Model and Verify the architecture of a Satellite Central Flight Software

Dave Thomas (dave.thomas@astrium.eads.net)
The ArchiDyn study

- Goal: to use AADL (with the behavior annex provided by IRIT) to describe the dynamics of satellite central flight software and analyze the contribution of modeling techniques in its validation.

- Modeling part: three levels of abstraction corresponding to three AADL models
  - L0: functional architecture (specification)
  - L1: logical architecture (design)
  - L2: concrete architecture (detailed design / implementation).

- Process & Methodology exploration part: a model-based approach for the construction and the validation of software architectures, allowing to check as soon as possible and gradually the implementation of satellite central software.
AADL modeling: Views and Concerns

**ARCHITECTURE: Views**

- **L0** Dataflows View (URD)
- **L1** Dynamic View (ADD)
- **L2** Structural View (SRD)

**ANALYSIS: Concerns**

- Logical behavior
- Temporal behavior
- Resource analysis
# Levels of abstractions

<table>
<thead>
<tr>
<th>Level</th>
<th>Contents</th>
<th>Inputs</th>
<th>Goals</th>
</tr>
</thead>
<tbody>
<tr>
<td>L0</td>
<td>Functional architecture (structure &amp; high level behavior)</td>
<td>URD/SRD</td>
<td>Specification verification, logical properties (invariants, safety, etc.)</td>
</tr>
<tr>
<td>L1</td>
<td>Logical architecture (platform independent)</td>
<td>ADD &amp; Budget Report</td>
<td>RT analysis: deadlines, scheduling, worst-case analysis</td>
</tr>
<tr>
<td>L2</td>
<td>Concrete architecture (platform dependent)</td>
<td>ADD &amp; CDD</td>
<td>Mutual exclusion, latency, data dependent analysis, code generation</td>
</tr>
</tbody>
</table>
L0 Level: Functional Architecture (specification)

- Modes (L0.1)
  - Communicating automata
  - Mode switching procedures

- Structural / Data flow (L0.2)
  - TM/TC
  - FDIR alarms
  - Devices
L0 Level: some AADL issues

- Mode switch procedures: how to describe actions to start when a mode switch occurs

- Composites states/modes
L1 Model: dynamic architecture (design)

- **L1.0**: semi-automatically generated model from L0.2
  - Support to design through the use of **patterns**
  - Reuse L0 information through **thread binding** mechanism

- **L1.1**: **temporal properties** (L1.1)
  - Dispatch protocol
  - Compute execution time
  - Deadline

- Simulation possible using **Cheddar**
L1 Model: Thread binding

- Traceability link: which thread executes a given function

-- The Allowed_Thread_Binding property specifies the set of threads that
-- are available for binding. The set is specified by a list of thread and
-- system component names. System names represent the threads contained in
-- them.

Allowed_Thread_Binding: inherit list of reference(thread, system)

applies to |system|

-- The Allowed_Thread_Binding_Class property specifies a set of thread and
-- system classifiers. These component classifiers constrain the set of
-- threads in the Allowed_Thread_Binding property to the subset that
-- satisfies the component classifier. The default value is inherited from
-- the containing system component. If this property has no associated
-- value, then all threads specified in the Allowed_Thread_Binding are
-- acceptable candidates.

Allowed_Thread_Binding_Class: inherit list of classifier(thread, system)

applies to |system|

-- System components of functional architecture are bound to the thread
-- specified by Actual_Thread_Binding property.

Actual_Thread_Binding: inherit reference|thread|

applies to |system|
L1 Model: how to model tasks synchronizations?

- L1.2: Periodic tasks have to synchronize with IO events
  - Tasks “with suspension”
L1 Model: modeling issues

- Event combination
- Multiple deadlines

thread PL_CYCL

features

-- Synchronization events
PLB PL ACQ in event port
   ArchDyn Properties::Arrival_Release => On Dispatch;
ArchDyn Properties::Absolute_Time => \( 125 \text{ ms} \);
};
PL_CYCL in event port
   ArchDyn Properties::Arrival_Release => On Dispatch;
ArchDyn Properties::Absolute_Time => \( 75.4 \text{ ms} \);
};
AOCS_DA3 in event port
   ArchDyn Properties::Arrival_Release => On Dispatch;
ArchDyn Properties::Absolute_Time => \( 114.7 \text{ ms} \);
};

properties

Dispatch_Release => Periodic;
Period => \( 125 \text{ ms} \);
SEI::Priority => 60;
end PL_CYCL;

thread implementation PL_CYCL.N1

annex Core_Behavior {

states

-- Application request (TC, MPO etc?) processing.
RQ : initial state;
-- Acquisitions/Control/Commands, cyclic TR and monitoring.
ACC : state;
-- DA3 command sending.
DA : state;
-- Wait states
waitingAcq : state;
waitingAOCS : state;
exit : final state;

transitions

-- PAYLOAD Requests management
RQ => waitingAcq (computation(1.9 ms, 1.9 ms));
-- Waiting for the end of PAYLOAD acquisitions
waitingAcq [PLB PL ACQ] => ACC ();
-- Acquisitions/Control/Commands, cyclic TR and monitoring.
ACC => waitingAOCS (computation(3.8 ms, 3.8 ms));
-- Waiting for the end of FF and AOCS processing (2 events)
-- Since there is no gate mechanism in Behaviour annex, two ways
-- are used : AOCS evt received first and FF evt received first.
waitingAOCS [PF_DA3] => waitingAOCS ();
--- PA ---
waitingAOCS [AOCS_DA3] => waitingPF ();
waitingAOCS [AOCS_DA3] => DA ();
waitingPF [PF_DA3] => DA ();
-- DA3 command sending.
DA => exit (computation(1.9 ms, 1.9 ms));

end PL_CYCL.N1;
L1 Model: Environment model (contracts ?)

- New properties to describe hypothesis on features
- Allow to simulate and verify step by step the model (even non-complete)
- Unitary tests when ports are not connected (hypothesis on env.)
- Then port connections replace arrival laws (integration tests)
L2 model: Concrete architecture (detailed design)

- SW static architecture + Platform
- Object-oriented design: objects, methods, data => data components
- RTOS API, User libraries => packages & hierarchical data components

```plaintext
object state: data basic::enumerated
  |ArchifDyn_Proper::Enumerated_Range => ("OFF", "ON"));
is_failed: data basic::boolean;
heaters_sel_conf: data basic::enumerated
  |ArchifDyn_Proper::Enumerated_Range => ("A", "B"));

coarse_HF_cycles_counter: data;
fine_HF_cycles_counter: data;
coarse_line: data;
fine_line: data;
-- Coarse regulation
    tab_coarse_last_command: data;
tab_coarse_temp: data;
    -- Fine regulation: fine_sync_TM
    tab_fine_temp: data;
tab_fine_int_power: data;
tab_fine_duty_cycle: data;
tab_fine_bus_voltage: data;
    -- Fine regulation: other data
    tab_fine_int_power_state_prev: data;
tab_fine_target_temp: data;
tab_fine_nb_next_ON_cycles: data;
    -- Logical lines and corresponding physical areas
    ... nb_max_lines_coarse: data;
end THR_CTRL;
```

```plaintext

MGR

FCT

CTRL

SWR

HAL

THR_CTRL

Get_State

Switch_On

Switch_Off
```

---

EADS Astrium © 2006
L2 model: platform dependent design

- Manage the way the architecture is implemented through connections mapping
  - Which object, mechanism or primitive provided by the execution platform is used to implement the L1 model

- Can be semi-automatically generated from L1 through implementation patterns

- Allow to detect a shared data that has to be protected
Process & Methodology: steps

- Identify functions & interactions
- Build a functional architecture
- Build a hardware architecture
- Bind functional to hardware
- Specify / Design execution components (tasks)
- Choose / Customize execution platform (RTOS and means of communication)
Process & Methodology: steps

- Identify functions & interactions
- Build a functional architecture
- Build a hardware architecture
- Bind functional to hardware
- Specify / Design execution components (tasks)
- Choose / Define execution platform (RTOS and means of communication)

Objectives:

**Incremental & Iterative** approach
- using patterns and refinements between each increment or iteration

Go downward to **code generation**
- progressive modeling, the model is kept during all the life cycle

L0 Functional architecture
L1 Logical architecture
L2 Concrete architecture
ANALYSIS

Critical tasks scheduling

CPU Load

63%

Temporal characteristics

LO -> L1 Patterns

SYSTEM

AOCS

RTC

ICB

Processor

T1

T2

Ti

Critical tasks scheduling

SYSTEM

AOCS

RTC

ICB

Processor

125ms

20ms

L0 -> L1 Patterns

Temporal characteristics

SYSTEM

AOCS

RTC

ICB

Processor

125ms

20ms

Analyzing CPU Load

63%
Concrete mechanisms
Implementation Patterns

Data flows

Scheduling analysis, tasks with suspension

Data flow analysis

Mutual exclusion, accurate behavior simulation
Progressive modeling
AADL Assessment: Benefits

- Currently no language is used to support Level 1 activities. AADL is a very good candidate to improve them.

- System view: functional, software, hardware

- Modes: highly used in space systems
  - AOCS modes
  - System modes
  - Hardware modes

- Reuse
  - Components
  - Patterns / Frameworks
## AADL Assessment: some issues...

### Modeling issues
- Structure Vs Behavior
- Linked threads or behavior annex
- Logical / Concrete (dispatch, protocols, …)

### Osate issues
- Subpackages
- Data subcomponent access
- …

### Topcased issues
- Access connections
- Diagram export function
- …

### Wish list
- Composite states
- Mode dependent features
- Connection binding
- Interrupt handling (IT handlers)
- Thread dispatch refinement
- Nested port connections
- Multiple inheritance
- Double-Port memory modeling
- Variable dequeue protocols
- Event combination
- Abstract ports
- Data subprogram reference
- …
Limitations

- **Tools**
  - Graphical editor
  - Model transformation, links between models (binding),
  - Analysis

- **Language**
  - Behavior annex status

- **Modeling**
  - Modeling rules for quality
  - Version management (iterations, increments, …)
  - Behavior description guidelines (several abstraction levels)
Perspectives: Model-based engineering at Astrium

- **Objectives**
  - Support to system & software design (including reuse)
  - Support to early V&V
  - Support to automatic code generation (also for rapid prototyping)

- **Perspectives**
  - L-1: Matlab/Simulink and UML
  - L0: UML
  - L1: AADL
  - L2: we do not really require such a detailed model
Questions ?