Cheddar Update and Research Roadmap

F. Singhoff+, P. Dissaux*, J. Legrand*, V. Gaudel+, A. Plantec+, S. Rubini+

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

Talk overview

1. Cheddar project : context and motivations
2. Verification approaches
3. Update and Research Roadmap

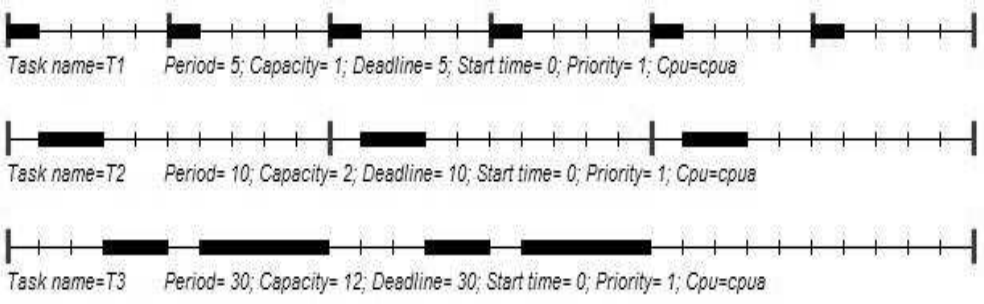
Cheddar project : context and motivations

- **Scheduling verification with real-time scheduling theory:**
  - **Modeling functions:** simplified models such as Liu & Layland periodic thread: processor demand + deadline.
  - **Classical schedulers:** Rate Monotonic, Earliest Deadline First.
  - **Analysis:** either with feasibility tests or simulations.

1. **Feasibility tests** : $$R_i = C_i + \sum_{j \in hp(i)} \left\lceil \frac{R_i}{P_j} \right\rceil \cdot C_j$$

   => architectures must meet applicability assumptions of the feasibility test.

2. **Simulation**:



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.**
  - Started in 2002 by Univ. of Brest, partnership with Ellidiss Tech. (provide open source and industrial support) since 2008.
  - Current project members (U. Brest & Ellidiss Tech.) : A. Plantec, S. Rubini, V. Gaudel, P. Dissaux, J. Legrand and F. Singhoff.
  - Support : Ellidiss Tech. and Conseil Régional de Bretagne
  - Other contributors: Télécom-Paris-Tech, ISAE, Univ. Lisboa

Talk overview

1. Cheddar project : context and motivations
2. Verification approaches
3. Update and Research Roadmap

Verification approaches

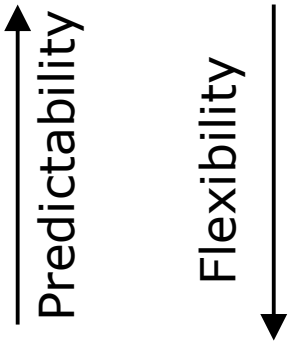
1. «Design pattern» approach
2. «Exhaustive simulations» approach

«Design pattern» approach:

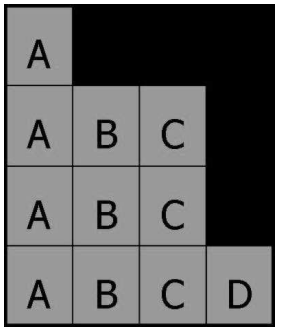
- **Define a set of AADL architecture design patterns of real-time systems.**
  - = models a typical thread communication/synchronization.
  - = set of constraints on AADL components/properties
- **For each design pattern, define feasibility tests that can be applied according to their applicability assumptions.**
- **Verification of a real-time system architecture model by a designer:**
  1. He checks compliancy of his model to one of the design-patterns … which then gives him which feasibility tests he can apply.
  2. Perform verifications with a tool implementing these feasibility tests (Cheddar tool).

«Design pattern» approach:

- **Examples of design-patterns :** Ravenscar, Synchronous data flow, Queued buffer, Blackboard, Unplugged. Related to standards and practitioner usages. Levels of predictability/flexibility.
- **Example of performance criteria that we look for and feasibility tests to compute them :**
  - A. Worst case thread response times.
  - B. Bounds on the thread waiting time due to data component access.
  - C. Deadlocks and priority inversions due to data component access.
  - D. Memory footprint analysis.
- **Criteria/Feasibility tests assigned to Design-pattern :**



1. Synchronous Data flows.....
2. Ravenscar ……...
3. Blackboard.....
4. Queued Buffer.....



Verification approaches

1. «Design pattern» approach
2. «Exhaustive simulations» approach

«Exhaustive simulations» approach

- **When to apply this type of verification:**
  - When architecture models are composed of specific schedulers, specific thread models or specific thread synchronizations. No feasibility tests available.
  - Assume deterministic schedulers and periodic/sporadic thread model.

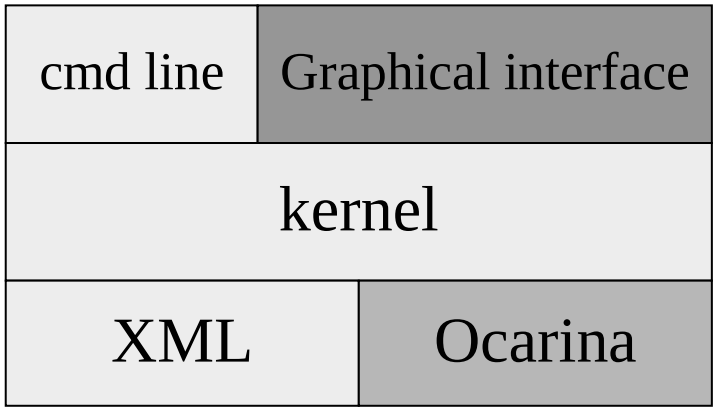
- **How to apply this verification:**
  1. Model specific scheduler.
  2. Analysis by exhaustive simulations : run scheduling simulation during hyper-period.

«Exhaustive simulations» approach

- **Cheddar language is composed of 2 parts:**
  1. **An Ada subset modeling the arithmetic/logical statements:**
  2. **A timed automaton language modeling timed synchronization between components/schedulers:**
     - Automata similar to simplified UPPAAL automata.
     - States, transitions, variables, clocks, synchronizations, guard and actions.
     - Why timed automata : usual language to model schedulers. Cheddar tools could be used with AADL/Behavioral annex.

- **Engineering process to handle Cheddar programs:**
  1. Design and test with the Cheddar program interpreter.
  2. Generate simulation software with Platypus : Ada components integrated into Cheddar tool. Large scale simulations/architectures.

Current Implementation of these verification approaches



Cheddar =

Cheddarlite =

Cheddarkernel =

- Architectures specified by “XML Cheddar” or AADL.
- Several AADL parsers:
  - Included into Cheddar : Ocarina, AADL V1
  - Tools producing “XML Cheddar”:
    1. AADLInspector (Ellidiss Tech, AADL V2)
    2. Ocarina AADLV2 (ISAE/Télécom-Paris-Tech)
    3. OSATE2 plug-in (Télécom-Paris-Tech, AADLV2)

Talk overview

1. Cheddar project : context and motivations
2. Verification approaches
3. Update and Research Roadmap

Update and Research Roadmap

- **Projects started in 2010:**
  1. AADL design-pattern recognition
  2. Multi-core real-time scheduling and cache analysis and AADL
  3. ARINC 653 scheduling analysis

Update and Research Roadmap

❑ AADL design-pattern recognition:

- ❑ **Support (2010/2013) :** Ellidiss Technologies & Conseil régional de bretagne.

- ❑ **Project objectives:**
  - ❑ Checking compliancy to a design-pattern of an AADL model is difficult to do for architecture designers.
  - ❑ How to automatically check compliancy ?
  - ❑ How to help the designer when his models are not compliant to any design-pattern? How to change the architecture model to make it compliant ?

- ❑ **Project status (Phd of Vincent Gaudel):**
  1. Component/property constraints of each AADL design-pattern are modeled with EXPRESS.
  2. Recognition tool is produced with a model driven engineering process from EXPRESS models of (1). Platypus.
  3. Investigate how to compose different AADL design-patterns

Update and Research Roadmap

- **Multi-core real-time scheduling and cache analysis with AADL:**
  - Real-time scheduling theory for multi-core processors is becoming mature … and there are very vew tools implementing multi-core real-time scheduling features..
  - Real-time scheduling theory for multi-core processors usually do not take care of cache systems.
  - Arrival of Stéphane Rubini, multi-core/cache systems expert.

- **Project objectives:**
  - How to model these systems (cache/memory systems) with AADL ?
  - How to help AADL designers to predict thread behavior with caches/multi-core real-time scheduling.

- **Project status:**
  - ISAE (J. Hugues).
  - Investigate the AADL modelling of the memory layout of a mono-processor target (VxWorks).
  - Investigate verification of compliancy of AADL models with Vxworks memory layout. REAL.

Update and Research Roadmap

- **ARINC 653 scheduling analysis:**
  - **Current Cheddar support of ARINC 653** is limited: exhaustive simulation only.
  - **University of Lisboa, in the context of the AIR project:**
    - Designed specific bin-packing methods for ARINC 653
    - Designed specific feasibility tests for ARINC 653
  - **Project objectives:**
    - Experiment U. Lisboa bin-packing/feasibility tests..
    - Implement ARINC features in Cheddar.
  - 2 years project between U. Brest and U. Lisboa.
  - Not directly related to AADL but may provide a new ARINC tool for the AADL community.

Update and Research Roadmap

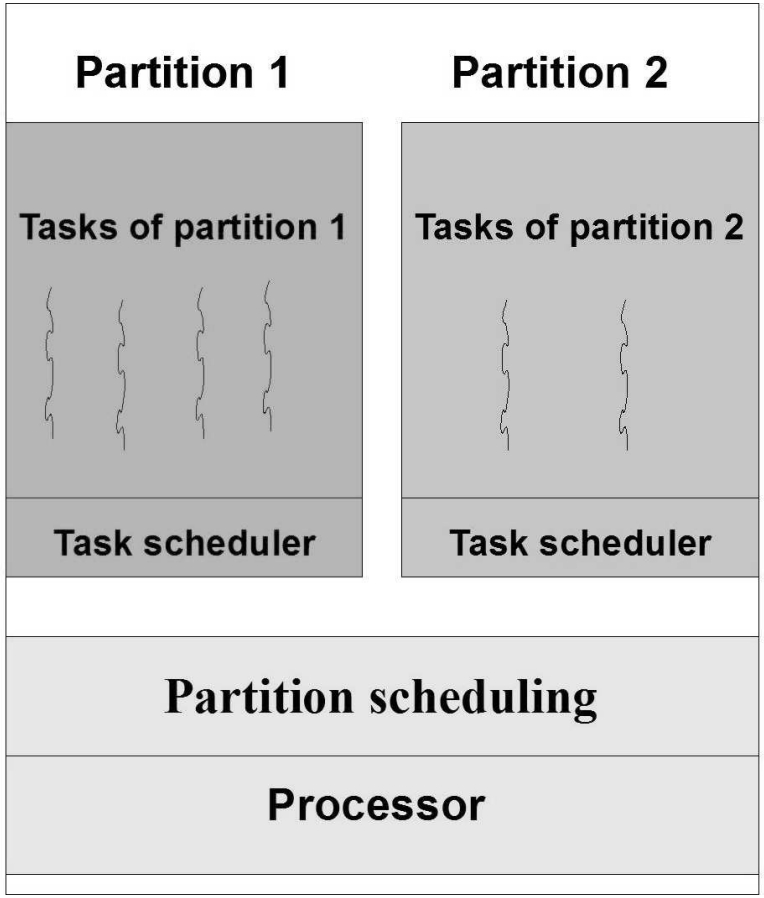
- **Projects started in 2010:**
  1. AADL Design-pattern recognition tool => *Vincent Gaudel's talk.*
  2. Multi-core scheduling & cache/memory analysis => memory AADL modeling and verification => *Stéphane Rubini's talk.*
  3. ARINC 653 scheduling analysis.

- **Cheddar current status:**
  - Cheddar web site: http://beru.univ-brest.fr/~singhoff/cheddar
  - Last release: July 2008

- **Roadmap : next release on summer 2011**
  - First release of the AADL design-pattern recognition tool
  - Multi-core real-time scheduling : global RM/DM/FP/EDF/LLF and Pfair scheduling
  - (Probably) first cache analysis tool
  - Major update of the Cheddar XML grammar (refactored by Platypus)
  - Various bug fixes (see Ellidiss trac website).

Example of an hierarchical scheduler (1/2)

- **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).



Example of an hierarchical scheduler (2/2)