Uses Ptolemy as execution engine)
2. APIs and OSGi service implementations
   for **Task-based processing** (usable w/o Ptolemy)
3. **Support for APIs and Tools**
   (Adapters to external packages etc.)



Scope: 1) Rich Client Platform (RCP) **Editor** (Uses Ptolemy as execution engine)

- This task is a driver for packaging Ptolemy using OSGi
- **Done**: Tycho build, Hudson CI, Wiki, Downloads available
- In progress: supporting hierarchy, setting parameters, reading Ptolemy models, running models
- To do: Lots!
  - Support configurations of different actor collections
  - Lots of UI work
  - Use Eclipse Layout Kernel (ELK) from Univ. of Kiel
  - After IP approval: Ptolemy source -> Eclipse GitHub



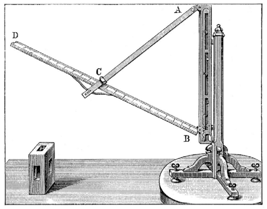
Scope: 2) APIs and OSGi service implementations for **Task-based processing** (usable w/o Ptolemy)

- Goals:
  - Represent a work item,
  - trace the progress of its processing
  - and represent the results
- Deliverables
  - Definition of Service APIs
  - Storage and consultation of traces that include timing, success and failure events.

This is an example of **a coordination language**, which is separate from a **computation language**. See Gelernter, Carriero, "Coordination languages and their significance," ACM, 1992

AKA **Data Provenance**: documenting data and processes to enable reproduction of an experiment See Altintas et al, "Provenance Collection Support In the Kepler Scientific Workflow System", 2006.)

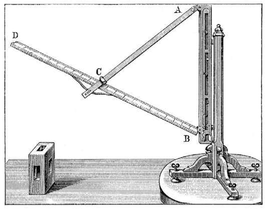
(Source: Erwin de Ley)



Task-based Processing: Task Model Details

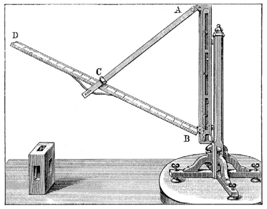
- A Task is created and then handed to a service broker that is responsible for finding a service implementation that either does the actual processing or further delegates the task. (**Coordination Language**)
- During processing (**Computation Language**), the status may change. These time stamped events are stored. (**Data Provenance**)
- If/when the task completes successfully, the results are stored.
  - The results may be simple (success/failure) or a large data set
  - Triquetrum will provide the storing of blocks of named values linked to the task that produced them.

(Source: Erwin de Ley)



Scope: 3) **Support for APIs and Tools** (Adapters to external packages etc.)

- Integration adapters to all kinds of things like external software packages, resource managers, data sources etc.
  - Example: An integration of that API with DAWNSci's Python AnalysisRPC – a Python-to-Java bridge
  - Example: Some trivial implementations to connect to SOAP web-services



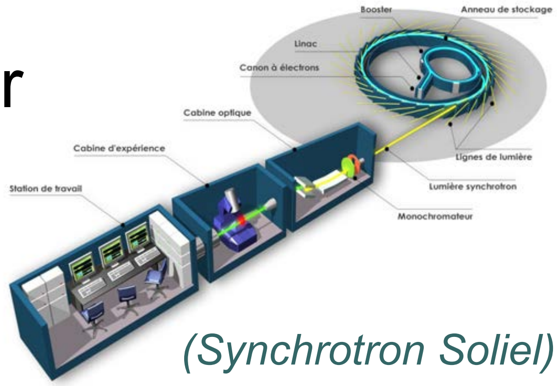
Passerelle: Ptolemy Inside Now, Triquetrum Inside Soon

passerelle

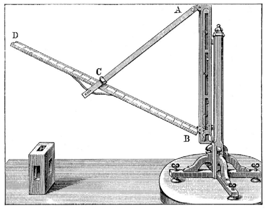
iSencia BELGIUM

- Workflows for control & data acquisition
- Automated telecom diagnosis and repair
- Start in early 2000's
- Used in Synchrotron Soleil,
  Diamond Light Source,
  European Synchrotron Radiation Facility (ESRF),
  Proximus (a Belgian telecom provider)
- Ptolemy II is the execution engine used by Passerelle
- One driver for moving Triquetrum to Eclipse:
  Passerelle was hosted at Eclipselabs@google, which closed down and Passerelle needed a new home.

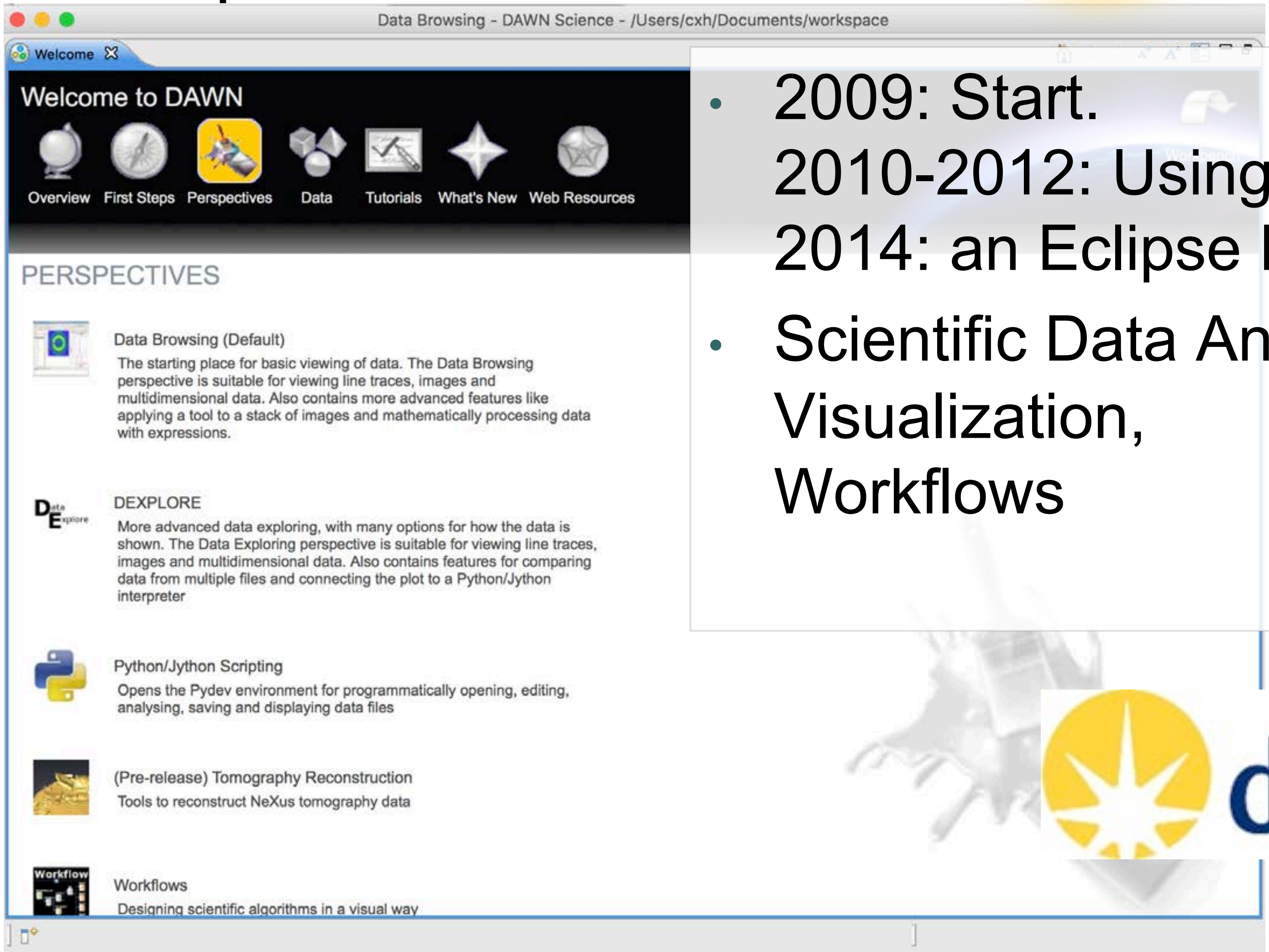
diamond



(Synchrotron Soliel)



Dawb -> DAWNSci


Data Browsing - DAWN Science - /Users/cxh/Documents/workspace

PERSPECTIVES

Data Browsing (Default)

The starting place for basic viewing of data. The Data Browsing perspective is suitable for viewing line traces, images and multidimensional data. Also contains more advanced features like applying a tool to a stack of images and mathematically processing data with expressions.

DEXPLORE

More advanced data exploring, with many options for how the data is shown. The Data Exploring perspective is suitable for viewing line traces, images and multidimensional data. Also contains features for comparing data from multiple files and connecting the plot to a Python/Jython interpreter

Python/Jython Scripting

Opens the Pydev environment for programmatically opening, editing, analysing, saving and displaying data files

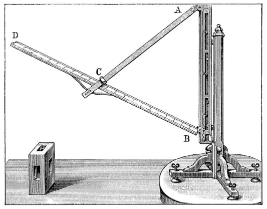
(Pre-release) Tomography Reconstruction

Tools to reconstruct NeXus tomography data

Workflows

Designing scientific algorithms in a visual way

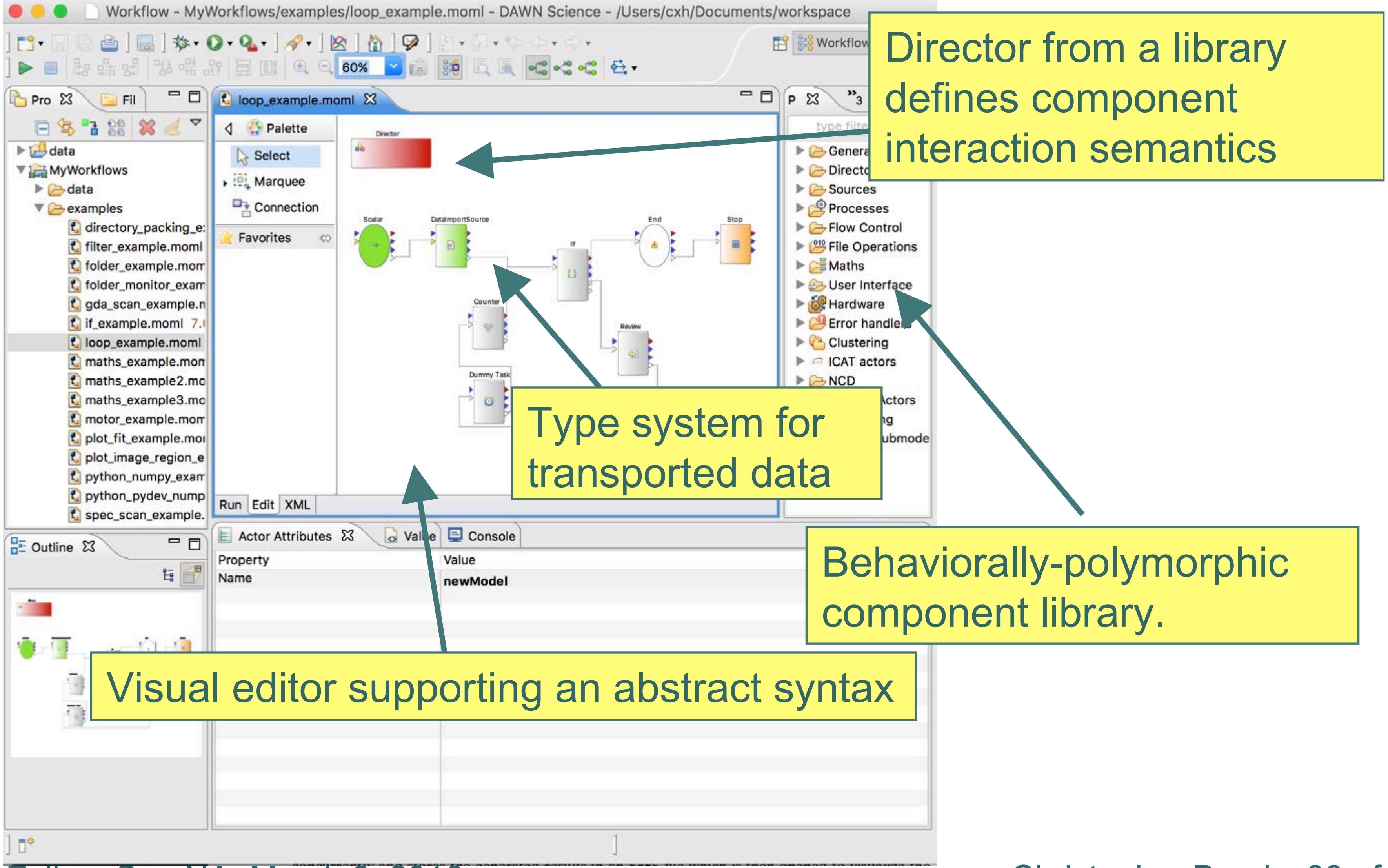
- 2009: Start.
  2010-2012: Using Passerelle
  2014: an Eclipse Project
- Scientific Data Analysis,
  Visualization,
  Workflows



DAWNSci: Block Diagrams vs. “Processing”

- Two input mechanisms for visualization, one that uses Passerelle block diagrams, the other called ‘Processing’, which allows a user to create a sequence of actors and run them.
- Processing is currently more popular because the block diagrams were not stable and user friendly.
- The actor type system could help by creating a Ptolemy graph and executing it using a Passerelle director.

DAWNSci Passerelle Block Diagram


Workflow - MyWorkflows/examples/loop_example.moml - DAWN Science - /Users/cxh/Documents/workspace

Property	Value
Name	newModel

Director from a library defines component interaction semantics

Type system for transported data

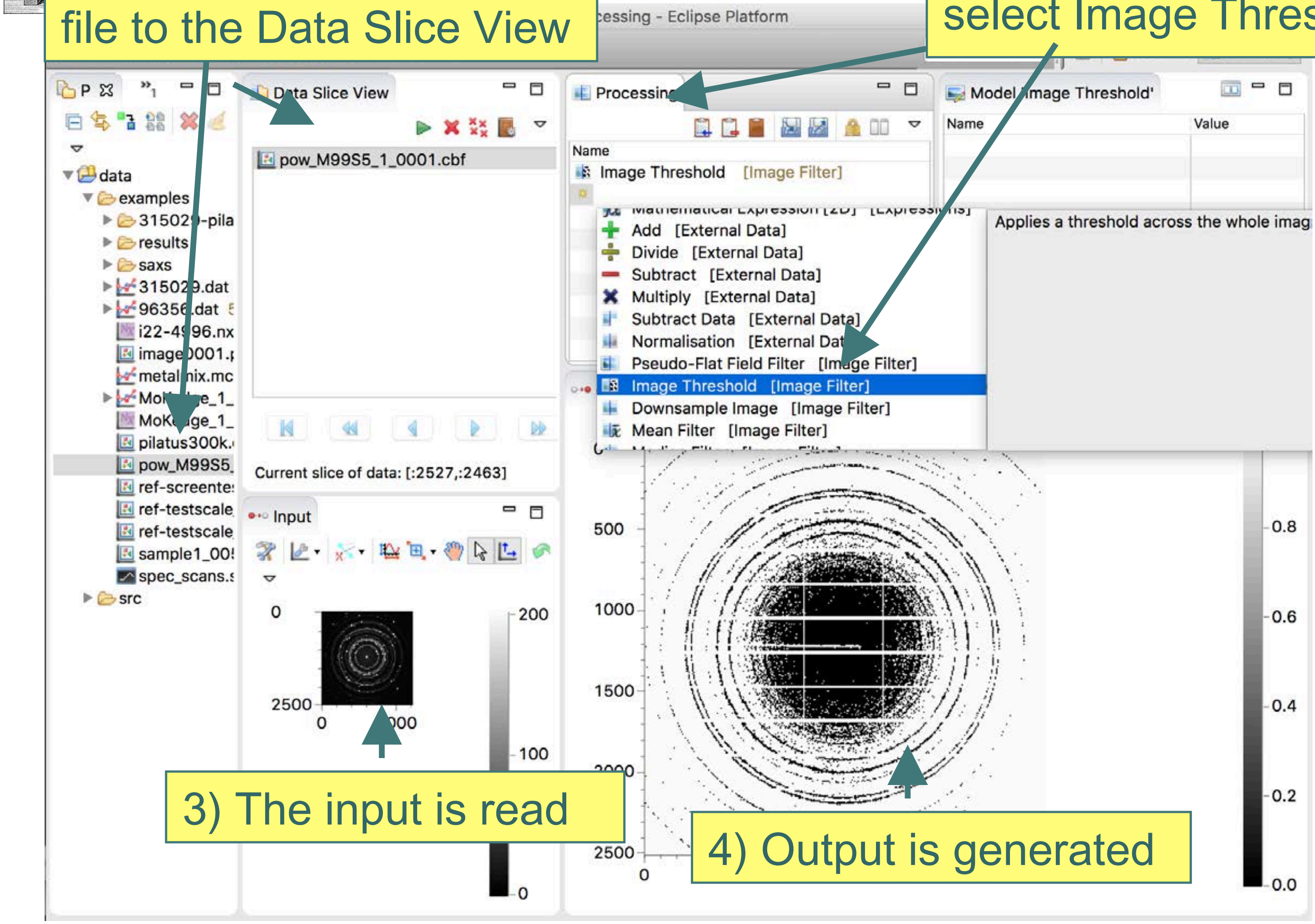
Behaviorally-polymorphic component library.

Visual editor supporting an abstract syntax

DAWNSci: Processing Perspective

1) Click and drag a data file to the Data Slice View

2) In the Processing View, select Image Threshold

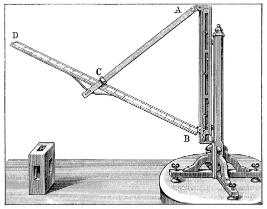


Current slice of data: [:2527,:2463]

Applies a threshold across the whole imag

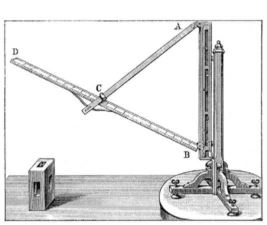
3) The input is read

4) Output is generated



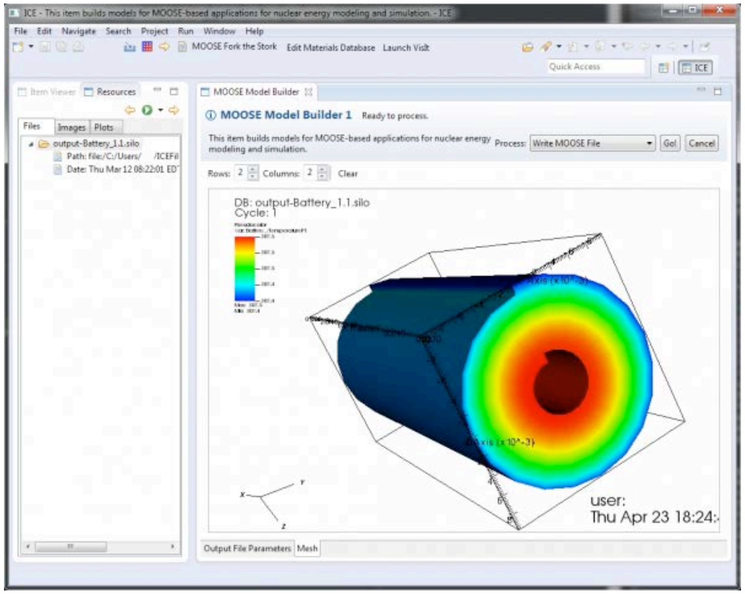
Ask not what you can do for Triquetrum, instead ask what Triquetrum can do for you

Matt Gerring (Diamond Light Source/DAWNSci):
“Triquetrum can deliver a stable user interface which helps the user to build legal graphs and create efficient data analysis pipelines”



ICE (Integrated Computing Env.)

- Support for model setup, launching, analyzing, managing I/O data
- 2009: start.
- 2014: an Eclipse Project


ICE - This item builds models for MOOSE-based applications for nuclear energy modeling and simulation. - ICE

MOOSE Model Builder 1 Ready to process.

This item builds models for MOOSE-based applications for nuclear energy modeling and simulation.

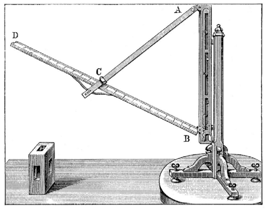
Process: Write MOOSE File

Rows: 2 Columns: 2

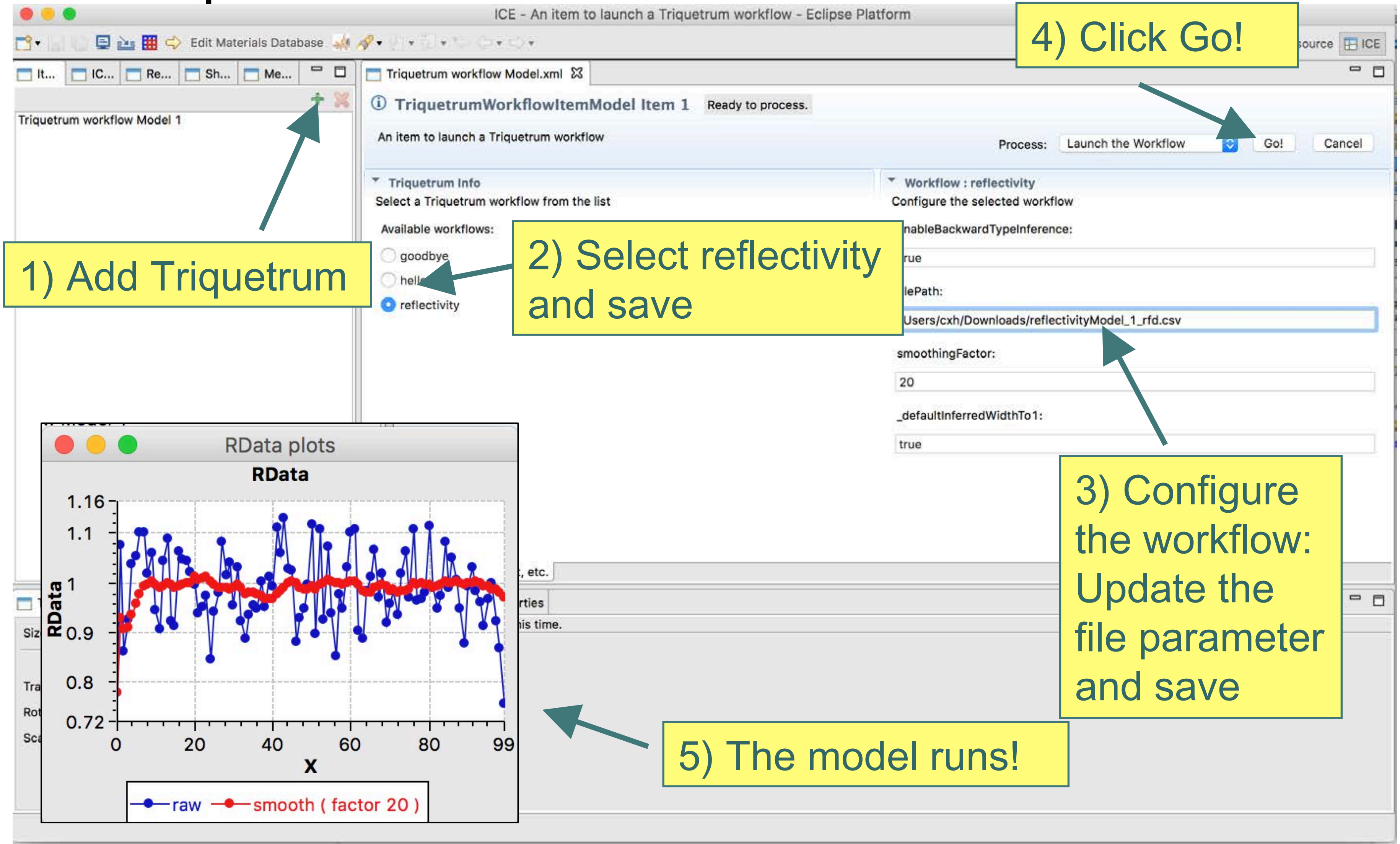
output-Battery_1.1.silo
Path: file:/C:/Users/ /ICEFil
Date: Thu Mar 12 08:22:01 ED

DB: output-Battery_1.1.silo
Cycle: 1

user:
Thu Apr 23 18:24:



ICE: Prototype of running a Triquetrum model



ICE - An item to launch a Triquetrum workflow - Eclipse Platform

Triquetrum workflow Model 1

Triquetrum workflow Model.xml

TriquetrumWorkflowItemModel Item 1 Ready to process.

An item to launch a Triquetrum workflow

Process: Launch the Workflow Go! Cancel

Triquetrum Info

Select a Triquetrum workflow from the list

Available workflows:

○ goodbye
○ hell
● reflectivity

Workflow : reflectivity

Configure the selected workflow

nableBackwardTypeInference:
rue

lePath:
Users/cxh/Downloads/reflectivityModel_1_rfd.csv

smoothingFactor:
20

_defaultInferredWidthTo1:
true

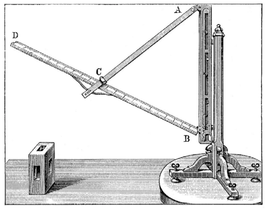
1) Add Triquetrum

2) Select reflectivity and save

3) Configure the workflow: Update the file parameter and save

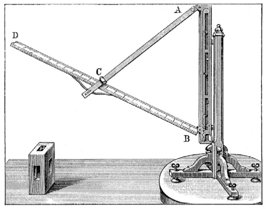
4) Click Go!

5) The model runs!



Triquetrum Outcomes (So Far)

- Eclipse Project Started December 2015
- Source: https://github.com/eclipse/triquetrum
  - 19.4k lines of code (source: https://www.openhub.net/p/triquetrum)
  - “well commented source code” (source: openhub)
  - Uses Ptolemy II jars (6.5Mb), checked in to GitHub using parallel IP Process.
- Wiki: https://wiki.eclipse.org/Triquetrum
  - See the wiki for RCP downloads built nightly!
- Hudson Continuous Integration: https://hudson.eclipse.org/triquetrum/
- Triquetrum-dev mailing list (10 members!)



Conclusion

- Triquetrum uses the Ptolemy II actor-oriented execution engine to provide **run time semantics** for use in workflows.

Questions?

eclipsecon
RESTON VIRGINIA 2016

Evaluate the Sessions

Sign in and vote at **eclipsecon.org**

-1 0 +1