System-Level Design Languages: Orthogonalizing the Issues

Kees Vissers

Tom Henzinger
Jörn W. Janneck
Luciano Lavagno
Edward Lee
Alberto Sangiovanni-Vincentelli
Kees Vissers

The GSRC Semantics Project

Ptolemy Miniconference
Berkeley, CA, March 22-23, 2001

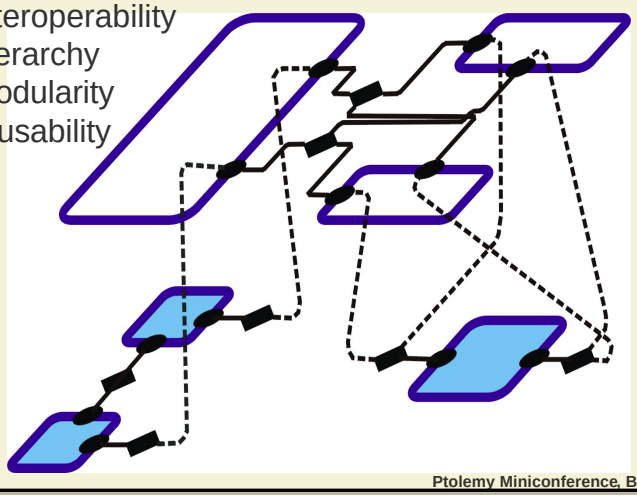
The Problem: System Level Design

- **Modeling**
- **Simulation**
- **Visualization**
- **Synthesis**
- **Verification**
- **Modularization**

The problem we are here to address is *interoperability* and *design productivity*. Not standardization.

Component-Based Design

interoperability
hierarchy
modularity
reusability



Interoperability Levels

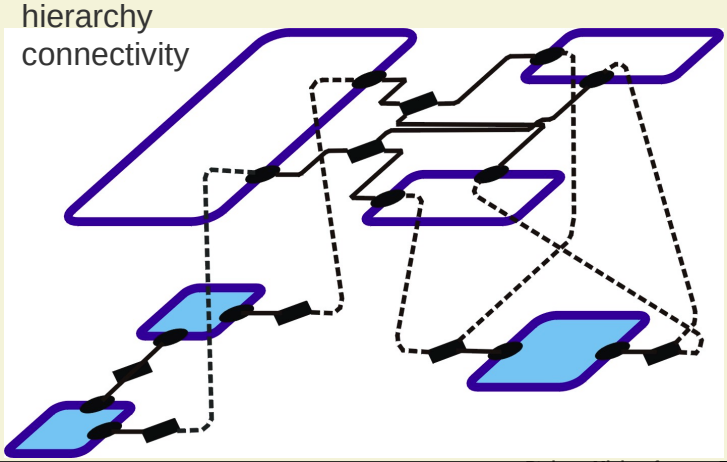
- **Code can be written to translate the data from one tool to be used by another.**
- **Tools can open each other's files and extract useful information (not necessarily *all* useful information).**
- **Tools can interoperate dynamically, exchanging information at run time.**

Principle: Orthogonalize Concerns in System Level Design Languages

- Abstract Syntax
- Concrete Syntax
- Syntactic Transformations
- Type System
- Component Semantics
- Interaction Semantics

Abstract Syntax

hierarchy
connectivity



Concrete Syntaxes

- Persistent file formats
- Close to the abstract syntax
- Make it extensible to capture other aspects
- Enable design data exchange
  - without customization of the tools

Most language discussions focus on concrete syntaxes, which are arguably the least important part of the design

MoML – An XML Concrete Syntax

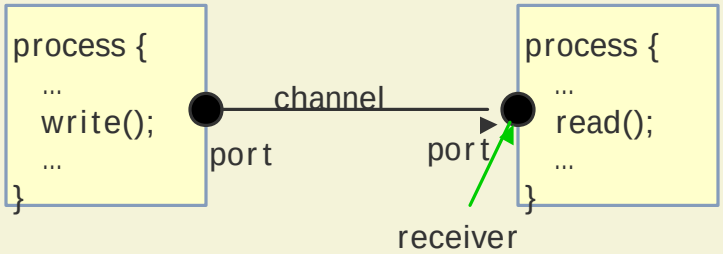
```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE model PUBLIC "…" "http://…">
<model name="top" class="path name">
  <entity name="source" class="path name">
    <port name="output"/>
  </entity>
  <entity name="sink" class="path name">
    <port name="input"/>
  </entity>
  <relation name="r1" class="path name"/>
  <link port="source.output"
 relation="r1"/>
  <link port="sink.input" relation="r1"/>
</model>
```

Component semantics

Entities are:

- **States?**
- **Processes?**
- **Threads?**
- **Differential equations?**
- **Constraints?**
- **Objects?**

Producer Consumer example



- Are actors active? passive? reactive?
- Are communications timed? synchronized? buffered?

Domains

- CSP – concurrent threads with rendezvous
- CT – continuous-time modeling
- DE – discrete-event systems
- DT – discrete time (cycle driven)
- PN – process networks
- SDF – synchronous dataflow
- SR – synchronous/reactive

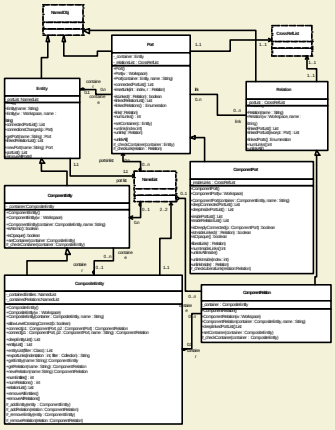
Each of these defines a component ontology and an interaction semantics between components. There are many more possibilities!

Ptolemy Project: Research

Ptolemy II –

- A reference implementation
- Testbed for abstract syntax
- Block diagram MoML editor
- Mutable models
- Extensible type system
- Testbed for system-level type

Ptolemy: structure = definitions



Ptolemy: structure = definitions