Architectural Semantics of AADL using Event-B

Meenakshi D'Souza
(Joint work with Manoranjan Satpathy and S. Ramesh)

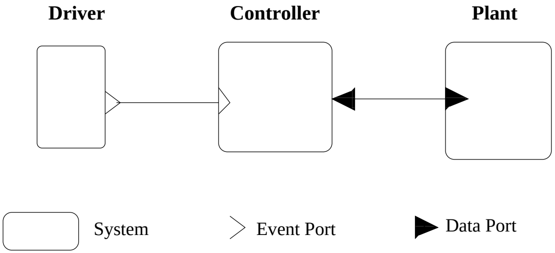
IIIT-Bangalore.

October 16th '14

AADL

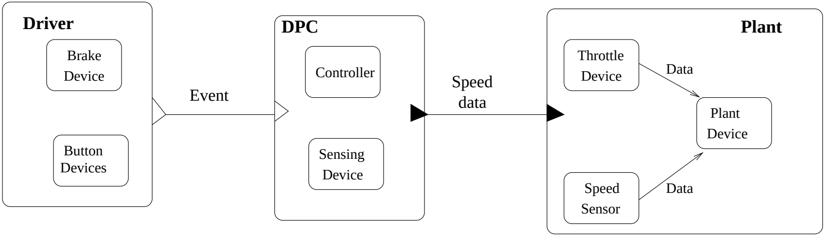
- AADL (Architecture Analysis and Design Language) is a standard language to specify architecture of embedded software.
- AADL is a component-style framework, components describe both software and hardware/platform entities.
- AADL supports modular definition of architectures through repeated refinements; pre-defined refinement constructs like sub-components, component arrays etc. are available.

AADL: Cruise Control System Model

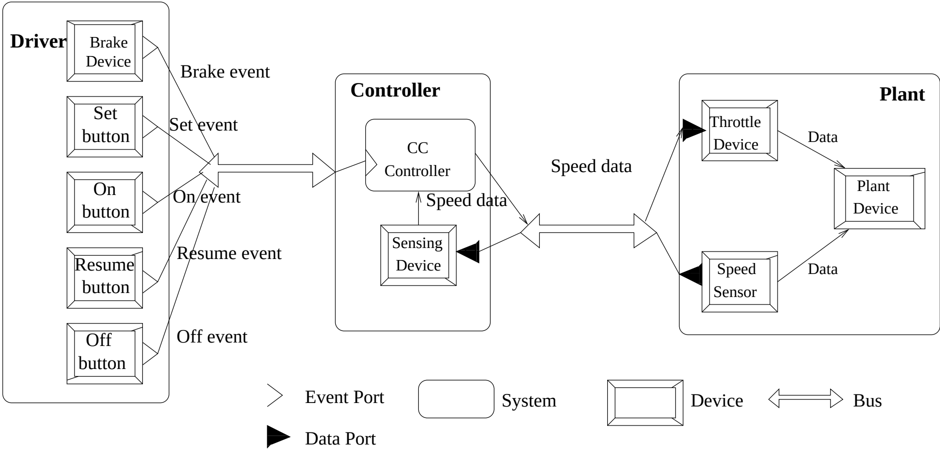


Abstract AADL model: Cruise Control System

AADL: Refined Cruise Control System Model



AADL: Refined Cruise Control System Model



AADL: In a nutshell

- Components:
  - Application software: Thread, process, data, subprogram, system
  - Platform/hardware: processor, memory, bus, device, virtual processor, virtual bus
- Communication across components: Data and event ports, synchronous call/return, shared access, end-to-end flows
- Modes and mode transitions
- Incremental modeling of large-scale systems: package, sub-components, arrays of components and connections
- Eclipse-based modeling framework, annexes for fault modeling, interactions and behavior modeling etc.

Properties in AADL

- AADL allows specification of many standard architectural properties through pre-defined property templates.
- Examples:
  - Sporadic server: `Allowed_Dispatch_Protocol => (sporadic);`
    `Scheduling_Protocol => (FIFO);`
    `Period => 50ms;`
    `Execution_Time => 10ms;`
    `Dispatch_Protocol => periodic;`
  - Processor speed: `Compute_Execution_Time => 700us..750us in binding PowerPC.Mhz350;`

AADL: Modeling large-scale systems

- AADL allows modular specification of large-scale architectures.
- Several syntactic constructs are available for incremental refinements.
  - Components extensions: types, implementation, sub-components, component arrays
  - Features, feature arrays, flows
  - Add or override properties

Refinements and properties in AADL

- AADL being a vast language, lacks exhaustive semantics.
- Refinements in AADL:
  - Type matching/extension, feature matching/extension etc. are all syntactic constructs.
  - No consistency checks for refinements.
- Inconsistency in properties:
  - End-to-end latency specified might be unachievable through the various thread dispatch properties and flow specifications.
  - Data specified for a particular port type while refining might be inconsistent.
  - Bandwidth provided by virtual buses in a component might not match the actual expected bandwidth specified as a property of the component.

Event-B

- Event-B is a formal modeling notation based on first order logic and set theory.
- Machines, basic unit of modeling in Event-B has a set of constants, variables along with their invariants declared.
- Dynamic behavior of machines is specified using events which have guards that dictate when they are enabled and actions that get executed.
- Refinements are formalized in Event-B and incremental models can be obtained using consistent refinements that can be formally proved.

Event-B: Cruise Control System Model

```
MACHINE  CCN0
SEES  context0
VARIABLES dbutton
cc_th
wbutton
INVARIANTS
    inv1  :  dbutton ∈ BUTTONS → BOOL
    inv2  :  wbutton ∈ BUTTONS → BOOL
    inv3  :  cc_th ∈ 0 ‥ MAXTH
EVENTS
    INITIALISATION  ≙.......
    press_swOn  ≙
    BEGIN
      act1  :  dbutton(switchOn) := TRUE
    END
    D_TO_W_swon  ≙
      ANY son
      WHERE
      grd1  :  son ∈ BOOL
      grd10  :  son = dbutton(switchOn)
      THEN
      act1  :  wbutton(switchOn) := son
      act10  :  dbutton(switchOn) := FALSE
      END
```

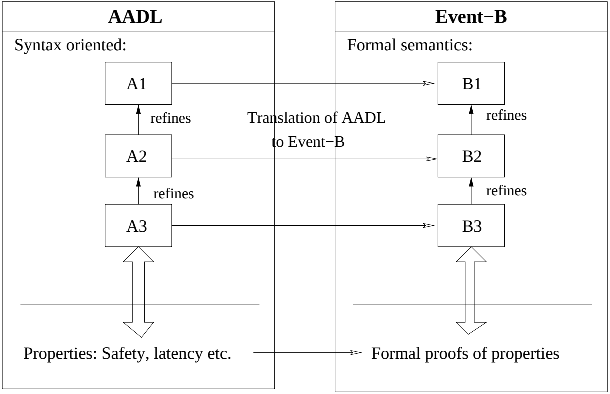
Event-B: Tool support

- Rodin tool platform provides tool support for modeling using Event-B.
- Consistency and refinement checking can be done through auto-generated set of proof obligations in Rodin.
- Proof obligations are discharged automatically or by interactive theorem proving.
- Thus, refinement relationship between an abstract model and its refined concrete model can be proved to be consistent in Event-B.

AADL semantics using Event-B

- We provide a semantics of AADL using Event-B.
- Machines in Event-B correspond to components in AADL.
- Events of Event-B represent all communication in AADL, including events and data exchanged through ports, shared variables, thread execution, mode transitions etc.
- Properties associated with components in AADL correspond to invariants and guards in Event-B.

AADL refinements using Event-B



AADL refinements using model decomposition in Event-B

- Rodin toolset has a decomposition plug-in which can perform model decomposition.
- Individual machines in a given Event-B model can be decomposed into (sub)-machines and events of the parent machine can be distributed amongst its decomposed machines.
- Synchronous composition of decomposed machines defines the behavior of the global model.
- Model decomposition in Event-B is again based on a sound theory of decomposition and refinements obtained through model decomposition can be checked for consistency within Event-B.

Cruise control system: Proving of architectural properties

- Event-B model corresponding to AADL model of cruise control system was arrived at through manual translation.
- AADL model included
  - System, devices, processes, threads, modes, buses, timing properties for thread dispatch, transmission bounds on buses
- Global variables were used to model time and timing properties of threads in Event-B.
- End-to-end latency requirement of AADL was specified using a set of invariants of the overall model.
- Correct latency requirements were discharged through 123 proof obligations, few of which were proved through interactive theorem proving.

Work in progress

- Our work so far has demonstrated that it is possible to provide semantics of architectural components in AADL using Event-B.
- Such a semantics will help designers to formally prove requirements relating to architecture (semi-)automatically.
- We are currently implementing a translator that will automatically generate (and decompose) incremental Event-B models from incremental AADL models.
- Such a translation framework, once implemented, will generate Event-B models automatically from AADL models, with minimal interactive proof obligations.