Generation of applications from AADL models using Ocarina
Involved in AADL since 2004

Used AADL as part of our research activities on middleware and Distributed Real-Time and Embedded systems

Key idea: use AADL to configure and deploy applications

- Use a compiler approach to generate support code for distribution, concurrency, buffer allocation and checks

Open source projects, GMGPL or BSD licenses:

- Ocarina: toolbox for AADL
- PolyORB-HI: C/RT-POSIX and Ada2005 runtimes for AADL
- POK: Partitioned OS and runtime for AADL

Partners: AADL committee, ESA, Thales, Ellidiss, UBO, LIP6

- Ocarina is a stand-alone tool for processing AADL models
- Fully supports both AADLv1 and AADLv2
- Prototype support for the Behavioral Annex (up to 2.9)
- Code generation facilities target AADL runtimes
  - Ada HI integrity profiles, with Ada native and bare board runtimes
  - C POSIX or RTEMS, for RTOS & Embedded
  - C/ARINC653 and partitioned kernel POK
  - User code can be Ada, C, C++, Esterel, Simulink, Lustre, SCADE
- Model to model transformations
  - WCET analysis of AADL runtime + user code: Bound-T for LEON
  - Model checking specifications using Colored or Timed Petri Nets
  - Constraint language to validate AADL model
Ocarina proposes an API to build your own AADL tools

- Like Ocarina itself, but also Cheddar (UBO), LabASSERT (ESA)
- Parsers, printers, semantic checks, model transformation

Add-ons

- Emacs and vim modes
- Eclipse plug-in for integration with OSATE
Ocarina distributions


- Ocarina 2.0 wavefront, daily snapshots
  - Binaries of Ocarina (release 1.2 and nightly builds)
    - For GNU/Linux, Windows, Solaris, Mac OS X, FreeBSD
  - Documentation and examples (30+ available)
  - Scientific papers on the use of AADL
  - Teaching materials for Master degree

- PolyORB-HI AADL runtimes
  - Two versions: Ada 2005 and C/RT-POSIX

- POK runtime
  - For IMA-like systems, using time and space partitioning
Ocarina examples

  - Examples from CMU/SEI, ASSERT, internal
  - For Linux, LEON and ERC32 platforms
  - Can be compiled for other native platforms

- Follow AADL data modeling, code generation, ARINC annexes
  - Actually, we write them to support code generation ;)

- A set of educational material is available
  - Build your own lab session using AADL
    - Then perform schedulability analysis using Cheddar, code generation, test
  - For master degree, or in-house tutorials
Building process for HI-DRE systems

User components

Generated application components

Integration

Final application

Configured middleware components

AADL Model

Architecture

Configuration

Real-time

Deployment

Analysis

Generated middleware components

Code Generation

Minimal middleware components

Deployment

Required middleware components
AADL to Ada
Experiments in IST-ASSERT
Ocarina’s AADL runtime #1: Ada

- **PolyORB-HI/Ada**
  - Target Ada Ravenscar and High-Integrity runtimes
  - Supports AADL semantics, v1 and v2
    - Need more tests to validate corner cases and extended use of AADL
  - 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
  - Supports native, LEON2, ERC32 targets

- **Validated in the context of IST-ASSERT with ESA**
The ASSERT MPC V2 demonstrator (2007)

AADL Process as Partition
AADL Thread as Ada Task object
AADL Data as Ada Protected object

Concurrency view

Data_Source: out event data port
Data_Sink: in event data port

Receiver
Local Object
update
500 ms
100 ms
Update
Read
Watch
Receiver_Thread Watcher_Thread
Sender_Thread

SC_1

SpaceWire
LEON TSIM

Physical view

< 1MB/node, Including RTOS And drivers (60%)

AADL Workshop – 2009/06/12
Generation of applications from AADL models using Ocarina
The ASSERT ESA demonstrator (2008)

- Stood + Ocarina + ASN.1 tools demo
- Seamless integration of SDL, SCADE, Simulink, C, Ada, ASN.1 and AADL
Example from the “Guide for the use of the Ada Ravenscar Profile in high integrity systems”

- Typical example of RT system patterns
- AADL generated code vs. Ada hand-coded

Same functional model
- Both are analyzable with RMA and RTA
- Shares same code quality enforced by Ada compiler

For LEON2 targets
- Penalty of 6% in memory size, equivalent WCET
- Big improvement in analysis
Model a simple radar
  - Integrated in Ocarina

Modeled with STOOD, checked by Cheddar, simulated

Code generated with Ocarina for LEON2

Let us see how it runs ..
AADL to C
Experiments in ANR Flex-eWare
Ocarina’s AADL runtime #2: C/RT-POSIX

- PolyORB-HI/C
  - Targets C/RT-POSIX and C/RTEMS
    - Set of macros to support other RTOS
  - Tested on multiple operating systems
    - Native, GNU/Linux
    - Restricted libc: GNU/Linux on Nintendo DS and Nokia 770
    - POSIX RTOS: RTEMS
  - Tests demonstrated a limited subsystem of RT-POSIX & libc is enough to support AADL
  - Performance comparable to the Ada version

- Used in the ANR Flex-eWare project by Thales
  - 500+ downloads of the MyCCM-HI toolchain
Flex-eWare project (2009)
Merging CCM and AADL

- Using ASSERT philosophy: combining notations
- LwCCM is interesting for system designers
  - Comfortable with the OMG
- Map onto AADL for consolidation
- Generate code using Ocarina
- Uses AADLv2
AADL to IMA
Ocarina’s AADL runtime #3: IMA-like

- **POK (http://pok.gunnm.org)**
  - A bare board AADL runtime: both an AADL runtime and a kernel
  - Finely tuned using AADL properties
  - Follow ARINC philosophy for time and space partitioning

- **Separate services as more as possible**
  - Restrict functionalities of each service
  - Fine-grain configuration
  - Ex: include static scheduler, not RMS

- **Configures resources of each layer**

- **Main goal: use ONLY needed functionalities**
  - Help the certification (cf. DO178B)
  - Low memory footprint
MBE development process for avionics system
- Full automatic process

Modeling guidelines
- Cf. ARINC653 annex

Validation
- Check structure

Code generation
- Provides certification hints

Runtime
- ARINC653 compliance
Partitioned architectures

Partition

Module/Kernel (provides time & space isolation)

Partition

Module/Kernel (provides time & space isolation)

Partition

Module/Kernel (provides time & space isolation)

Partition

Module/Kernel (provides time & space isolation)

Bus

Intra-partition communication

Isolation across partitions

Inter-partition communication

Intra-partition communication

Isolation across partitions

Inter-partition communication

Inter-partition communication
Case study

Diagram showing the generation of applications from AADL models using Ocarina.
To conclude

- **Not presented**
  - REAL: a constraint language to check properties on system
    - E.g. Bell-LaPadula, Biba, ARINC consistency, …
  - AADL-to-Petri Nets: Timed and Colored, to perform model checking on architectural behavior
  - Bound-T integration: compute WCET of AADL runtime
  - Ocarina Eclipse plug-in: combination with OSATE
  - Behavioral annex: to test and debug the draft ;)
  - Automatic execution of model: integrate compilation and run on simulator or real hardware in one click
  - Code coverage of the model’s generated code
  - …