AADL performance analysis with Cheddar : a summary

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

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

1. The Cheddar project: context and motivations
2. Simple real time scheduling analysis with Cheddar/AADL
3. Multi-resources analysis with Cheddar/AADL
4. Using user-defined schedulers and thread dispatching rules with Cheddar/AADL
5. Conclusion and roadmap
The Cheddar project: context and motivations (1/2)

Real time scheduling theory:

1. Analytical analysis (feasibility tests):
   \[ \sum_{i=1}^{n} \frac{C_i}{P_i} \leq 69\% \]

2. Scheduling Simulation analysis:
   - Compute time-lines and perform analysis (e.g., check thread deadline).
   - Sometimes leads to a proof (model-checking = simulation on base period).
Few industrial projects apply real time scheduling theory.

Cheddar project expects to increase its usability by:
- Providing tools which allow to automatically perform analysis.
- Investigating relationships with design languages (AADL).
- Extending the theory with practitioner requirements (e.g., memory footprint analysis).

Cheddar project:
1. Started in May 2000 by the Univ. of Brest.
2. November 2004, partnership with ENST (Cheddar relies on Ocarina).
Talk overview

1. The Cheddar project: context and motivations
2. Simple real time scheduling analysis with Cheddar/AADL
3. Multi-resources analysis with Cheddar/AADL
4. Using user-defined schedulers and thread dispatching rules with Cheddar/AADL
5. Conclusion and roadmap
Simple real time scheduling analysis with Cheddar/AADL (1/3)

- Simplest way to use Cheddar: give AADL V1 properties + some other Cheddar specific properties.

- AADL V1 provides most of required properties. Extra required properties:
  - Properties related to usual schedulers (e.g., POSIX 1003.1b properties, quantum, preemptivity, ...).
  - Thread properties (e.g., jitter, offset, priority ...)
  - When shared resources are accessed by thread? Thread behavior?
  - Ambiguities to express thread precedence relationships from AADL connections.
Simple real time scheduling analysis with Cheddar/AADL (2/3)

Example 1: periodic thread + POSIX 1003.1b scheduler

```
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;
    ....
  properties
    Scheduling_Protocol => RATE_MONOTONIC;
    Cheddar_Properties::Preemptive_Scheduler => true;
    Cheddar_Properties::Scheduler_Quantum => 3 ms;
end proc0.i;

processor implementation rma_cpu.i
  properties
    Scheduling_Protocol => RATE_MONOTONIC;
    Cheddar_Properties::Preemptive_Scheduler => true;
  properties
    Scheduling_Protocol => RATE_MONOTONIC;
    Cheddar_Properties::Preemptive_Scheduler => true;
end rma_cpu.i;

system implementation a_system_Impl
  subcomponents
    a_cpu : processor rma_cpu.i;
    an_application : process proc0.i;
  properties
    ...
```
Simple real time scheduling analysis with Cheddar/AADL (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.
Talk overview

1. The Cheddar project: context and motivations
2. Simple real time scheduling analysis with Cheddar/AADL
3. Multi-resources analysis with Cheddar/AADL
4. Using user-defined schedulers and thread dispatching rules with Cheddar/AADL
5. Conclusion and roadmap
An AADL model may contain information on different resources (processor, memory, networks, …).

AADL allows to jointly manage several resources.

Thread communications by event data ports.

Memory footprint analysis with queueing system analytical tools.
Queueing system models: define producer rate and consumer rate, in order to compute criteria such as message waiting time or number of waiting messages.

Define new queueing system models M/P/1 and P/P/1:
- Take into account AADL threads dispatching (periodic, sporadic).
- Take into account thread scheduling (e.g., Rate Monotonic).

Define feasibility tests from these queueing systems models ⇒ worst case memory footprint analysis based on P/P/1.
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 implementation Producer.i
  properties
    Dispatch_Protocol=>periodic;
    ...
  end Producer.i;
thread implementation Consumer.i
  properties
    Dispatch_Protocol=>periodic;
    ...
  end Consumer.i;
Multi-resources analysis (4/4)

Buffer simulation

Analysis from simulation

Worst case queueing system analysis (based on P/P/1)
AADL tools interoperability

- For most AADL designers, using Cheddar alone is difficult => investigate Stood and Cheddar interoperability.

- Define properties that we look for:
  A. The worst case thread response times;
  B. The bounds on the thread waiting time due to data access;
  C. The deadlocks and priority inversions due to data access,
  D. …

- Define design patterns to be analyzed:
  1. Synchronous Data flows
  2. Mutex protected shared Data
  3. Blackboard
  4. Queued Buffer
Talk overview

1. The Cheddar project: context and motivations
2. Simple real time scheduling analysis with Cheddar/AADL
3. Multi-resources analysis with Cheddar/AADL
4. Using user-defined schedulers and thread dispatching rules with Cheddar/AADL
5. Conclusion and roadmap
The Cheddar domain specific language (1/6)

- The Cheddar language aims at modelling of real time schedulers and thread dispatching rules.

- Modelling a real time scheduler requires:
  
  1. Modelling arithmetic and logical statements (eg. how to compute priorities, how to select a thread).
  2. Modelling timing and synchronization relationships between threads and schedulers (eg. when threads must be released by schedulers, how schedulers must work all together, ...).
The Cheddar domain specific language (2/6)

- The Cheddar language is composed of 2 parts:

1. An Ada subset modelling the arithmetic/logical statements:
   - A Cheddar program is a set of sections (sub-programs).
   - Types of section:
     - Start sections: variable declaration and initialization.
     - Priority sections: compute priorities during simulation.
     - Election sections: choose the thread to run.

2. A timed automaton language modelling timed synchronization:
   - A set of UPPAAL like timed automata modelling thread and scheduler behavior.
   - States. Transitions. Transitions may express synchronization, guards and clock statements.
The Cheddar domain specific language (3/6)

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

- **ARINC 653 scheduling** (hierarchical scheduling) :
  1. Choose when each partition must be activated. This scheduling is fixed at design time.
  2. Run tasks of a given partition according to a fixed priority scheduler (eg. Rate Monotonic).
The Cheddar domain specific language (4/6)

- Modelling such a kind of hierarchical system with AADL version 1 require to:
  - Model the architecture point of view (AADL V1).
  - Model the scheduler behavior (Cheddar programs).
The Cheddar domain specific language (5/6)

Example 3: AADL Version 1
modelling the architecture

thread T1 ...
thread T2 ...
...
process implementation partition1.Impl
  subcomponents
    T3 : thread T3.Impl;
    T4 : thread T4.Impl;
  properties
    Scheduling_Protocol
    => Automaton_User_DEFINED_Protocol;
end partition1.Impl;

processor implementation arinc.Impl
  properties
    Scheduling_Protocol
    => Automaton_User_DEFINED_Protocol;
...
system implementation auto_arinc.Impl
  subcomponents
    arinc : processor arinc.Impl;
    partition1 : process partition1.Impl;
    partition2 : process partition2.Impl;
  properties
    Actual_Processor_Binding => reference
                            arinc applies to partition1;
    Actual_Processor_Binding => reference
                            arinc applies to partition2;
end auto_arinc.Impl;
The Cheddar domain specific language (6/6)

Cheddar programs modelling a hierarchical scheduler:

- **Partition 1**
  - **Activate_Partition1**
    - **wait1**?
    - **partition_clock:=0**
    - **partition_clock:=10**
  - **Activate_Partition2**
    - **wait2**?
    - **partition1_duration:=0**
    - **wakeup1**!
  - **Restart**
  - **Pended**
    - **partition1_duration:=0**
    - **wakeup1**!
    - **partition1_duration = partition1_capacity**
  - **Wait_Priority**
    - **Logical and arithmetic statements**
    - **party1_priority!**

- **Partition 2**
Talk overview

1. The Cheddar project: context and motivations
2. Simple real time scheduling analysis with Cheddar/AADL
3. Multi-resources analysis with Cheddar/AADL
4. Using user-defined schedulers and thread dispatching rules with Cheddar/AADL
5. Conclusion and roadmap
Conclusion and roadmap

Current status of Cheddar/AADL:
- First release in November 2005.
- Cheddar web site: http://beru.univ-brest.fr/~singhoff/cheddar

Roadmap:
   - Fixed bugs + Cheddar language with AADL version 1
2. November 2008, Stood/Cheddar experiments:
   - AADL tool interoperability: design patterns in AADL/behavioral annex.
   - Behavioral annex meta-model and Ada parser (should work with Ocarina).
   - Towards AADL V2: from Cheddar program to behavioral annex?