Cheddar : about the usability of the real-time scheduling theory

P. Dissaux*, J. Legrand*, A. Plantec+, S. Rubini+, L. Lemarchand+, V. Gaudel+, S. Li+, C. Fotsing+, F. Singhoff+

***Ellidiss Technologies, France**

+University of Brest/UBO, Lab-STICC/UMR 6485, France

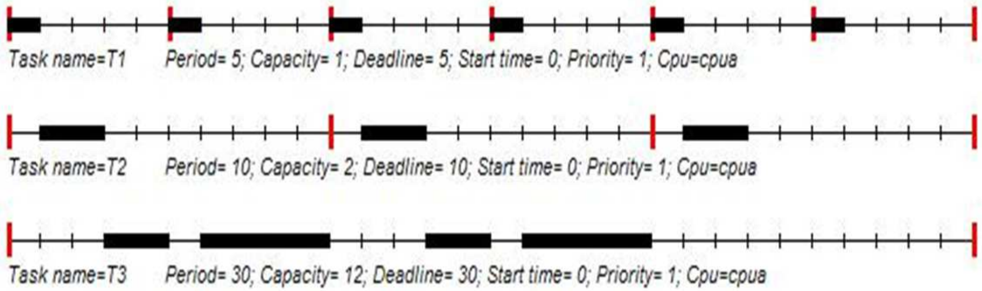
Talk overview

1. Cheddar : context and motivations
2. What we investigate
3. A design pattern approach to enforce scheduling analysis
4. Conclusions and ongoing works

About scheduling analysis and its use

- **Real-time scheduling theory:**
  - **Simplified models of functions :** e.g. periodic task: processor demand + deadline.
  - **Analysis:** either with feasibility tests or simulations.

1. **Scheduling Simulation**:



2. **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 assumptions of the feasibility test.

Does people really use real-time scheduling theory ?

- **Real-time scheduling theory, verification with analytical methods and/or simulations:**

  1. Most important theoretical results proposed between 1974 and 1994 (simple uniprocessor architectures).
  2. Technologies are compliant with (POSIX 1003.1b operating systems, Ada/Ravenscar profile, …).
  3. Strong demand from engineers.
  4. …

Not used as much we can expect ☹

Some possible explanations

1. **This theory can not be applied on some architecture types** (e.g. multiprocessor/distributed/hierarchical systems).

2. **Require advanced skills to be used:**
   - ❑ Numerous theoretical results: how to choose the right one ?
   - ❑ Numerous assumptions for each result.
   - ❑ How to abstract/model a system to access schedulability ? (e.g. dependency)

3. **Engineers must be helped to use tools:**
   - ❑ How and when performing this analysis ?
   - ❑ How to write models to be analyzed ? Which design languages ?
   - ❑ How to safely use scheduling tools ?

4. **...**

Cheddar : context and motivations

- **How to increase usability of the real-time scheduling theory?**

  - Started in 2002 by Univ. of Brest, partnership with Ellidiss Tech. (provides industrial support) since 2008.
  - **Main supports :** Ellidiss Tech., Brittany council, Thalès communication, OSEO
  - **Other contributors :** Télécom-Paris-Tech, ISAE, Univ. Lisbon, Virtualys

Talk overview

1. Cheddar : context and motivations
2. What we investigate
3. A design pattern approach to enforce scheduling analysis
4. Conclusions and ongoing works

Cheddar : what we investigate

3. **Engineers must be helped to use tools:**

- **Cheddar tool =**

  analysis framework (queueing system theory & real-time scheduling theory)

  + ADL (architecture description language)

  + simple model editor.

- **Two versions :**
  - Open source (Cheddar) : educational and research.
  - Industrial (AADLInspector) : Ellidiss Tech product.

Only providing scheduling tools is not enough

Cheddar : what we investigate

2. Require advanced skills to be used:

- Numerous theoretical results: how to choose the right one ?
- Numerous assumptions for each result.

Investigate if ADLs may help to use scheduling analysis tools:

- Model based approach
- Involved in AADL standardization. MARTE/UML (Thalès communication).
- Standard ADLs help : modeling of real-time architecture, pivot language
- **But usually not enough : most of the time too flexible or to rich**

But how to be sure that an ADL model can be analyzed by a scheduling tool ?

What are the compliant feasibility tests ?

What we look for : how to automatically perform scheduling analysis ?

Talk overview

1. Cheddar : context and motivations
2. What we investigate
3. A design pattern approach to enforce scheduling analysis
4. Conclusions and ongoing works

A “design pattern” approach to increase real-time scheduling usability

- **Define a set of architectural design patterns of real-time systems.**
  - = models a typical task communication or synchronization.
  - = set of constraints on entities/properties of the system.
- **For each design pattern, define feasibility tests that can be applied according to their applicability assumptions.**
- **Schedulability analysis of a real-time system architecture model by a software architecture designer:**
  1. He checks compliancy of his model with 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.

A “design pattern” approach to increase real-time scheduling usability

- **Specification of various design patterns:**
  - **Time-triggered :** time triggered architecture
  - **Ravenscar** : PCP shared data
  - **Black board :** readers/writers synchronization
  - **Queued buffer :** producer/consumer synchronization
  - **…**
  - **Compositions of design patterns.**

- **Expressed with AADL:**
  1. **Software components:**
     - **Thread :** flow of control that executes a program (e.g. Java or POSIX thread).
     - **Data :** any data structure in a program (e.g. C++ class).
     - **Process :** models a virtual address space containing threads and data.
  2. **Hardware components:**
     - **Processor, bus, memory unit :** parts of the execution environment.

Example : «time-triggered» design pattern

- **Design pattern definition :** threads are independent from a scheduling point of view as communications are made at predefined times (e.g. sending on completion time, receiving on release time).

- **Constraints defining this design pattern (modeling architecture and applicability assumptions) :**
  - Constraint 1 : all threads are periodic
  - Constraint 2 : threads start at the same time
  - Constraint 6 : thread communications only with data port connections
  - …

- **Simplest design pattern … but :**
  - 10 feasibility tests are available in Cheddar for this design pattern.
  - 64 cases depending on feasibility tests applicability assumptions (value of component properties).

    => Finding the right feasibility tests to compute is not so easy, even here.

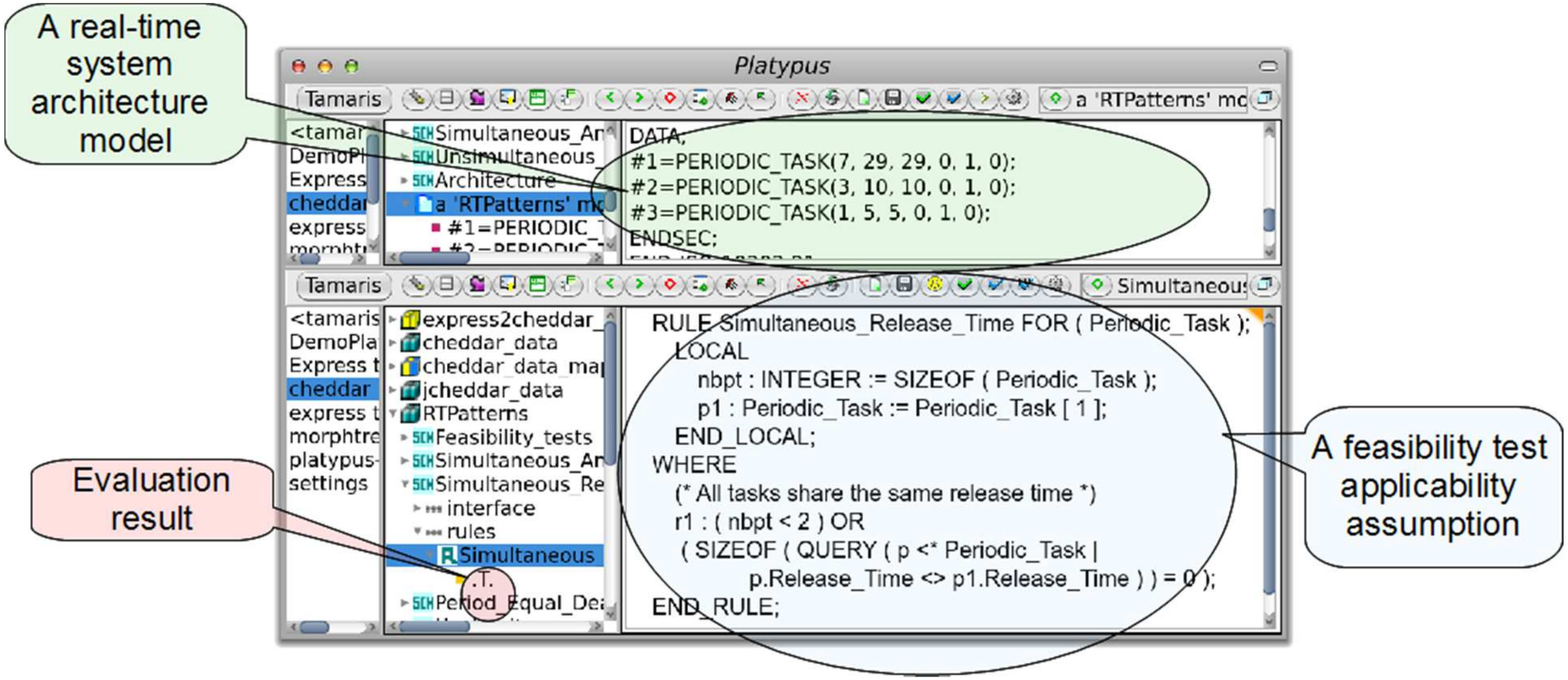
A "design pattern" approach to increase real-time scheduling usability

- **Define a set of architectural design patterns of real-time systems.**
  - = models a typical task communication or synchronization.
  - = set of constraints on entities/properties of the system

- **For each design pattern, define feasibility tests that can be applied according to their applicability assumptions.**

- **Schedulability analysis of a real-time system architecture model by a software architecture designer:**
  1. He checks compliancy of his model with 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.

Checking compliancy of a real-time system architecture model to a design pattern (2/2)



- **Top right part:** real-time system architecture model to verify.
- **Bottom right part:** modeling of a feasibility test applicability assumption.
- **Left part:** result of the model compliancy analysis.

Talk overview

1. Cheddar : context and motivations
2. What we investigate
3. A design pattern approach to enforce scheduling analysis
4. Conclusions and ongoing works

Conclusion and future works

- **Summary on Cheddar:**
  - Cheddar web site: http://beru.univ-brest.fr/~singhoff/cheddar
  - Investigate how to increase usability of real-time scheduling analysis.
  - Focus since 2008: tool interoperability and tool-chains integrating scheduling tools

- **Did we increased usability of real-time scheduling?**
  - Integrated into AADLInspector (Ellidiss Technologies), TASTE (ESA/Ellidiss Tech.),
  - Lectures/Labs in many universities.
  - Different research projects.
  - Too few industrial case studies and returns of experience.

- **Ongoing projects:**
  - Different projects around AADL : subset Annex, mutiprocessor/cache/memory AADL modeling and scheduling analysis
  - Multi-Frame scheduling & TDMA for Software Radio Protocol : Thalès Communications
  - SMART :comparison Cheddar & Marzhin : Ellidiss Tech & Virtualys
  - ARINC 653 support : Ellidiss Tech & U. Lisbon

A "design pattern" approach to increase real-time scheduling usability

- **Why choosing AADL:**
  1. Real-time features: typed components (threads, data, process …).
  2. Pivot language: international standard. Tools exist (Ocarina, OSATE).
  3. Compliant with real-time scheduling theory

- **An AADL model is a set of:**
  1. **Software components:**
     - **Thread :** flow of control that executes a program (e.g. Java or POSIX thread).
     - **Data :** any data structure in a program (e.g. C++ class).
     - **Process :** models a virtual address space containing threads and data.
  2. **Hardware components:**
     - **Processor, bus, memory unit :** parts of the execution environment.
  3. **System components :** model deployment of software on hardware components

Checking compliancy of a real-time system architecture model to a design pattern (1/2)

- **Compliancy of AADL models to design patterns can be checked:**
  - **Either by Platypus.**
  - **Or directly by Cheddar :** a part of Cheddar is generated by Platypus for such a purpose.

- **Platypus = Pharo + STEP:**
  - Pharo : open-source Smalltalk. Editing/handling of model/meta-model.
  - STEP : ISO 10303. Includes Express (data/constraint modeling language).
  - Open-source tool, available at: http://cassoulet.univ-brest.fr/mme.

- **Meta-models handled by Platypus in order to build the compliancy tool inside Cheddar:**
  - A model for each design pattern which includes rules encoding constraints of the design pattern.
  - A model for each feasibility test which includes rules encoding its applicability assumptions.