TELECOM PARIS
école nationale supérieure des télécommunications

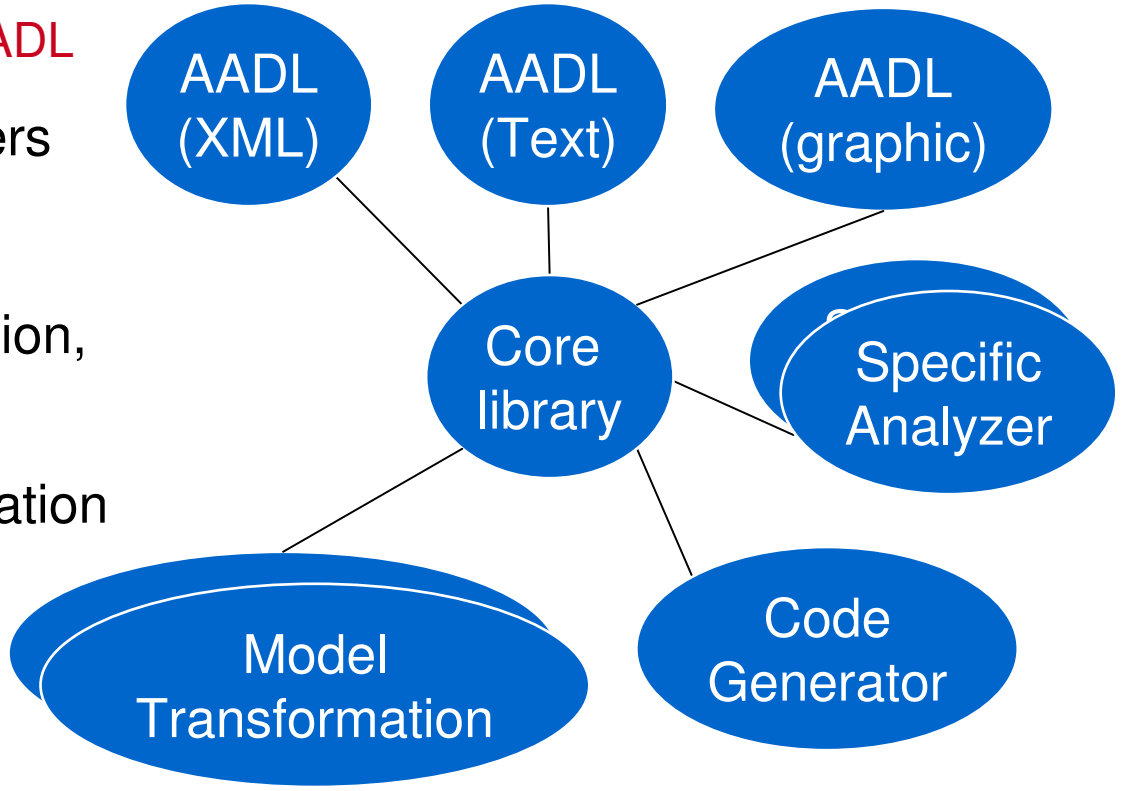
Ocarina, an AADL-to-X generator: status & work in progress

Jérôme Hugues, ENST

get

Ocarina Tool Suite

- Library to manipulate AADL
  - ⇨ Parsers and printers
  - ⇨ Semantic checks
  - ⇨ Model transformation, code generation
  - ⇨ Run-time configuration
- Ocarina 1.0
  - ⇨ Code generator
    - Ada/PolyORB
    - Ada/PolyORB-HI
  - ⇨ V&V
    - Schedulability (Cheddar)
    - Petri Nets



Petri Nets

Ada, C

AADL

Run-Time

HI-middleware for AADL: PolyORB-HI

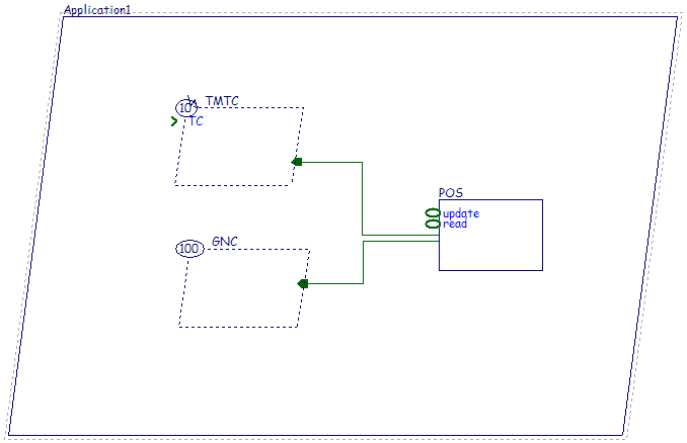
- AADL defines a semantics for a runtime
  - ⇨ Possible to derive code from the architecture
  - ⇨ Containers completed with application-specific code
- Hard-Real Time systems specific constraints
  - ⇨ Ravenscar profile + Ada High-Integrity restrictions
  - ⇨ No allocator (task, memory), no dispatching
- Exploit AADL properties and patterns to generate code
  - ⇨ Runtime entities correspond to some AADL patterns
  - ⇨ Configured through AADL properties (priority, stack, protocol, …)
- Code generation strategies
  - ⇨ Abiding to HI restrictions, Ada compilation checks, etc.

Ocarina features

- Combine AADL, compilers, model checkers, validation tools
  - ⇨ Automate as much as possible the analysis of a system
- Fully automatized process from AADL models down to executable code, using Ocarina and 3rd party tools like Cheddar and GNAT
- Currently: target Ada, import of C code using either socket or SpaceWire to communicate
- Future work: inclusion of SCADE, generation of C code (RTEMS, POSIX runtimes), increase support for AADL 2

Building application from AADL & code

- Application sample (control system)
  - ⇨ Periodic, aperiodic, shared variable
- Refinement from high-level view to deployment
  - ⇨ Model + function code
- Down to code generation
  - ⇨ System is schedulable
  - ⇨ Has no deadlock
  - ⇨ Can be generated for PolyORB-HI
  - ⇨ Fully respects all compile-time restrictions
  - ⇨ Then runs on **tsim** (LEON2 simulator)
- Ready for certification & deployment
  - ⇨ System comes with models and code



+

```
system implementation toy_example.sample_1
subcomponents
  P1   : processor the_processor;
  P2   : processor the_processor;
  GNC  : process GNC_Proc;
  TMTC : process TMTC_Proc;
properties
  Actual_Processor_Binding
                => reference P1 applies to GNC;
  Actual_Processor_Binding
                => reference P2 applies to TMTC;
end toy_example.sample_1;
```

What is available from ENST

- http://ocarina.enst.fr/
  - Binaries of Ocarina (release 1.0 and nightly builds)
  - Documentation and examples
  - Scientific papers on the use of AADL
- http://aadl.enst.fr/polyorb-hi
  - Case studies using Ocarina & PolyORB-HI
  - Binaries for GNU/Linux, ERC32, LEON
  - AADL models

Conclusion and Ongoing Work

- AADL proved it is interesting for our partners to build and generate code
  - ⇨ IST-ASSERT, AdaCore, Thalès, SAGEM, MBDA
- Ocarina enables the construction of applications from AADL models
  - ⇨ By incorporating functional code in the architecture
  - ⇨ By generating “glue code” to execute it
  - ⇨ By ensuring code quality
- Some case studies are available, need more
  - ⇨ Do not hesitate to send us models !

Feedback required to speed up the process and ensure its scalability !