AADL performance analysis with Cheddar: a review

P. Dissaux*, J. Legrand*, A. Plantec+, F. Singhoff+

*Ellidiss Technologies, France
+University of Brest/UBO, LISyC, France
Talk overview

1. **Cheddar project: context and motivations**
2. «Design pattern» approach
3. «Exhaustive simulations» approach
4. Roadmap for the next year
Real time scheduling theory

1. Analytical methods/feasibility tests:

\[ \sum_{i=1}^{n} \frac{C_i}{P_i} \leq 69\% \]

2. **Simulation**: compute GANTT + analysis. Sometimes exhaustive simulation (hyper-period).

---

**Context**
Cheddar project: context and motivations

- Few industrial projects apply real time scheduling theory.
- Cheddar project: expects to increase the usability of real time scheduling theory.
- Some milestones:
  1. Started in May 2000 by the Univ. of Brest.
Two approaches to investigate performances of AADL models

1. «Design pattern» approach:
   - Choose architecture examples proposing usual designs for the synchronization/communication between AADL threads. Assign them feasibility tests that we can automatically apply.
   - Which design patterns and feasibility tests should we select?

2. «Exhaustive simulations» approach:
   - When architecture models are composed of specific schedulers/thread models. Modeling and verification with exhaustive simulations.
   - Which modeling language? How to automatically build simulation software?
«Design pattern» approach: which AADL design patterns?

1. **Four examples of thread/synchronization/communication designs** related to standards, practitioners, predictability, flexibility: Synchronous data flows, Ravenscar, Blackboard, Queued Buffer.

2. **Define properties that we look for:**
   - A. Worst case thread response times.
   - B. Bounds on the thread waiting time due to data access.
   - C. Deadlocks and priority inversions due to data access.
   - D. Memory footprint analysis.

3. **Define design patterns to be analyzed:**

<table>
<thead>
<tr>
<th>Predictability</th>
<th>Flexibility</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>1. Synchronous Data flows ..........</td>
<td></td>
</tr>
<tr>
<td>2. Ravenscar .........................</td>
<td></td>
</tr>
<tr>
<td>3. Blackboard..........................</td>
<td></td>
</tr>
<tr>
<td>4. Queued Buffer.......................</td>
<td></td>
</tr>
</tbody>
</table>

### Performance Criteria

<table>
<thead>
<tr>
<th>Context</th>
<th>Design-Pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A</td>
</tr>
<tr>
<td></td>
<td>A</td>
</tr>
<tr>
<td></td>
<td>B</td>
</tr>
<tr>
<td></td>
<td>C</td>
</tr>
<tr>
<td></td>
<td>A</td>
</tr>
<tr>
<td></td>
<td>B</td>
</tr>
<tr>
<td></td>
<td>C</td>
</tr>
<tr>
<td></td>
<td>D</td>
</tr>
</tbody>
</table>
The «Ravenscar» design pattern (1/3)

- **Description:** Ravenscar profile of Ada 2005 = periodic threads + fixed priority scheduling + shared data + PCP.

- **Properties to check:** worst case thread response time + Bounds on the thread waiting time due to data access (feasibility test).

- **How to perform analysis:**
  - Analytical methods exist and are already implemented into Cheddar.
  - How to express Ravenscar with AADL version 1 in order to perform analysis with Cheddar?
    1. Definition of Cheddar specific properties (thread, processor, data).
Example 1: (AADL version 1)

```plaintext
thread implementation T3.i
  properties
  Source_Text => "mes_threads.c";
  Dispatch_Protocol => Periodic;
  Compute_Execution_time => 1 ms .. 2 ms;
  Deadline => 10 ms;
  Period => 10 ms;
end T3.i;
thread implementation fifo2.i
  properties
  Dispatch_Protocol => Background;
  Compute_Execution_time => 1 ms .. 3 ms;
  Cheddar_Properties::POSIX_Scheduling_Policy => SCHED_FIFO;
  Cheddar_Properties::Fixed_Priority => 5;
  Cheddar_Properties::Dispatch.Absolute_Time => 4 ms;
end fifo2.i;
process implementation proc0.i
  subcomponents
  a_T3 : thread T3.i;
  ....
processor implementation rma_cpu.i
  properties
  Scheduling_Protocol => RATE_MONOTONIC;
  Cheddar_Properties::Preemptive_Scheduler => true;
  Cheddar_Properties::Scheduler_Quantum => 3 ms;
end rma_cpu.i;
system implementation a_system.impl
  subcomponents
  a_cpu : processor rma_cpu.i;
  an_application : process proc0.i;
  properties
  ...
```

Context Design-Pattern
The «Ravenscar» design pattern (3/3)

Compute simulation

Analysis from scheduling simulation or with feasibility tests (eg. deadlines, response times)

Scheduling simulation, Processor arinc :
- Number of preemptions : 760
- Number of context switches : 3205
- Task response time computed from simulation :
  T1 => 6/worst 6/best 6.00000/average
  T2 => 56/worst 35/best 46.81667/average
  T3 => 10/worst 4/best 6.00000/average
  T4 => 1/worst 1/best 1.00000/average
- No deadline missed in the computed scheduling : the task set seems to be schedulable.
The «queued buffer» design pattern (1/3)

- **Description:** AADL event data ports model message exchanges between threads. Events can be stored into buffer before consumption.

- **Properties to check:** worst case thread response time (feasibility test) + port memory footprint.

- **How to perform analysis:**
  - No feasibility test available to check memory footprint with AADL periodic/aperiodic threads and real time schedulers.
  - Memory footprint analysis with queueing system theory: feasibility tests based on queueing system models (J. Legrand)
The «queued buffer» design pattern (2/3)

Example 2: event data port connections

processor implementation cpu_rm.i
properties
Scheduling_Protocol => Rate_Monotonic;
...
end cpu_rm.i;
process implementation p0.i
subcomponents
Producer1 : thread Producer.i;
Producer2 : thread Producer.i;
Consumer1 : thread Consumer.i;
connections
event data port Producer1.Data_Source ->
Consumer1.Data_Sink;
event data port Producer2.Data_Source ->
Consumer1.Data_Sink;
end p0.i;

thread Producer
Features
Data_Source : out event data port;
end Producer;
thread Consumer
features
Data_Sink : in event data port;
end Consumer;

thread implementation Producer.i
properties
Dispatch_Protocol=>periodic;
...
end Producer.i;
thread implementation Consumer.i
properties
Dispatch_Protocol=>periodic;
...
end Consumer.i;
The «queued buffer» design pattern (3/3)

- Buffer simulation
- Analysis from simulation
- Worst case queueing system analysis (based on P/P/1)
Two approaches to investigate performances of AADL models

1. «Design pattern» approach:
   - Choose architecture examples proposing usual designs for the synchronization/communication between AADL threads. Assign them feasibility tests that we can automatically apply.
   - Which design patterns and feasibility tests should we select?

2. «Exhaustive simulations» approach:
   - When architecture models are composed of specific schedulers/thread models. Modeling and verification with exhaustive simulations.
   - Which modeling language? How to automatically build simulation software?
Exhaustive simulations» approach

- The Cheddar language is composed of 2 parts:

  1. An Ada subset modeling the arithmetic/logical statements:

  2. A timed automaton language modeling timed synchronization:
      - Network of simplified UPPAAL automata.
      - States, transitions, variables, clocks, synchronizations, guard, actions (Cheddar subprograms).
      - Why timed automata? Usual language to model schedulers. Cheddar tools could be used with AADL/Behavioral annex.
Example of an hierarchical scheduler (1/3)

- **Partition** = application with timing and memory isolation.

- **ARINC 653 scheduling** (hierarchical scheduling):
  1. Compute when each partition must be activated. This scheduling is fixed at design time.
  2. Tasks of a given partition are scheduled all together with a fixed priority scheduler (e.g. Rate Monotonic).

- **Ravenscar + Synchronous data flows.**
Example of an hierarchical scheduler (2/3)

Modelling such a kind of hierarchical system with AADL version 1 require to:

1. Model the architecture point of view (AADL V1).
2. Model the scheduler behavior (Cheddar programs) => Behavioral annex with AADL V2.
Example of an hierarchical scheduler (3/3)

- **Restart**
  - partition\_clock:=0
  - partition\_clock=10
  - wakeup1?
  - partition\_clock=6

- **Pended**
  - partition\_duration:=0
  - wakeup1!
  - partition\_duration= partition\_capacity
  - partition\_election!

- **Wait\_Priority**
  - partition\_duration< partition\_capacity
  - partition\_priority!

- **Activate\_Partition1**
  - Activate\_Partition2

**Context**
- Design-Pattern
- Simulation
Engineering process to build and use a Cheddar model

1. Design and test the model with the Cheddar program interpreter.
2. Generate simulation software with Platypus: Ada components integrated into Cheddar. Large scale simulations.
1. Cheddar project: context and motivations
2. «Design pattern» approach
3. «Exhaustive simulations» approach
4. Roadmap for the next year
Roadmap for the next year

☐ Current status Cheddar/AADL:
  • Cheddar web site: [http://beru.univ-brest.fr/~singhoff/cheddar](http://beru.univ-brest.fr/~singhoff/cheddar)
  • Open-source but industrial Support from Ellidiss Technologies.
  • AADL V1 only, relies on Ocarina (ENST, [http://ocarina.enst.fr](http://ocarina.enst.fr)).

☐ Roadmap for 2009/2010:
  1. Annual release of Cheddar
     • Corrected bugs
     • May be, a start of AADL version 2 support
  2. Tool integration within the ESA/ASSERT-Lab
  3. Modeling/simulation of schedulers
     • Design Cheddar program for “usual” schedulers: POSIX, aperiodic servers, Peter’s case study …
     • Development of Cheddar program tools.
     • Behavioral annex models and tools implemented for Cheddar Programs.
  4. New AADL design patterns