Hierarchical, Heterogeneous Modeling and Design

A model of computation governs the interaction of components at each level of the hierarchy. A submodel exposes a domain-polymorphic interface that governs the inter-domain semantics.
Ptolemy II

- Java based, network integrated
- Many domains implemented
- Multi-domain modeling
- XML syntax for persistent data
- Block-diagram GUI
- Extensible type system
- Code generator on the way

http://ptolemy.eecs.berkeley.edu

Domains Status

Domains we understand well:
- Dataflow
- Process networks
- CSP
- Discrete events
- Continuous time
- Synchronous reactive
- Finite state machines

Our focus is particularly on how these domains support real-time QOS

Domains we are working on:
- Publish & subscribe
- Time triggered
Concept Demonstration

- Networked sensors and actuators
- Multiple, networked controllers, controllees
- Hierarchical, heterogeneous design
- Domain polymorphic components
- Discovery
- Mutable systems

Experimental Setup

1451.2

Ethernet Hub

Telemonitor     Agilent
Tilt sensor     NCAP
Networked Smart Sensors

Abstraction of the Sensor as a Software Component
Smart Sensor + Ptolemy II

Tilt sensor connected to a plotter.

Issues Raised

- Concurrency management with I/O
  - Separate thread handles communication
  - Rendezvous with computational thread
  - How to maintain time consistency?
  - How to ensure no deadlock?
Planned - Discovery

1. Hub
2. Ethernet
3. Register a Service
4. Agilent
5. NCAP
6. 1451.2
7. Telemonitor
8. Tilt sensor
9. JINI

Planned - Discovery

1. Hub
2. Ethernet
3. Discover a Service
4. Agilent
5. NCAP
6. 1451.2
7. Telemonitor
8. Tilt sensor
9. JINI
This would provide a vendor-neutral software component.
Actuator Setup

Proxy
- LegoSensor1
- LegoDriver1

serial port
- IR tower
- Lego Mindstorm

Linking the Tilt Sensor and Actuators

Diagram showing connections between
- TiltSensor
- ScaleX
- SumL
- TimeScope18
- LegoDriver14
**Mutations – Dynamic Structural Changes to the Model**

- Thread-safe Ptolemy II kernel
  - Mutual exclusion protocol in the Workspace object.

- Domains control when mutations are committed.
  - Mutations are queued with the Manager object.
  - Manager executes mutations between *iterations*.
  - Meaning of “iteration” is domain-dependent.

- In this example:
  - The event thread in the UI queues mutation requests
  - The executing model commits the mutations at safe points.

**Publish and Subscribe**

- Use Jini to discover the publish/subscribe fabric.
  - Our current realization returns a JavaSpaces interface.
  - Future realization will use OCP

- Real time
  - Prioritized delivery, handling
  - QOS is not part of JavaSpaces.
Clock Publisher/Subscriber

Distributed Lego Controller

Tilt sensor data published, controller subscribes.

left = 4 * xTilt - 2 * yTilt
right = 4 * xTilt + 2 * yTilt
Other Examples We Have Implemented

- Other Lego models:
  - Modal controller for navigation
  - Feedback of sensor data
- Hybrid systems:
  - Car tracking example
  - Helicopter multi-modal controller
- Pioneer robot control
  - Multi-agent coordination
  - Jini discovery of robots
  - Publish-and-subscribe task distribution

Styles of Publish and Subscribe Interactions

- time stamped events?
- globally time stamped?
- reliable delivery?
- ordered delivery?
- signal coordination?
- synchronous delivery?
- blending of multiple publishers?
- dynamic redirection/resourcing?
- persistence?
- history?
A Key Idea

- We need a variety of interaction mechanisms.
- In the prototype,
  - Jini delivers an interaction mechanism service by delivering code that realizes that interaction mechanism.
- A "meta OCP" could similarly deliver any of several interaction mechanisms.

Example 1

- Component says:
  - "I need a reliable stream-based delivery mechanism to get sampled data from here to there."
- Meta-OCP says:
  - "OK, here’s some code for you and the recipient of your data."
- Delivered code uses TCP/IP and sockets, bypassing any central infrastructure.
  - E.g., Transporting audio data.
Example 2

- Component says:
  - "I need a shared data repository visible to a number of components."
- Meta-OCP says:
  - "OK, here's some code for you and the recipient of your data."
- Delivered code interacts with a Linda-style tuple space.
  - E.g., reading the current temperature from a sensor.

Example 3

- Component says:
  - "I need to send time-stamped data that must be delivered and dealt with within 3 msec."
- Meta-OCP says:
  - "OK, here's some code for you and the recipient of your data."
- Delivered code interacts with TAO.
  - E.g., deliver motion control data.
Next Steps

- OCP integration
- Define publish and subscribe semantics
- Discovery of sensor/actuator services
- Abstraction of sensor/actuator services
- Real-time QOS
- Time-driven domain (Masaccio)
- Multi-robot coordination
- Improved UI (particularly to help debugging)

The Demo Builders...

- Chamberlain Fong
- Christopher Hylands
- Jie Liu
- Xiaojun Liu
- Steve Neuendorffer
- Sonia Sachs
- Win Williams