About architecture description languages and scheduling analysis

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

***Ellidiss Technologies, France**

+Lab-STICC/UMR6285, University of Brest/UBO, UEB, France

Talk overview

1. Real-time scheduling theory
2. About usability of real-time scheduling theory
3. A design-pattern approach to increase usability of real-time scheduling theory
4. Conclusion

Real-time scheduling theory (1/2)

- **Real-time systems:**
  1. Functions of real-time systems have timing constraints to meet.
  2. Deadlines
  3. **How to check deadlines at design time ?**

- **Timing constraints analysis with real-time scheduling theory (sometimes called “Rate Monotonic Analysis”):**
  1. **Modeling functions:** simplified models of task = processor demand + deadline (e.g. periodic task model).
  2. **Use of standard scheduling algorithms** and protocols (e.g. Fixed priority scheduling, PCP).
  3. **Verification:** feasibility tests (or schedulability tests).

Real-time scheduling theory (2/2)

- **Example of a feasibility/schedulability test**: worst case response time of tasks (Joseph & Pandia 1986):

$$R_i = C_i + \sum_{j \in hp(i)} \left\lceil \frac{R_i}{P_j} \right\rceil \cdot C_j \leq deadline_i$$

- **Applicability assumptions:**
  - Periodic tasks, scheduled by a preemptive fixed priority scheduler.
  - Deadlines are equal to periods.
  - Tasks are independent.
  - We have a critical instant : all tasks start at the same time (same release time).
  - **...**

Talk overview

1. Real-time scheduling theory
2. About usability of real-time scheduling theory
3. A design-pattern approach to increase usability of real-time scheduling theory
4. Conclusion

About usability of real-time scheduling theory (1/2)

- **About feasibility/schedulability tests in the mono-processor case:**
  - Seems to be a simple method.
  - Compliant with operating systems features (POSIX 1003 standard).
  - Standalone verification tools exist: Rapid-RMA, MAST, SymTA/S, Cheddar, …
  - Strong demand from designers on this analysis method.

- **But few people/project actually perform analysis with tools implementing real-time scheduling theory.**

- **Many possible explanations:**
  1. Not suitable on some architecture types (e.g. specific multi-core/distributed/hierarchical systems).
  2. Difficult to use by architecture designers.
  3. …

About usability of real-time scheduling theory (2/2)

2. Difficult to use by architecture designers:

- **Numerous feasibility tests and applicability assumptions:**
  How to chose the feasibility test to apply? How a designer can be sure that his architecture model is compliant with a feasibility test?

- **Automatic verification : usually limited interoperability level between model editors and verification tools:**
  - Need Architecture Description/modeling Languages that :
    - Are compliant with real-time scheduling theory : provide data required by this type of analysis.
    - Need a common/accurate semantic
    - Are pivot language

Talk overview

1. Real-time scheduling theory
2. About usability of real-time scheduling theory
3. A design-pattern approach to increase usability of real-time scheduling theory
4. Conclusion

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

- **Architecture Analysis and Design Language (AADL) :**
  - An AADL model is a set of components, connections, properties.

- **Why AADL:**
  1. Real-time features: thread, processor components, ...
  2. Compliant with real-time scheduling theory : component properties (deadlines, periods, …).
  3. Pivot language: international standard. Many tools exist.

- **But AADL is a very rich language :** several features for thread communication/synchronization.
  - How a designer can be sure that his architecture model is conforming to a feasibility test when mixing several types of thread connection?
  - What is allowed? What is forbidden?

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

- **Define a set of architectural design patterns of real-time systems.**
  - Models a typical thread communication or synchronization.
  - Set of constraints on entities/properties of the architecture model.
  - Ex: Ravenscar, Time-triggered, specific to companies, ...
- **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 an architecture designer:**
  1. He checks compliance 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.

The «time-triggered» design pattern (1/2)

- **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 AADL threads are periodic
  - Constraint 2 : threads start at the same time
  - Constraint 6 : thread communications only with data port connections
  - …

- **Criterion to compute:** worst case thread response time.

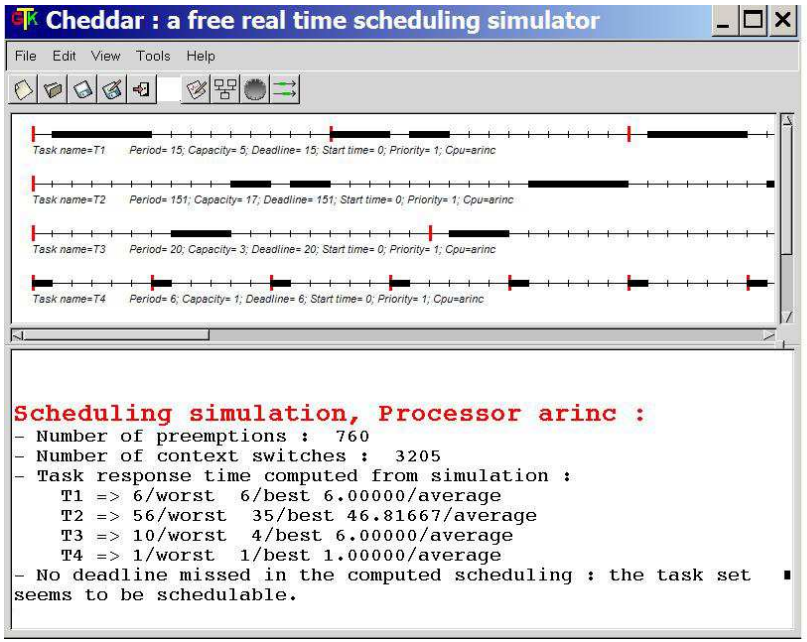
- **Simplest design pattern … but :**
  - 10 feasibility tests are available in Cheddar for this design pattern.
  - 64 cases depending on feasibility tests applicability assumptions (value of component properties).

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

The «time-triggered» design pattern (2/2)

- **Two steps scheduling analysis of an AADL model:**
  1. Check compliance of the AADL model with the «time-triggered» design pattern.
  2. Computes corresponding schedulability tests. **Cheddar :** implements various feasibility tests.

Analysis from feasibility tests (worst case response times)



Checking compliance of a real-time system architecture model to a design pattern (1/3)

- **Compliance tool automatically produced by a model driven engineering tool.**

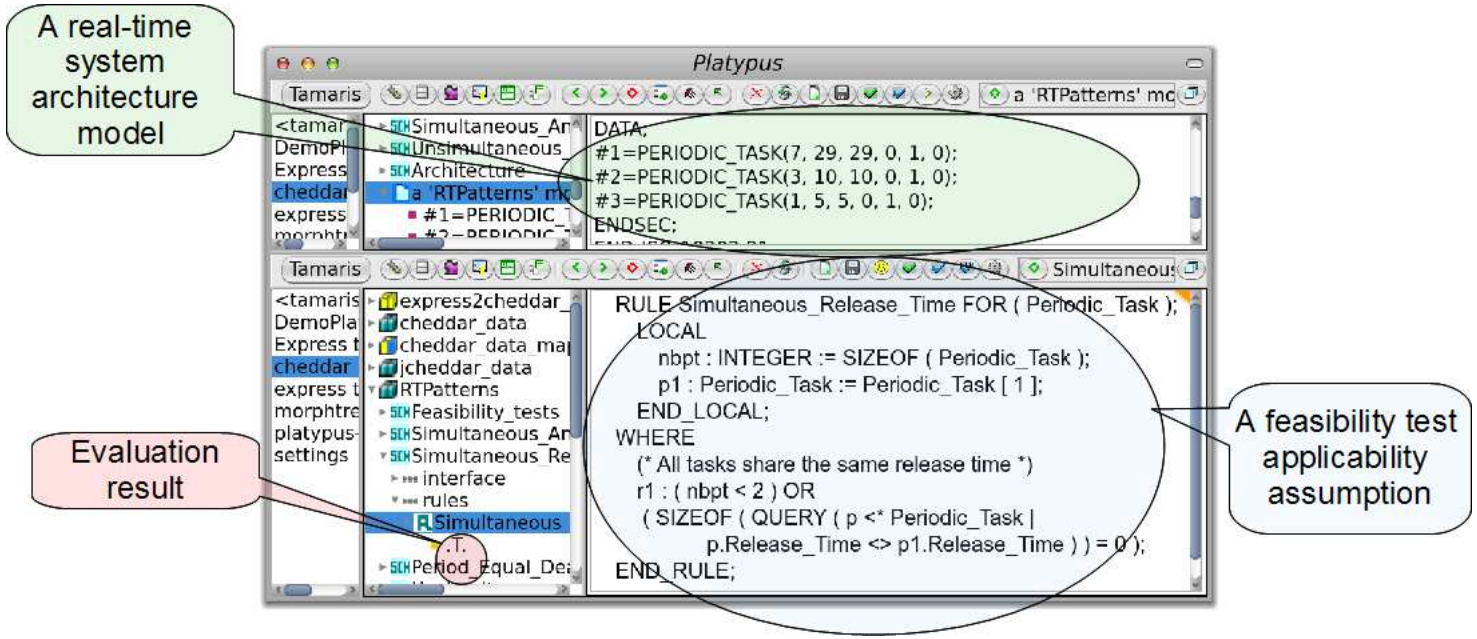
- **Platypus :**
  - Implementation of STEP.
  - Includes EXPRESS: data and constraint modeling language.

- **Models/Meta-models handled by Platypus in order to build the compliance tool:**
  1. Models for design patterns which include constraints modeling each design pattern.
  2. Models for feasibility tests which include constraints modeling applicability assumptions of each feasibility test.
  3. Models for the relationships between (1) and (2).

- **Constraints can be checked by Platypus or by a software generated by Platypus.**

Checking compliance of a real-time system architecture model to a design pattern (2/3)

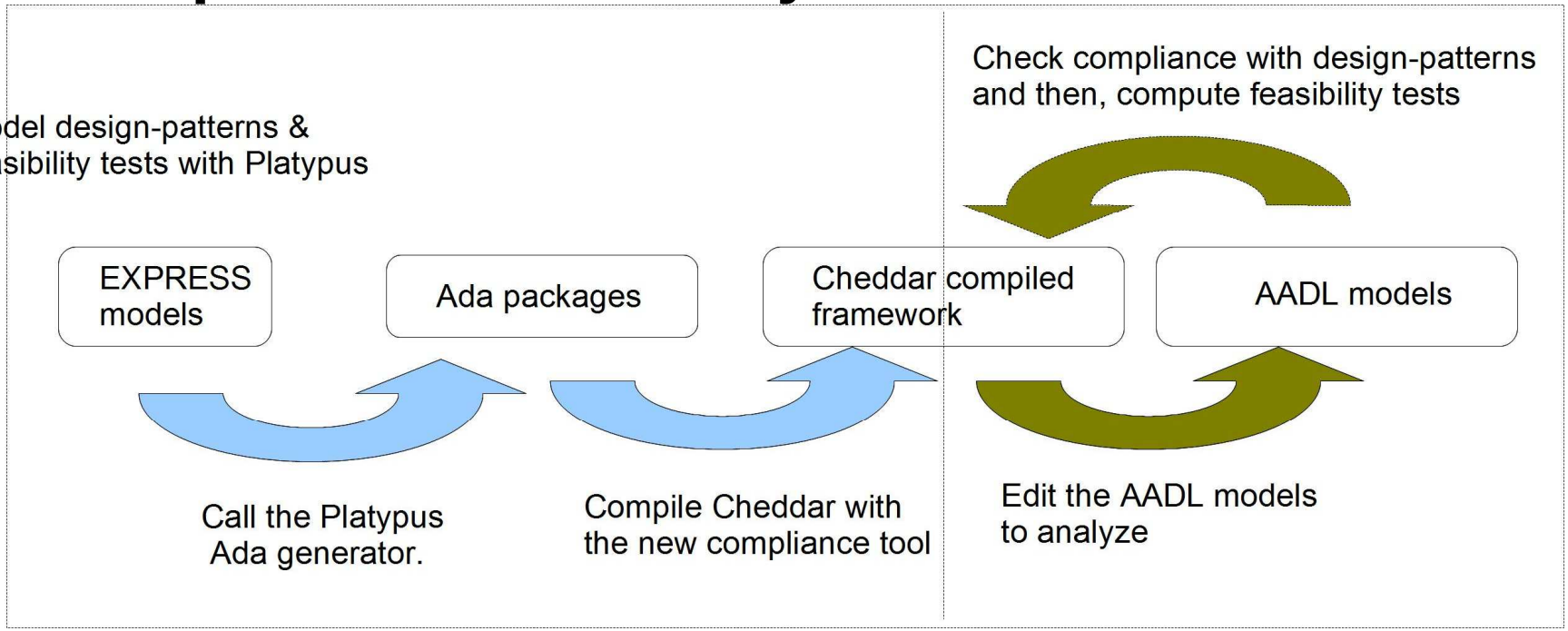
- **Compliance checked by Platypus:**



- **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 compliance analysis.

Checking compliance of a real-time system architecture model to a design pattern (3/3)

❑ Compliance checked by Cheddar:



- **Blue:** to produce the compliance tool as part of Cheddar.
- **Green:** architecture (AADL) analysis with Cheddar.

Conclusion

- **Summary :** real-time scheduling analysis tools are difficult to apply.
  - Define design patterns and assign feasibility tests to them: What is mandatory? What is forbidden?
  - 2 steps analysis : design-pattern compliance checking, and then feasibility/schedulability analysis.

- **Preliminary results:**
  - We can automatically produce compliance tool.
  - Compliance tool has a reasonable response time: allows verification during model editing.

- **Composition of design patterns?**
  - Some architecture models are composed of several design patterns.
  - How to check compliance and schedulability analysis?

- **Raised issues:** we use AADL subsets. Kind of semantic documentation. What is the suitable ADL? Where should we attach semantic?