Computing for Embedded Systems

Edward A. Lee
UC Berkeley

computers without actuators and sensors are destined to look like this.

State of the Art Lecture
IEEE Instrumentation and Measurement Technology Conference
Budapest, Hungary, May 21-23, 2001

What is an Embedded System?

- One or more computers
  - but not first-and-foremost a computer
- Interaction with physical processes
  - sensors, actuators, processes
- Reactive
  - operating at the speed of the environment
- Heterogeneous
  - hardware/software, mixed architectures
- Networked
  - adaptive software, shared data, resource discovery
Why is Embedded SW an Issue?

- Embedded systems becoming networked
  - more complex, more vulnerable
  - can no longer use static point designs
- Focus on non-functional properties is new for SW
  - real-time, fault recovery, power, security, robustness
- Neglected area
  - computer science has largely ignored it
  - best-of-class methods don't help much

E.g. Object-Oriented Design

- Call/return imperative semantics
- Concurrency is via ad-hoc calling conventions
  - band-aids: futures, proxies, monitors
- Poorly models the environment
  - which does not have call/return semantics
- Nothing at all to say about time

Object modeling emphasizes inheritance and procedural interfaces.

We need to emphasize concurrency and communication abstractions.
E.g. Real-Time Corba

Component specification includes
- worst case execution time
- typical execution time
- cached execution time
- priority
- frequency
- importance

This is an elaborate prayer…

Alternative View of SW Architecture:
*Actors with Ports and Attributes*

Model of Computation:
- Messaging schema
- Flow of control
- Concurrency

Examples:
- Time triggered
- Process networks
- Discrete-event systems
- Dataflow systems
- Publish & subscribe

*Key idea:* The model of computation is part of the framework within which components are embedded rather than part of the components themselves.
Examples of Actors+Ports Software Architectures

- Simulink (The MathWorks)
- Labview (National Instruments)
- OCP, open control platform (Boeing)
- SPW, signal processing worksystem (Cadence)
- System studio (Synopsys)
- ROOM, real-time object-oriented modeling (Rational)
- Port-based objects (U of Maryland)
- I/O automata (MIT)
- VHDL, Verilog, SystemC (Various)
- Polis & Metropolis (UC Berkeley)
- Ptolemy & Ptolemy II (UC Berkeley)
- …

What a Program Might Look Like
Ptolemy II – An Open Source Software Laboratory

Ptolemy II –

Emphasis is on building a framework supporting experimentation with models of computation and their interactions.

http://ptolemy.eecs.berkeley.edu

Example: Controlling an Inverted Pendulum

The Furuta pendulum has a motor controlling the angle of an arm, from which a free-swinging pendulum hangs. The objective is to swing the pendulum up and then balance it.
Execution

An execution of the model displays various signals and at the bottom produces a 3-D animation of the physical system.

Model by Johan Eker

Top-Level Model

The top-level is a continuous-time model that specifies the dynamics of the physical system as a set of nonlinear ordinary differential equations, and encapsulates a closed loop controller.

Framework by Jie Liu
Higher-Order Components

Complex submodels are generated automatically from equations describing the physical dynamics.

A Modal Controller

The controller itself is modal, with three modes of operation, where a different control law is specified for each mode.
The Discrete Controllers

Three discrete submodels (dataflow models) specify control laws for each of three modes of operation.

Framework by Steve Neuendorffer

Edward A. Lee, Berkeley, 15

The Graphical Animator

A 3-D graphical animation is constructed modularly as a parametric scene graph that is driven by the physical model.

Framework by Chamberlain Fong

Edward A. Lee, Berkeley, 16
The Key Idea

- Components are actors with ports
- Interaction is governed by a model of computation
  - flow of control
  - messaging protocols

So what is a model of computation?

- It is the “laws of physics” governing the interaction between components
- It is the modeling paradigm

Model of Computation

- What is a component? (ontology)
    Constraints? Objects (data + methods)?
- What knowledge do components share? (epistemology)
  - Time? Name spaces? Signals? State?
- How do components communicate? (protocols)
  - Rendezvous? Message passing? Continuous-time signals?
    Streams? Method calls? Events in time?
- What do components communicate? (lexicon)
  - Objects? Transfer of control? Data structures? ASCII text?
Domains – Ptolemy II Realizations of Models of Computation

- CSP – concurrent threads with rendezvous
- CT – continuous-time modeling
- DE – discrete-event systems
- DDE – distributed discrete-event systems
- DT – discrete time (cycle driven)
- FSM – finite state machines
- Giotto – time driven cyclic models
- GR – graphics
- PN – process networks
- SDF – synchronous dataflow
- xDF – other dataflow

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

Heterogeneity – Hierarchical Mixtures of Models of Computation

- Modal Models
  - FSM + anything
- Hybrid systems
  - FSM + CT
- Mixed-signal systems
  - DE + CT
  - DT + CT
- Complex systems
  - Resource management
  - Signal processing
  - Real time
Hierarchical, Compositional Models

Schedulers (Directors) are nested hierarchically, each interacting with components through the Executable interface. Directors themselves implement this same interface, so the model is compositional.

Key Advantages

- Domains are specialized
  - lean
  - targeted
  - optimizable
  - understandable

- Domains are mixable (hierarchically)
  - structured
  - disciplined interaction
  - understandable interaction
Ptolemy II

- Open source framework
- Block diagram user interface
- XML persistent file format
- Java based
- Network integrated
- Fully extensible
- Experiment with models of computation

http://ptolemy.eecs.berkeley.edu

Example Ptolemy II Experiment:
Connecting to Smart Sensors

Simple demonstration project for DARPA software-enabled control (SEC) project.
Networked Smart Sensors

HTTP queries
Ethernet
Agilent
NCAP
HTTP interface
1451.2
Telemonitor
Tilt sensor

Abstraction of the Sensor as a Software Component

HTTP queries
Ethernet
Agilent
NCAP
1451.2
Telemonitor
Tilt sensor
Discovery – Realizing More Specialized Communication with the Sensors

Hub
Ethernet
Register a Service

Agilent NCAP
1451.2
Telemonitor Tilt sensor

JINI

Ethernet
Discover a Service

Agilent NCAP
1451.2
Telemonitor Tilt sensor
Discovery – Realizing More Specialized Communication with the Sensors

This would provide a vendor-neutral software component.
Smart Sensor + Ptolemy II

Tilt sensor connected to a plotter.

Actuator Setup

Proxy

serial port

Lego Mindstorm

IR tower
Distributed Lego Controller

Tilt sensor data published, controller subscribes.

\[
\text{left} = 4 \times x_{\text{Tilt}} - 2 \times y_{\text{Tilt}} \\
\text{right} = 4 \times x_{\text{Tilt}} + 2 \times y_{\text{Tilt}}
\]

Issues Raised

Concurrency management with I/O

- Separate threads handling I/O communication?
- Rendezvous with computational thread?
- Publish & subscribe buffering strategy?
- How to maintain time consistency?
- How to ensure no deadlock?
- Is HTTP a good way to communicate with sensors?
Conclusions

- Actors with ports are better than objects with methods for embedded system design.
- Models of computation matter a great deal, and specialization can help.
- Ptolemy II provides an excellent open architecture for experimentation.

http://ptolemy.eecs.berkeley.edu