Overview of the Ocarina toolset

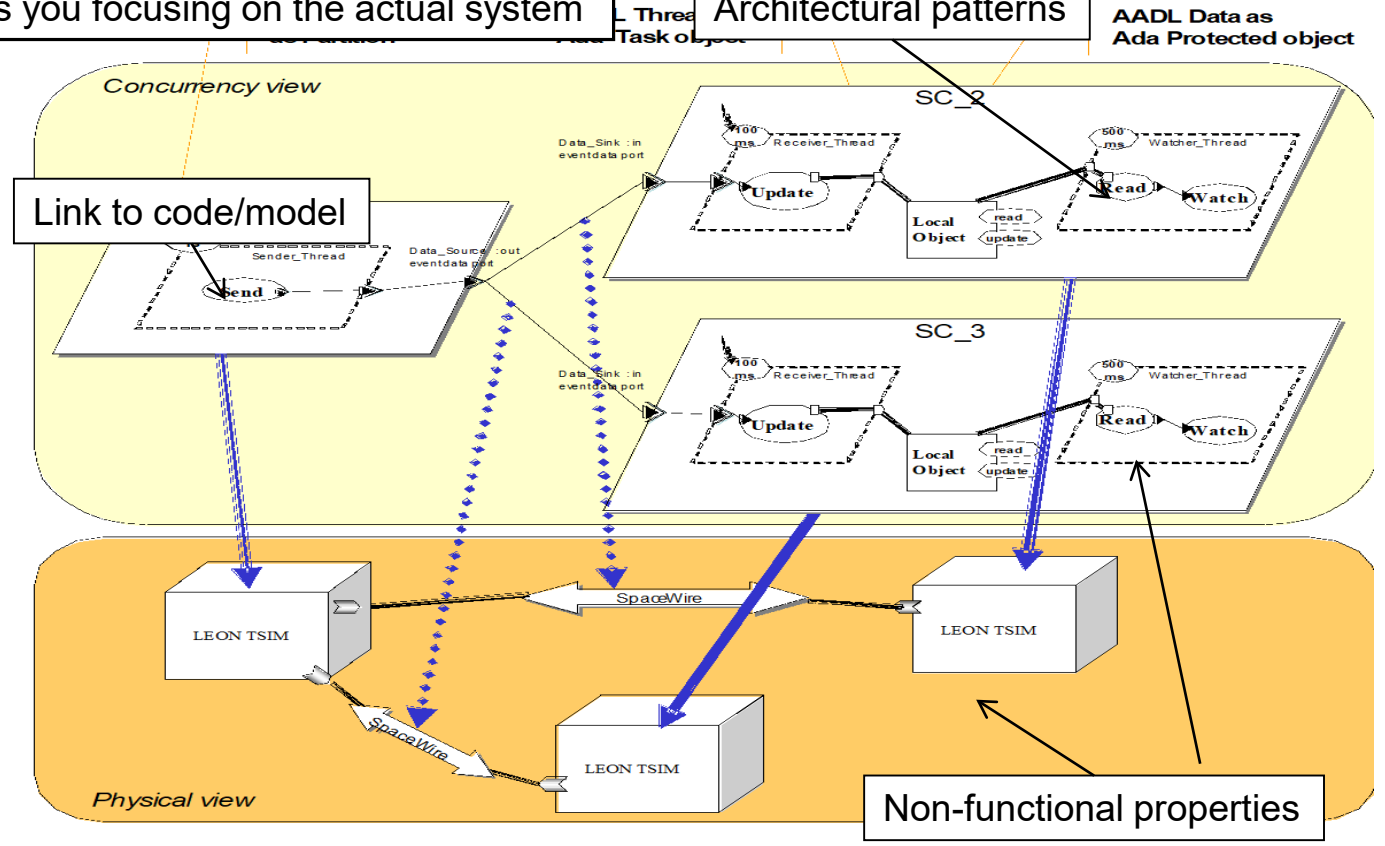
Jérôme Hugues – ISAE-SUPAERO

MBS2E/AADL

http://openaadl.org

Architecture helps you focusing on the actual system

Architectural patterns



Link to code/model

Non-functional properties

AADL covers many parts of the V cycle: model checking, scheduling, safety and reliability (ARP4754) and code generation

J. Hugues contributions to SAE AADL since 2006

Lead of the Ocarina toolset

Middleware synthesis :
Ada, C (POSIX, ARINC653),
TRL 5 with ESA (ECSS E-40)

Scheduling: Cheddar, MAST
TRL 4 with ESA

Model checking: Petri Nets (Time and CPN), LNT
TRL 2 (PhD contribution)

Link with SysML & Capella
TRL 1, under evaluation

4 books on AADL, tutorials available

Focus: AADL and code generation

MBSE paradigm shift: **model == code**

AADL has a full execution semantics

- Allow for full analysis
  - Scheduling, security, error, behavior
- But also ***generate system***

AADL standards combined ::=

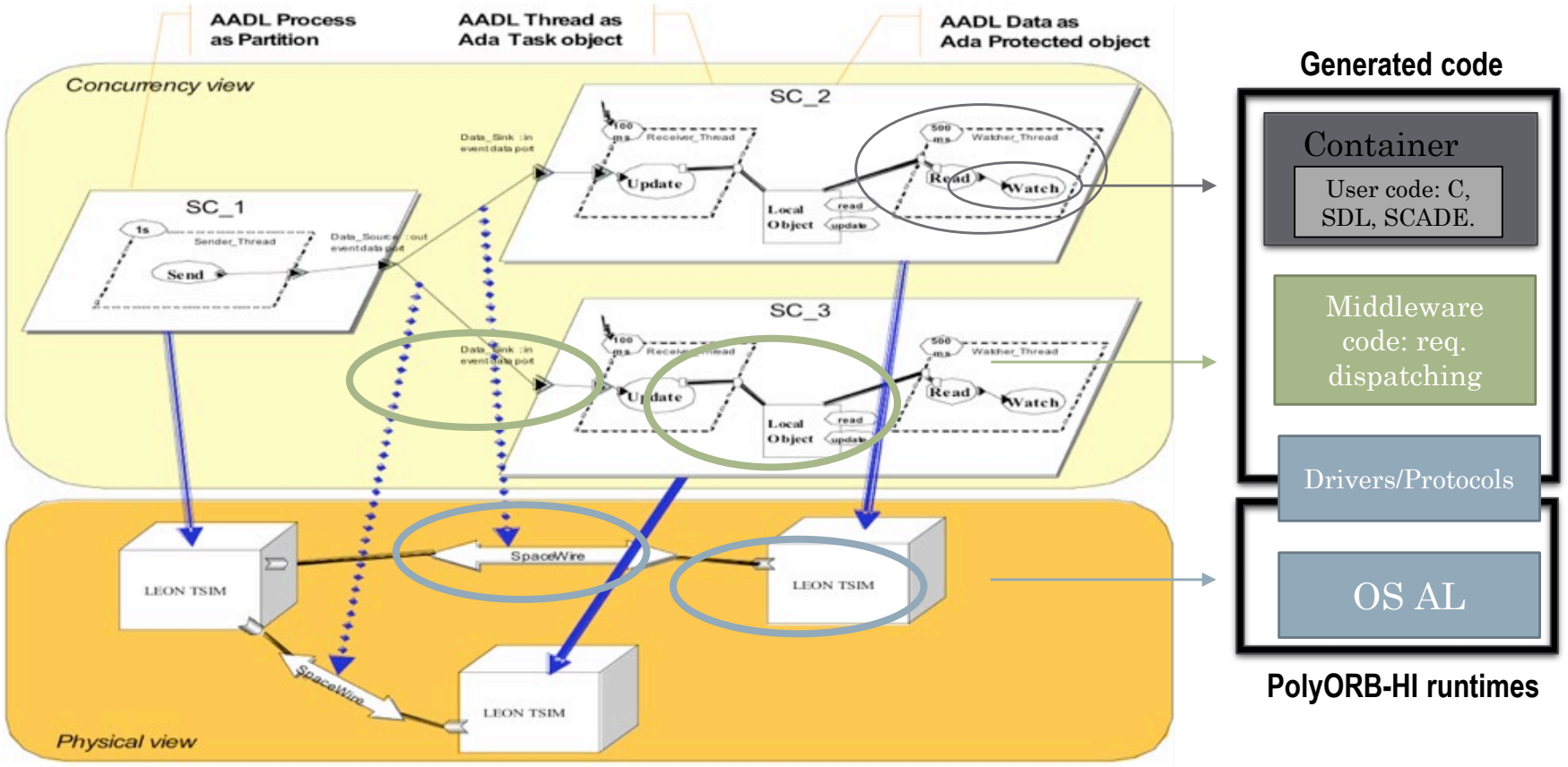
AADL + data modeling + code generation

→ implementation code

Code generation and middleware using Ocarina
Architecture-Centric *Implementation-Driven* Integration Process

AADL patterns for ***implementable*** systems: tasks, queues, buffers, protocols

Instantiate middleware using PolyORB-HI runtime libraries



Ocarina overview

http://openaadl.org

Both a library and a command line tool per initial requirement, , since 2006 (!)

Integrated to third-party tools: Osate, AADLInspector, TASTE (ESA)

Main feature: code generation

- **Ada HI integrity profiles**, with Ada native and bare board runtimes
- **C RTOS**: RT-POSIX, RTEMS, Xenomai, for RTOS & Embedded
- **ARINC653 :** DeOS (DDC-I), VxWorks 653 (Wind river)

User code can be Ada, C, C++, Esterel, Simulink , Lustre, SCADE

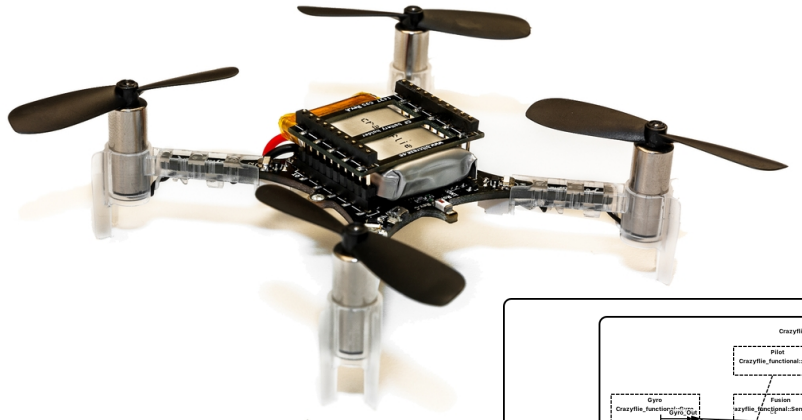
- Support through their respective code generator

Automatic evaluation of code coverage running scenarios

Link to scheduling analysis tools: Cheddar, MAST

Model checking using LNT, Petri Nets

AADL to Ada code



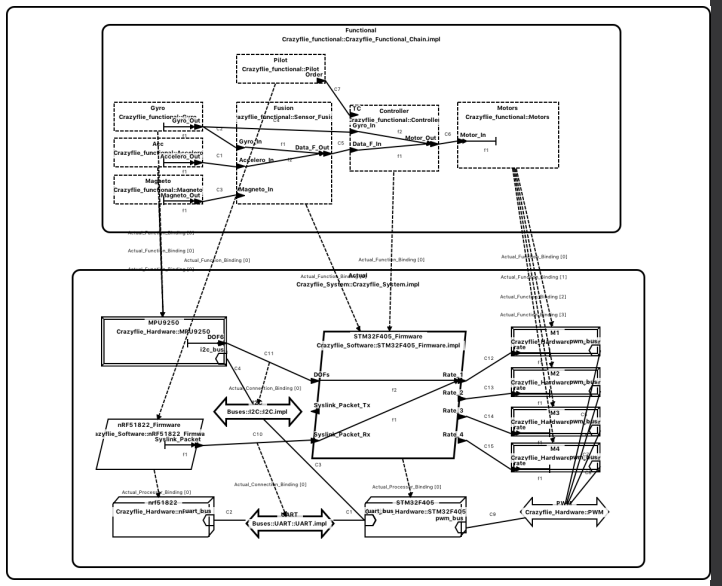
Target Ada Ravenscar and High-Integrity runtimes

Based on the Ravenscar & HI Ada profiles

- Meets stringent requirements for High-Integrity systems, e.g. ESA
- Checked at compile-time by Ada compiler, GNAT

Completion of adaptations towards SPARK2014 underway

- “Gold Level” (absence of runt-time erros + proof of key properties)



Example project with ISAE

- Model of Crazyflie UAV
- Code generation for SPARK
- Proof of Simulink Controller

AADL to C code

C target part of TASTE ESA requirement for space systems

- Follow mission-critical guidelines
  - No memory allocation: static resources, threads, etc.

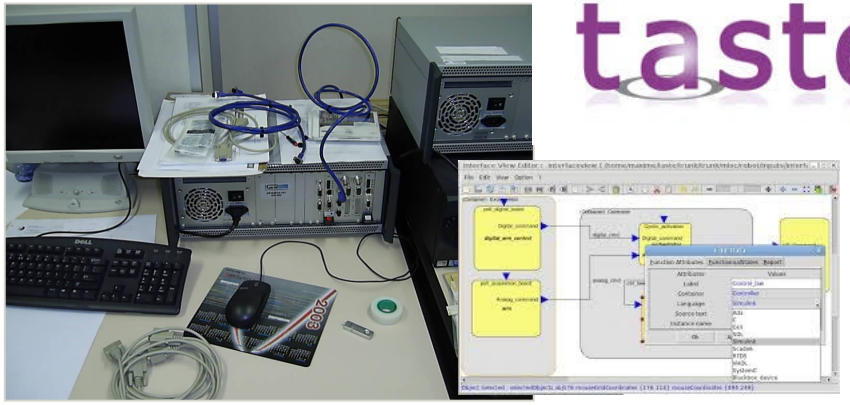
Multiple OS supported

- Supported: RT-POSIX, C/RTEMS, Xenomai, FreeRTOS, VxWorks,

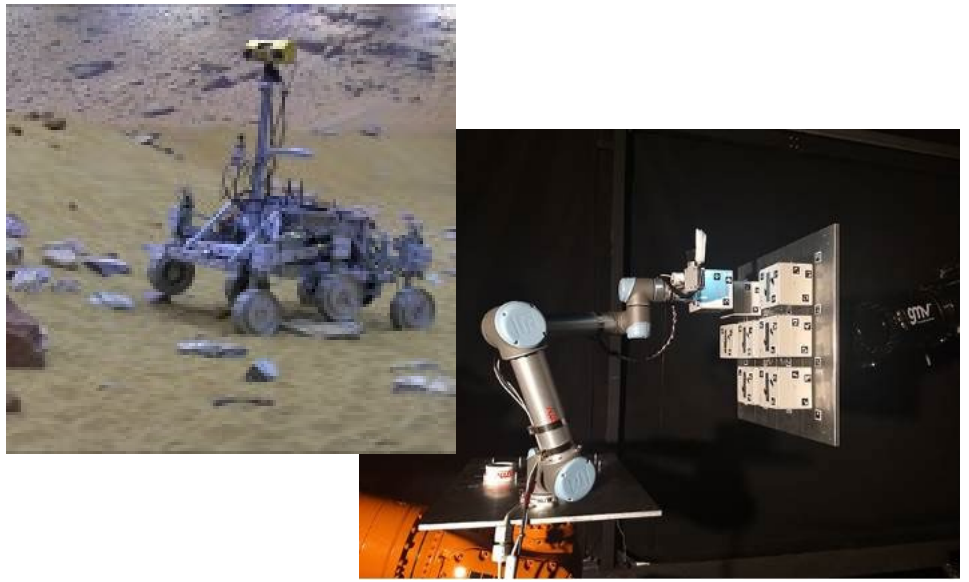
Tested on different configurations

- POSIX RTOS: Linux, RTEMS, eLinOS (Linux)
- Specific APIs: RTEMS, Xenomai, VxWorks 6.2

Cost for qualification material evaluated to 1 man-year



TASTE toolchain with ESA (2008)



H2020 PERASPERA SRC1 (2018)
Code generation for space robotics

AADL to ARINC653

Rely on ARINC653 API instead of runtime

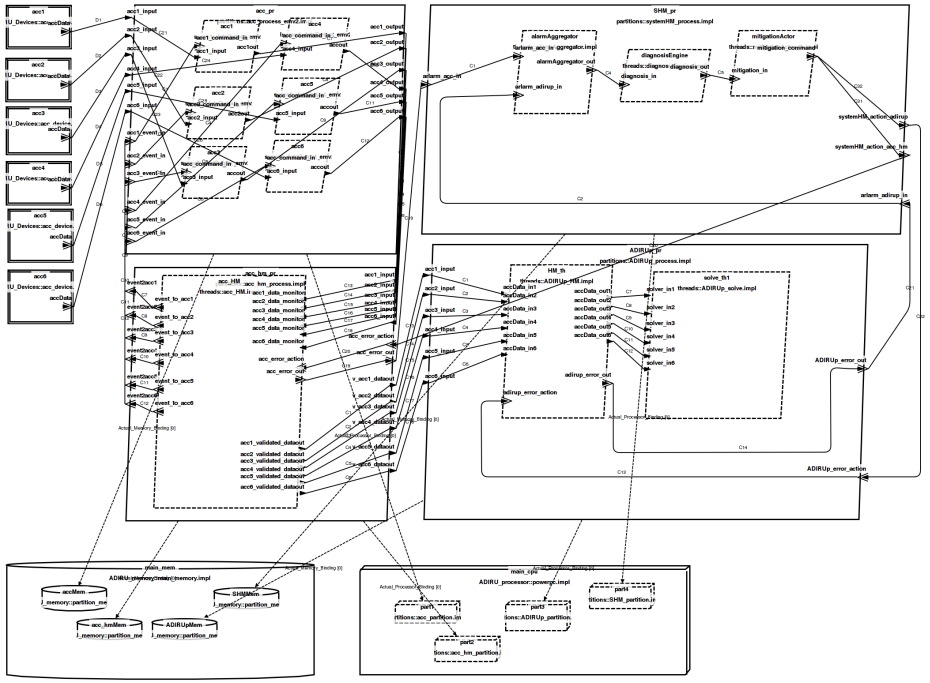
Leaner approach

Tested with

- DDC-I DeOS, WRS VxWorks653

Combined with OSATE analysis capabilities

- Resolute checks for ARINC653 modeling patterns
- EMV2 safety analysis
- Code generation using Ocarina



Joint project with SEI (2015)
Unified workflow
Safety analysis + Code generation

See http://aadl.info/aadl/demo-arinc653/

Other contributions:

- SysML/AADL coupling
- Simulation with FMI
- Scheduling analysis
- Formal methods

Books and tutorials on openaadl..org

Thanks !

(just a small overview, see you this afternoon)