MODELING HETEROGENEOUS APPLICATIONS IN PTOLEMY II

Patricia Derler, Edward A. Lee
University of California, Berkeley

International Symposium on System-on-Chip 2011
Tampere, Finland, October 31
MODELING FORMALISMS AND TOOLS

UML [Wikipedia Commons]

AADL [AADL community Wiki]

Fault Trees [Wikipedia]

IEC 61499

Matlab/Simulink [http://www.mathworks.com]

Labview [http://www.ni.com/labview/]

MODELICA
FOCUS IN THE PTOLEMY PROJECT

• Modeling, simulation, and design of concurrent, real-time, embedded systems

• Executable models rather than descriptive models

• Semantics matters (more than syntax)

• Useful semantics imply constraints on designers

• Heterogeneity may be better than generality

Freedom from choice!
COMPONENTS - ACTORS

• Component interactions in object-oriented programming

• An alternative: Actor oriented

The use of the term "actors" for this dates back at least to the 1970s [Hewitt, Agha, Dennis, Kahn, etc.]
ACTOR-ORIENTED MODELING LANGUAGES

- ASCET (time periods, interrupts, priorities, preemption, shared variables)
- Autosar (software components w/ sender/receiver interfaces)
- CORBA event service (distributed push-pull)
- Dataflow languages (many variants over the years)
- LabVIEW (structured dataflow, National Instruments)
- Modelica (continuous time, constraint-based, Linkoping)
- MPI (message passing interface, parallel programming)
- Occam (rendezvous)
- OPNET (discrete events, Opnet Technologies)
- SCADE (synchronous, based on Lustre and Esterel)
- SDL (process networks)
- Simulink (continuous time, The MathWorks)
- SPW (synchronous dataflow, Cadence, CoWare)
- VHDL, Verilog (discrete events, Cadence, Synopsys, ...)
- ...

The semantics of these differ considerably in their approaches to concurrency and time. Some are loose (ambiguous) and some rigorous. Some are strongly actor-oriented, while some retain much of the flavor (and flaws) of threads.
A PTOLEMY MODEL
SEMANTICS - MODEL OF COMPUTATION

- Synchronous Dataflow (SDF)
- Process Network (PN)
- Rendezvous
- Finite State Machine (FSM)
- Discrete Event (DE)
- Continuous

More in my talk on Wednesday!

• Director
• Actor
• Port
• **Director**
• **Actor**
• **Port**
COMMUNICATION MECHANISM

- Director
- Actor
- Port
- Receiver
COMMUNICATION METAMODEL

- Director
- Actor
- Port
- Receiver
ABSTRACT EXECUTABLE SEMANTICS

- **preinit**: E.g., Partial evaluation (esp. higher-order components), set up type constraints, etc. Anything that needs to be done prior to static analysis (type inference, scheduling, …)

- **init**: E.g., Initialize actors, produce initial, outputs, etc., E.g., set the initial state of a state machine.

- **prefire**
- **fire**
- **postfire**

```java
if (prefire()) {
    fire();
    postfire();
}
```

- **wrapup**
Amorphous

Color is an interactive style (KPN, publish - subscribe, procedure call, ...)

Hierarchical

Color is an MoC, with constraints. The meaning is clear at each level of the hierarchy
HETEROGENEITY

• Hierarchical compositions of distinct modeling styles

• Combined to take advantage of the unique capabilities and expressiveness of each style

• The MoC is clear at each level of the hierarchy

• At each level of the hierarchy, the model is understandable
TRAFFIC LIGHT EXAMPLE

This is a diagram of a traffic light system. The diagram includes various components such as CarLight, SR Director, WirelessDirector, Clock, PoissonClock, RadioChannel, and TimedDelay. The system is designed to manage traffic light signals for cars and pedestrians, with states and transitions defined by conditions and actions.

Implementations:

- Normal state:
  - guard: Ok_isPresent
  - output: Ok

- Error state:
  - guard: Error_isPresent

The diagram shows the flow of transitions between different states, including guard conditions and output actions. The system is intended to ensure smooth traffic flow by implementing rules for changing light states.
BACKGROUND ON HIERARCHICAL MULTIMODELING

- Statecharts [Harel 87]
- Ptolemy Classic [Buck, Ha, Lee, Messerschmitt 94]
- SyncCharts [André 96]
- *Charts [Girault, Lee, Lee 99]
- Colif [Cesario, Nicolescu, Guathier, Lyonnard, Jerraya 01]
- Metropolis [Goessler, Sangiovanni-Vincentelli 02]
- Ptolemy II [Eker, et. al. 03]
- Safe State Machine (SSM) [André 03]
- SCADE [Berry 03]
- ForSyDe [Jantsch, Sander 05]
- ModHelX [Hardebolle, Boulanger 07]
CYBER-PHYSICAL SYSTEMS

Multiple computers, comprising of sensors and actuators, connected on a network that act and react on events to meet timing constraints.
CHALLENGES IN MODELING CPS

- Modeling distributed computations
- Modeling time
  - Execution time
  - Reaction time
  - Timing requirements
  - Time synchronization
  - Time on distributed platforms
- Modeling networks
TIME IN MODELING

• Timing is not part of the software semantics

• Correct execution of a program in conventional programming languages such as C has nothing to do with how long it takes to do something.

• Issues:
  Programming languages, Virtual memory, Caches, Dynamic dispatch, Speculative execution, Power management (voltage scaling), Memory management (garbage collection), Just-in-time (JIT) compilation, Multitasking (threads and processes), Component technologies (OO design), Networking (TCP), …
EXECUTION TIME ANALYSIS IS NOT EVERYTHING

• Paths through the software (undecidable)

• Detailed model of the microarchitecture and memory system

• Complete knowledge of the execution context

• Many constraints on preemption/concurrency

• Results are only valid for the exact same hardware and software

• Fundamentally, the ISA of the processor has failed to provide an adequate abstraction

THE TIME CHALLENGE

Distributed platforms have different notions of time

Platform clocks drift

Platform clocks drift at varying rates

Multiple clocks on a single platform: SW/HW clocks
MODELING DISTRIBUTED SYSTEMS

Director mediates between actors
Difficult to maintain different notions of time

One hierarchy level is not enough
MODELING DISTRIBUTED SYSTEMS

Hierarchies:
- Opaque composite actors
- Embedded directors maintain time
MODELING DISTRIBUTED SYSTEMS

Top level: Oracle time

Platform time is defined with respect to oracle time

- Platform1 time = \( f_1(t_0) \)
- Platform3 time = \( f_3(t_0) \)
- Platform2 time = \( f_2(t_0) \)

\[ f_x(\text{Oracle time}) \]
CLOCK SYNCHRONIZATION

Master
Slave
adjust
clock
rate:

\[ f_{\text{Oracle}}(\text{Oracle time}) \]

adjust
clock
value:

\[ f_{\text{Oracle}}(\text{Oracle time}) \]

Oracle time

Oracle time

Patricia Derler - 2011
Distributed platforms communicate via networks. Networks have latencies, e.g., CAN Bus, TTEthernet.
Physical connections vs. Logical connections

Logical connections are lost
MODELING NETWORKS

Aspect-oriented modeling

Quantity managers [Balarin03] and schedulers to simulate network latency

MODELING EXECUTION TIME

The diagram illustrates the flow of execution time between different platforms and components. Each component is labeled with its execution time, indicating the computational load or processing time associated with it. The physical interface connects the platforms, facilitating the exchange of data or signals. The diagram also includes elements such as sensors (Sensor1, Sensor2), computations (Computation1, Computation2, Computation3, Computation4), and actuators (Actuator1), all interconnected to model the execution time flow in a comprehensive system.
MODELING EXECUTION TIME

How is execution time computed?

Which time line to use for specifying execution time?

Aspect-oriented programming: annotate component with execution time and “piggyback” simulation
DISTRIBUTED DISCRETE-EVENT MODELS

DE as a application specification language, semantic basis for obtaining determinism in distributed real-time systems.
DISTRIBUTED DISCRETE-EVENT MODELS

• Platform1 time
• Logical time

• Platform2 time
• Logical time

• Platform3 time
• Logical time

Logical time describe the execution semantics

New time line: logical time

Oracle time
Platform time
Logical time
PTIDES: AN APPLICATION

Programming temporally integrated distributed event systems [Zhao07]

- Discrete event model for execution
- Relates logical time to platform time at sensors, actuators and network interfaces
- Requires bounded error between platform clocks: Relies on clock synchronization
- Causally related events are processed in time-stamped order

A PTIDES MODEL

Platform time progresses

Platform time

Oracle time

Logical time

Platform time

Physical interface

Network fabric

Local Event Source

Sensor delays, dynamic network latencies, clock synchronization error

response time ≤ logical time delay

send time stamp and value and over network

increase time stamp

time stamp ≤ platform time

time stamp ≥ platform time

Sensor 1
model time delay d4

Sensor 2
model time delay d5

merge

Actuator 1
physical interface

physical plant

increase time stamp

time stamp ≤ platform time

time stamp ≥ platform time

platform time progresses

Oracle time

Logical time

Platform time

Physical interface

Network fabric

Local Event Source

Sensor delays, dynamic network latencies, clock synchronization error

response time ≤ logical time delay

send time stamp and value and over network

increase time stamp

time stamp ≤ platform time

time stamp ≥ platform time

platform time progresses

Oracle time

Logical time

Platform time

Physical interface

Network fabric

Local Event Source

Sensor delays, dynamic network latencies, clock synchronization error

response time ≤ logical time delay

send time stamp and value and over network

increase time stamp

time stamp ≤ platform time

time stamp ≥ platform time

platform time progresses

Oracle time

Logical time

Platform time

Physical interface

Network fabric

Local Event Source

Sensor delays, dynamic network latencies, clock synchronization error

response time ≤ logical time delay

send time stamp and value and over network

increase time stamp

time stamp ≤ platform time

time stamp ≥ platform time

platform time progresses

Oracle time

Logical time

Platform time

Physical interface

Network fabric

Local Event Source

Sensor delays, dynamic network latencies, clock synchronization error

response time ≤ logical time delay

send time stamp and value and over network

increase time stamp

time stamp ≤ platform time

time stamp ≥ platform time

platform time progresses

Oracle time

Logical time

Platform time

Physical interface

Network fabric

Local Event Source

Sensor delays, dynamic network latencies, clock synchronization error

response time ≤ logical time delay

send time stamp and value and over network

increase time stamp

time stamp ≤ platform time
SUMMARY

Ptolemy

• actor-oriented framework

• models of computations via directors

• abstract semantics

• heterogeneity via hierarchies

Modeling cyber-physical systems

• modeling time in distributed systems

• modeling networks

• discrete-event models

• PTIDES