Architecture
Modeling and Analysis for
Embedded Systems

Oleg Sokolsky

CIS700
Fall 2005
Overview

• Background
  - Architecture description languages
  - Embedded and real-time systems
• AADL: ADL for embedded systems
• Analysis of embedded systems with AADL
Architecture vs. behavior

- How it is constructed vs. what does it do?

- Traditionally, behavior was considered more important
Software and hardware architectures

- **Software architecture:**
  - fundamental organization of a system, embodied in its components,
  - their relationships to each other and the environment, and
  - principles governing its design and evolution

- **Hardware architecture:**
  - Interfaces for attaching devices
  - Instruction set architecture
Components, ports, and connections

- Components are boxes with interfaces
- Component interfaces described by ports:
  - Control
  - Data
  - Resources
- Connections establish control and data flows
- The nature of components may be abstracted
  - Hardware or software, or hybrid
Software ADLs

- **Wright**
  - Connector-based: CSP connector semantics
  - Configuration and evolution support
- **ACME**
  - Interchange format: weak semantics or constraint enforcement, little analysis
- **MetaH**
  - Strong component semantics
  - Specification of non-functional properties
Overview

• Background
  - Architecture description languages
  - Embedded and real-time systems
• AADL: ADL for real-time systems
• Analysis of embedded systems with AADL
Embedded system architectures

- Both hardware and software aspects are important
  - Increasingly distributed and heterogeneous
- Tight resource and timing constraints
- Multimodal behaviors
  - Some components are active only in certain circumstances
    - E.g., fault recovery
- Analysis is important
Real-time systems

• The science of system development under resource and timing constraints
  - System is partitioned into a set of communicating tasks
  - Tasks communicate with sensors, other tasks, and actuators

  • Impose precedence constraints

  s
  Task 1 — Task 3 — Task 4
  s
  s
  s
Task execution

- Tasks are invoked periodically or by events
  - Must complete by a deadline
- Tasks are mapped to processors
- Tasks compete for shared resources
  - Resource contention can violate timing constraints
Real-time scheduling

• Processor scheduling
  - Task execution is preemptable
  - Tasks assigned to the same processor are selected according to priorities
  - Priorities are assigned to satisfy deadlines
    • Static or dynamic
• Resource scheduling
  - Mutual exclusion
    • Often non-preemptable
  - Correlated with processor scheduling
Overview

• Background
  - Architecture description languages
  - Embedded and real-time systems
• AADL: ADL for real-time systems
• Analysis of embedded systems with AADL
AADL highlights

- Architecture Analysis and Design Language
- Oriented towards modeling embedded and real-time systems
  - Hardware and software components
  - Control, data, and access connections
- Formal execution semantics in terms of hybrid automata
- SAE standard AS-5506
AADL components

Software components
- Thread
- Thread group
- Data
- Subprogram
- Process

Platform components
- Processor
- Memory
- Bus
- Device
- System

System components
Component interfaces (types)

- **Features**
  - Points for external connections
    - E.g., data ports

- **Flows**
  - End-to-end internal connections

- **Properties**
  - Attributes useful for analysis
Component implementations

- Internal structure of the component
  - Subcomponents are type references
  - Connections conform with flows in the type
  - External features conform with the type
  - Internal features conform with subcomponent types
Features and connections

- **Communication**
  - Ports and port groups
  - Port connections
- **Resource access**
  - Required and provided access
  - Access connections
- **Control**
  - Subprogram features
  - Parameter connections
Ports and port groups

- Ports are typed
  - Data component types
- Ports are directional
  - Input, output, or bi-directional
- Synchronous or asynchronous communication
  - Event, data, or event data ports
    - Input event and event data ports have queues
    - Input data ports have status flags for new data
Data components

- Data component types represent data types
- Data component type can have subprogram features that represent access methods
- Data component implementations can have data subcomponents that represent internal data of an object
- Data component types can also be used as types of data ports and connections
Thread components

- Thread represents a sequential flow of control
  - Can have only data as subcomponents
- Threads are executable components
  - Execution goes through a number of states
    - Active or inactive
  - Behaviors are specified by hybrid automata
Thread states

- **Uninitialized Thread**
- **Initialize**
- **InitializeComplete**
- **Active**
  - **Member of current mode**
  - **Active In New Mode**
  - **Active In Init Mode**
  - **Inactive**
    - **Not member of current mode**
  - **Deactivate**
  - **DeactivateComplete**

- **Active Dispatch**
  - **Suspended**
  - **Computation**
  - **Fault**
  - **Recovered**
  - **Repaired**

- **Terminate**
  - **Finalize**
  - **FinalizeComplete**
  - **Terminated Thread**
Thread Hybrid Automata
Thread properties

- Dispatch protocol
  - periodic, aperiodic, sporadic, or background
- Period
  - For periodic and sporadic threads
- Execution time range and deadline
  - for all execution states separately (initialize, compute, activate, etc.)
Thread dispatch

- Periodic threads are dispatched periodically
  - Event arrivals are queued
- Non-periodic threads are dispatched by incoming events
- Pre-declared ports
  - Event in port Dispatch
    - If connected, all other events are queued
  - Event out port Complete
    - Can implement precedence
Subprograms

- Data subprograms are features of data components
- Server subprograms are features of threads
- Represent entry points in executable code
- No static data
  - External data access through parameter and access connections
- Data subprograms are called within a process
- Server subprograms are called remotely
Other software components

- **Process**
  - Represents virtual address space
  - Provides memory protection
- **Thread group**
  - Organization of threads within a process
  - Can be recursive
- **Subprogram**
  - Represents entry points in executable code
  - Calls can be local or remote
Platform components

• Processor
  - Abstraction of scheduling and execution
  - May contain memory subcomponents
  - Scheduling protocol, context switch times
• Memory
  - Size, memory protocol, access times
• Bus
  - Latency, bandwidth, message size
Port connections revisited

- Event connections support n-n connectivity
- Data connection support 1-n connectivity
  - One incoming, multiple outgoing
Port connections revisited

- Semantic port connection
  - Ultimate source to ultimate destination
    - Thread, processor, or device
- Type checking of connections
  - Directions and types must match
Immediate and delayed connections

- Data connections between periodic threads
Component bindings

- Software components are bound to platform components
- Binding mechanism:
  - Properties specify allowed and actual bindings
  - Allows for exploration of design alternatives
Putting it all together: systems

- Hierarchical collection of components
Putting it all together: systems

- A different perspective on the same system
Modes

- **Mode**: Subset of components, connections, etc.
- **Modes** represent alternative configurations

Mode Diagram:

- **Nominal**
- **Degraded**
- Fault propagation and recovery pathways
Mode Switch

• Mode switch can be the ultimate source of an event connection
• Switch effects:
  – Activate and deactivate threads
  – Reroute connections
• Switch can also be local to a thread
  – Change thread parameters
• Switch takes time:
  – Threads need to be in a legal state
  – Activation and deactivation take time
Overview

• Background
  - Architecture description languages
  - Embedded and real-time systems
• AADL: ADL for real-time systems
• Analysis of embedded systems with AADL
Static architectural analysis

- **Type checking**
  - Types of connected ports
  - Allowed bindings
  - Do all connections have ultimate sources and destinations

- **Constraint checking**
  - Does the size of a memory component exceed the sizes of data components bound to it?
Dynamic architectural analysis

- Relies on thread semantics
- Processor scheduling

```
T1: Period => 20ms
    Compute_Deadline => 20ms
    Compute_Execution_Time => [200us, 500us]

T2: Period => 35ms
    Compute_Deadline => 35ms
    Compute_Execution_Time => [1ms, 5ms]

T3: Period => 100ms
    Compute_Deadline => 100ms
    Compute_Execution_Time => [2ms, 7ms]
```

Scheduling_protocol => RM  

RMA tool
Dynamic architectural analysis

• Advanced processor scheduling

```
Scheduling_protocol => Slack_Server
```

State space exploration
Summary

• Architectural modeling and analysis
  - aids in design space exploration
  - records design choices
  - enforces architectural constraints

• AADL
  - Targets embedded systems
  - Builds on well-established theory of RTS
  - As a standard, encourages tool development