Automatically adapt Cheddar to users need

AADL Standards Meeting, Toulouse

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

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

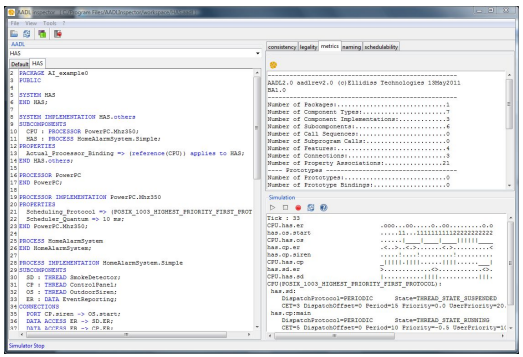
October 18, 2011

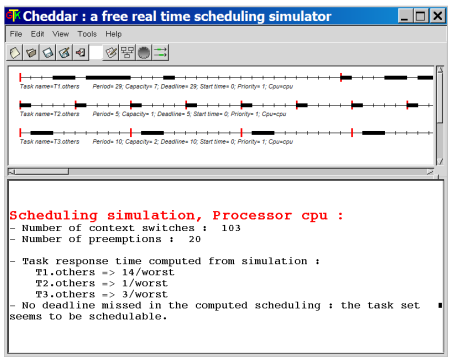
Outline

The Cheddar project

Simplify the use of the Real-Time scheduling theory.

- A pragmatic approach
  - Since 2008, partnership with Ellidiss Tech. for open source and industrial support
- A tooled approach: the Cheddar tool
  - Freely available as a standalone tool
  - Can be run with STOOD or with AADLInspector (Ellidiss technology support).





The Cheddar project

A community

- Academic research and educational.
- Industrial utilizations ?

The Cheddar tool is continuously evolving.

Two challenges

- Make Cheddar reusable and adaptable to user's specific requirements.
- Allow Cheddar users to adapt Cheddar themselves.

The Cheddar tool

Engines for real-time architecture evaluation:

- Analysis of AADL models with feasibility tests.
- Exhaustive simulations
  - with classical schedulers: *Rate Monotonic*, *Earliest Deadline First*.
  - with specific schedulers written into the Cheddar language.

"Design pattern" approach

Define a set of AADL architecture design patterns for real-time systems.

- Models a typical thread communication/synchronization.
- Set of constraints on AADL components/properties.

For each design pattern, all feasibility tests that can be applied according to their applicability assumptions are explicitly declared.

Verification of a real-time system architecture model

- The designer uses a tool that detects which design pattern his architecture is compliant with.
- The designer performs feasibility tests.

Validation with exhaustive simulations

Consists in verifying timing constraints on an AADL architecture. A Scheduling simulation runs during hyper-period.
Two possibilities:

- Use one of the built-in schedulers.
- Provide and use your own scheduler programmed into the Cheddar language.

The Platypus tool

Platypus is a general purpose workbench for the building of target systems with the help of code generators. It allows:

- The specification, the verification and the validation of meta-models.
- The implementation of code generators.

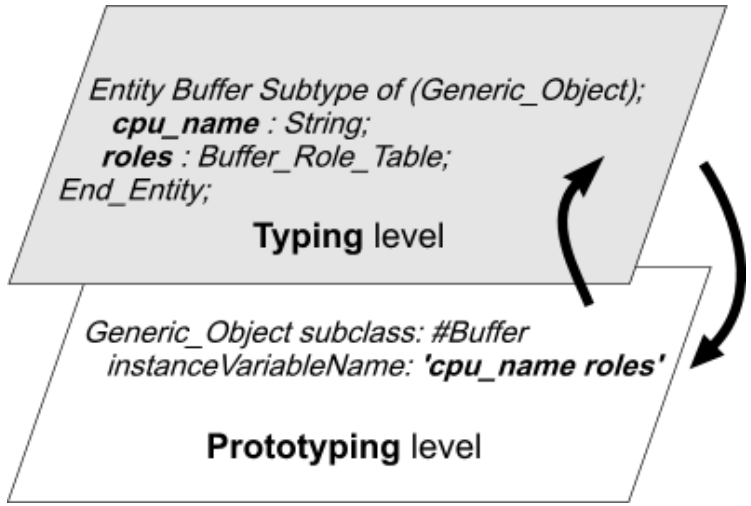
Main idea behind Platypus

Make it possible the very early validation of meta-models.

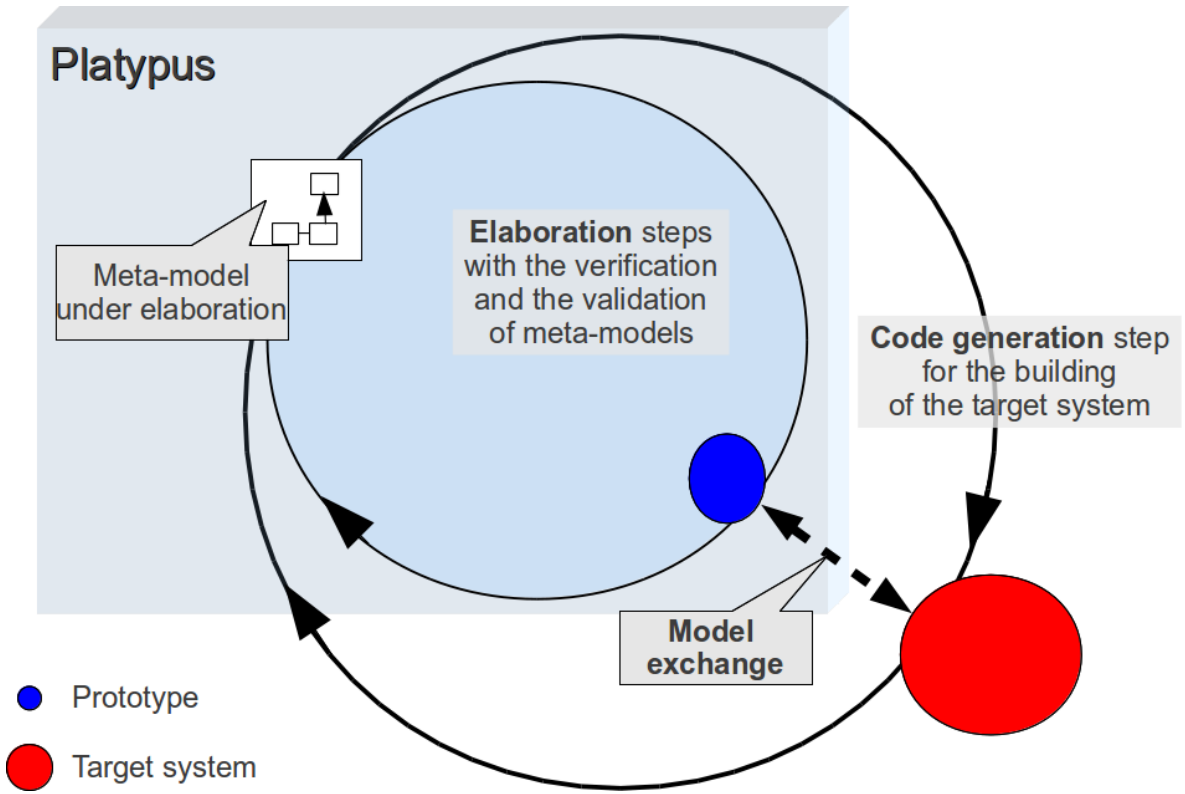
The Platypus tool

- Platypus is made of an EXPRESS modeling language workbench implemented into Smalltalk.
- A meta model is represented at two levels
  - A set of Smalltalk classes (high level, programmability)
  - A set of EXPRESS entities (Static types, constraints)

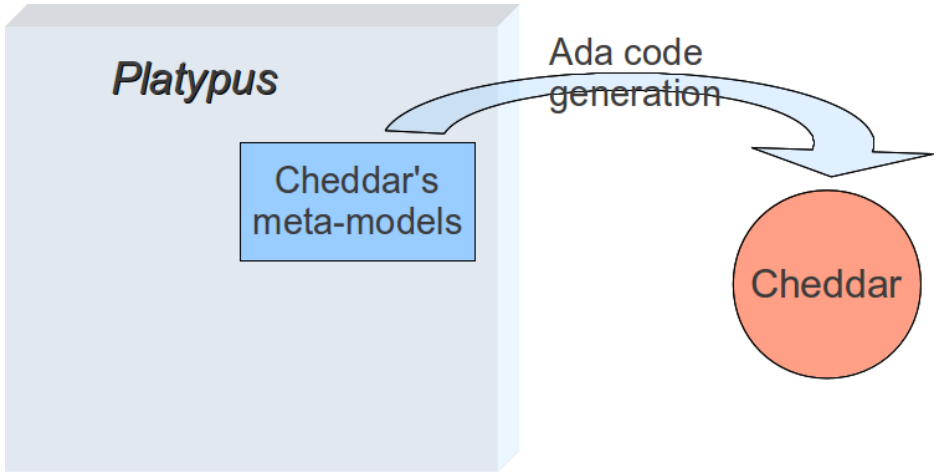
Code generators: either into EXPRESS or into Smalltalk.



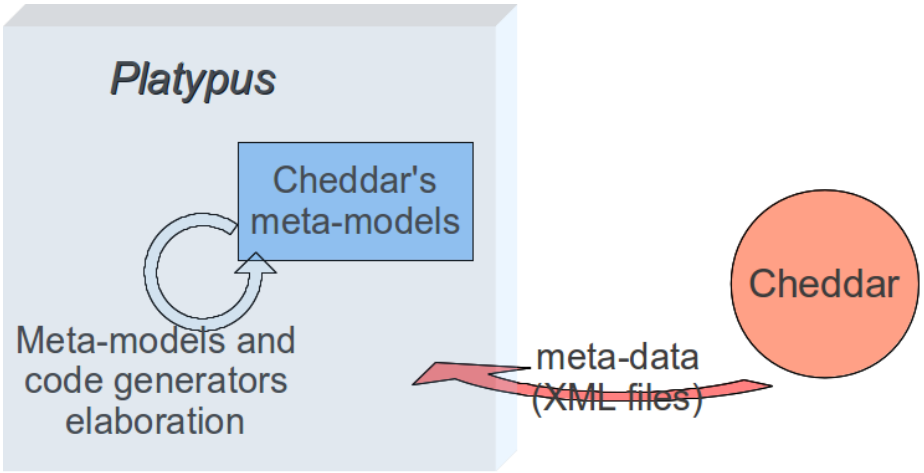
An iterative design process



Two iteration steps



1. Code generation step.
2. Meta-models en code generators elaboration steps.



Evolution of the Cheddar tool

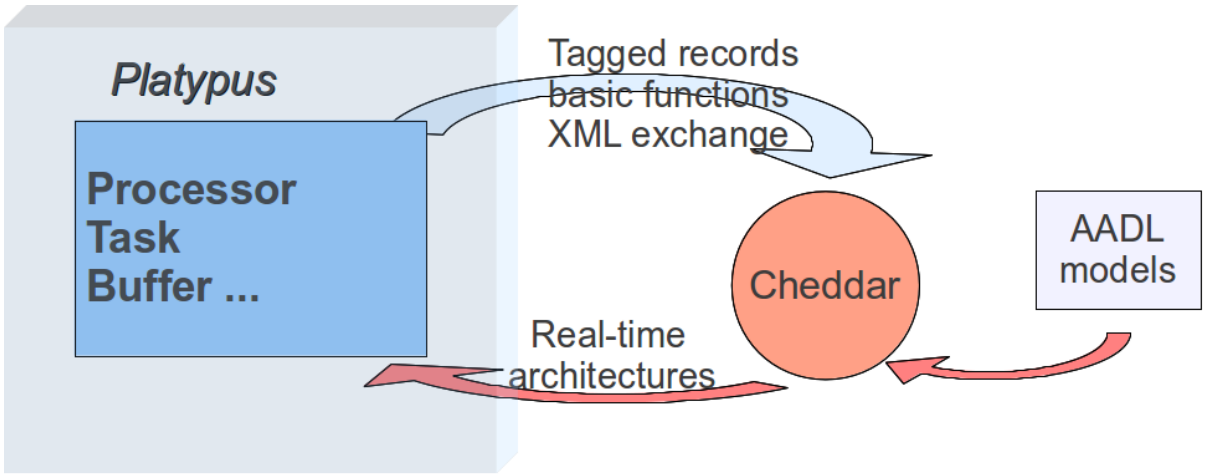
Adaptations must be possible at all levels:

- **Architecture meta-models**
  - Very specific needs that imply very fine adaptations of the Cheddar engines.
- **Cheddar engines**
  - Specific schedulers.
  - New patterns for the selection of feasibility tests.

Evolution of the architecture meta-models

These meta-models constitute the core of Cheddar:

- **An evolution requires manual Ada programming** to integrate the generated code.
- Example: adding of multi-core/cache management.

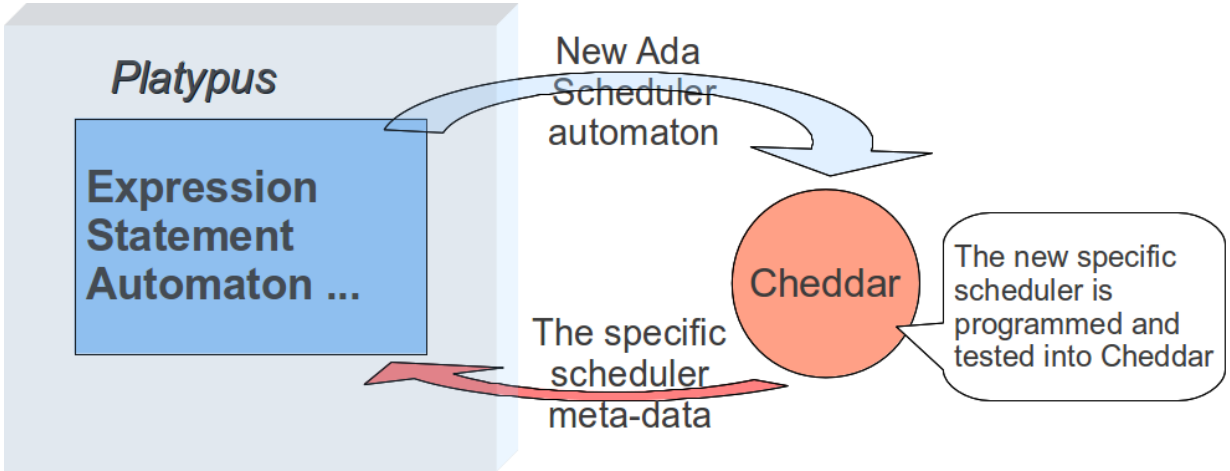


- Validation with realistic AADL architectures.
- Example: enforce domain constraints.

Adding of a specific scheduler

The new scheduler is first implemented into the Cheddar language.

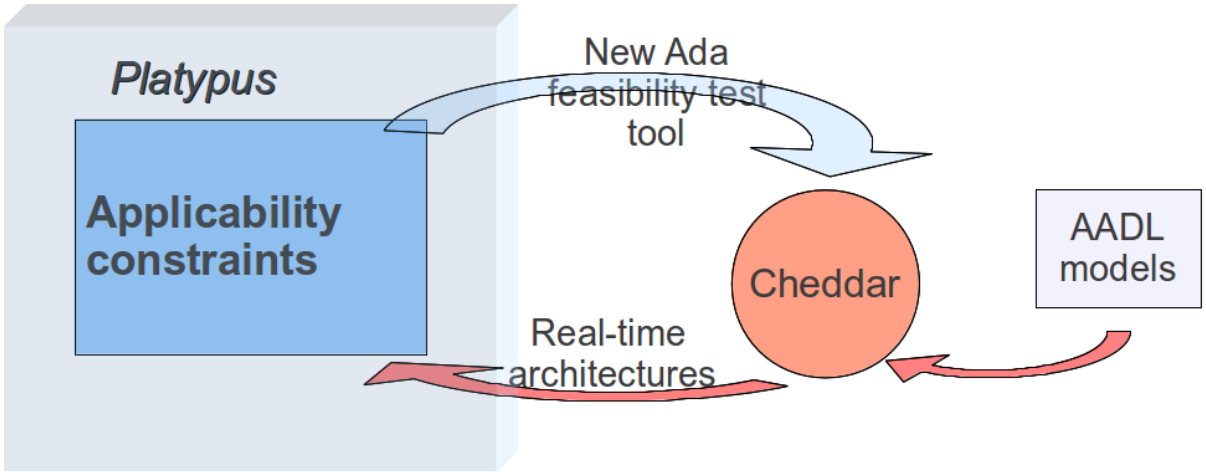
- **It is first tested with the Cheddar interpreter then, the corresponding Ada code can be generated and integrated into Cheddar.**
- The result is a specific version of Cheddar.



Adding a new pattern for feasibility tests

The new pattern is first specified and tested into Platypus.

- Applicability assumptions are designed as additional rules that constraint further the generic layer.
- Rules are translated to Ada, the result is a new version of Cheddar.



Current status

- 30% of the Cheddar Ada code is automatically generated (16500 lines).
- This code is generated from the Architecture and the Cheddar language meta-models (1650 lines).
- We have an example (a running prototype) for the *Synchronous Data Flow* and *Ravenscar* patterns. So, we are ready to build a first version of the generator for the design pattern tool.

	Meta-models	Code generators
Architecture meta-models		
Specific schedulers		
Feasibility test patterns	Current research	

works not finished

Possible developments

- The elaboration capabilities of Platypus are not used enough
  - Some structural issues remain
  - No constraint in the Architecture and Cheddar language meta-models.
- Other part of Cheddar could be generated (User interface, architecture and user input checking...)