About TELECOM ParisTech
1330 students: engineer, Master, PhD degrees
- C/S, Network, Signal, Electronics

160 full-time professors

10 000 TELECOM ParisTech engineers all over the world

Research in ENST is part of the LTCI laboratory
- LTCI is part of French CNRS, for communication technologies

Our C/S dept, focuses on: software & system engineering, network technologies, security, quantum networks, cryptology, natural languages processing, databases, semantic Web.

Worldwide Industrial & Academic partnerships:
- Europe, US, Asia
Building a DRE is still a complex issue:
- RT-CORBA, DDS are only partial solutions
- Still difficult to analyze (scheduling, dimensioning)

Goal: propose a methodology, middleware and tools for DRE
- Validation & Verification, configuration, deployment
- Automate the process as much as possible
- Scale up to complex systems
- Ensure reusability (process, code, models, know how, etc)

Architecture as key enabler to build DRE systems
- Key to system validation, scalability, support for application, …
Open Source projects: PolyORB
http://libre.adacore.com/polyorb

Middleware for DRE is a moving target
- Configurability: tuning middleware components
- Genericity: deriving new repartition functions
- Non-functional needs: QoS, determinism

Many successful stories for mission-critical apps.
- UIC, Armada: Not a COTS, yet efficient
- TAO “family”: adaptive, difficult to analyze

PolyORB: Revisit Middleware to provide a COTS MW for DRE
- Distribution kernel + “personalities”
- Tailorable: CORBA, DSA, DDS, SOAP with high reuse factor
- Verifiable: amenable to formal verification
- Industrial support by AdaCore
Challenge: configuration of the middleware
- The architecture governs both the configuration and deployment
- Needs a (simple) way to express both
- Should not impede late binding decision such as selection of the run-time environment, model analysis tools
  - Still difficult to achieve with UML/profiles/meta-models (for now)

AADL as a vehicle to address (most) issues
- Express architecture, with analysis in mind and enough expression power to describe “real” systems

Ocarina: suite of AADL tools to generate middleware
- See Flex-eWare and ASSERT
Flex-eWare project: CCM to AADL model transformation

Jérôme Hugues, TELECOM ParisTech
About the Flex-eWare

Flex-eWare is a French-funded R&D project

- [http://www.flex-eware.org](http://www.flex-eware.org)
- From January 2007 to December 2009

**Partners:**
- Thales, Orange Labs, Schneider, Trialog, ST, INRIA, LIP6, Telecom ParisTech, CEA

**Flex-eWare main objective**
- Federate and unify French R&D on component based architectures for embedded systems, from deep embedded to software real-time embedded systems
- Consolidate existing technologies
- To support flexibility in architecture description languages
Flex-eWare technology

Why ? Lot of variability in specifications
- Impact projects, product lines, add costs
- Vendor locks are a real problem

What ?
- To bridge component-based frameworks, mostly CCM and Fractal
  - Thales: CCM is OMG’s CORBA Component Model
  - Orange: Fractal is INRIA’s, built around micro-Kernel-like architecture
- Both have ADL, APIs, code generators, etc.

Flex-eWare component model (FCM) to unify concepts
- In one unified process, unified formalism
- Mapped onto other formalisms depending on the selected runtime
One formalism usually implies also one runtime
- Including costly like CORBA on top CORBA CCM

Usually big assets, hard to migrate
- Designers prefer high-level formalisms, or the one they know

Variability in tools (formalisms, API, performance) increases cost, vendor locks

Solution: take advantage of OMG MDA: PIM, PSM
Formalism/Runtime for CBSE

- **One input formalism: FCM**
  - Synthesis of common industrial practice
  - High-level to fit system designers need

- **Two tracks in the project**
  - Fractal
  - CCM

- **TELECOM ParisTech involved in the CCM track**
  - Goal: to provide support for runtime

- **Thales is a long time OMG defender: CCM, DDS, MARTE are co-designed by Thales**

- **Modeling is good, but need a performant runtime**
  - CCM drags CORBA dependencies
OMG’s answer to Java EJBs

- Based on OMG CORBA

Multiple languages: IDL (interface), IDL3 (component), CIDL (lifecycle), CIF (API for implementation),

Associated compilers (IDL, ..) and runtime
About CORBA CCM

- CORBA CCM has a few implementations
  - CIAO, MICOCMM, OpenCCM
  - Usually follows the OMG rules

- CCM is an increment of CORBA, built over CORBA
  - Ideally, can take vendor#1 for CCM, vendor#2 for CORBA

- Implies large OO code space
  - In the 10MB range because of large patterns: factories, homes, …

- CCM and Real-Time?
  - CCM used mostly in the RT world, see Schmidt et al. papers
  - But still big code space, hard to analyze, or even trust

- LwCCM addresses some issues, but still large in size
  - Remove some services like transaction, introspection, …
Flex-eWare’s approach: Merging CCM and AADL

- Keep the best of each

- **LwCCM is interesting for system designers**
  - Comfortable with the OMG world
  - Large model base, know how, modellers
  - Extension: COAL to model precise deployment, modes

- **AADL is better for system integrator**
  - More precise definition of resources, semantics
  - V&V tools: schedulability, latency, resource, ...

- **Ocarina proposes AADL to code generators**
  - Minimal footprint through optimal code generation
  - Generate code instead of relying on large runtimes
Solution: take advantage of MDE capabilities to decouple each stage of the process

Use OMG’s languages (IDL3, CIDL) + extensions to ease deployment

Map onto AADL for consolidation

Generate code using Ocarina
Benefits

- From the system designer PoV: same input, no change
- From the system integrator PoV: completely different system
  - Container is built on top of AADL runtime services
  - Lighter source base: x100 factor
- Amenable to all AADL analysis + processing by Ocarina
  - Better code quality than full OO, important for embedded RT
About Flex-eWare

- Flex-eWare’s contribution to AADL
  - Shows how to place one formalism on top of AADL
  - To ease transition, to please engineers
  - But also to bring analysis capabilities

- First results beyond expectation: significant reduction factor
  - One component: 20 KB (vs ca. 100 KB)
  - One thread: 100 KB of memory vs. unpredictable

- Sources to be released by end of February 2009
  - See http://www.flex-eware.org for details
Ocarina, an AADL-to-X generator: status & work in progress

Jérôme Hugues, TELECOM ParisTech
AADL in TELECOM ParisTech

- 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 required code
- vs. relying on huge framework a-la CORBA

**Open source projects:**
- Ocarina: toolbox for AADL
- PolyORB-HI: C/RT-POSIX and Ada2005 runtimes for AADL
Ocarina features

- Ocarina is a stand-alone tool for processing AADL models
- Ocarina proposes an API to build your own AADL tools
  - Like Ocarina itself, but also Cheddar (UBO), AADL2SDL (ESA)
  - Parsers, printers, semantic checks, model transformation
  - Compiler-based approach, rather than model-to-text
- Fully supports AADLv1 and partial AADLv2
- Code generation facilities target AADL runtimes
  - Available for both AADLv1 and AADLv2 models
  - Ada HI integrity profiles, with Ada native and bare board runtimes
  - C POSIX or RTEMS, for RTOS & Embedded
- Experimental: Bound-T (WCET), Petri Nets
- Integration of SCADE, Simulink, ESTEREL in progress
Ocarina visibility

- Used in the IST-ASSERT (9/2004 -> 1/2008) projects
  - Validated on industrial case studies

- Ocarina & AADL used jointly in ANR Flex-eWare and MOSIC
  - Evaluation of DRE models performance, code generation
  - In a CCM context, mapped onto AADL models

- Ocarina is part of the TopCased project
  - To propose Ocarina as a plug-in for OSATE
  - Part of the “model bus” philosophy of Eclipse

- Ocarina featured on http://libre.adacore.com
  - Open source projects hosted by AdaCore
  - Enhance visibility from the Ada community
  - Highlight benefits of AADL tools for the HI domain
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
  - Scientific papers on the use of AADL
  - Teaching materials for Master degree

- **PolyORB-HI AADL runtime**
  - Ada 2005 and C/POSIX
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 from ESA
- Supports native, LEON2, ERC32 targets
  - With Ethernet or SpaceWire connections
  - Runtime can be configured to use other drivers
- Validated in the context of IST-ASSERT
PolyORB-HI/C

- Targets C/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
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_Sink: in event data port

Data_Source: out event data port

SC_1

Sender_Thread

Send

SC_2

Receiver_Thread

Update

Local Object

Update

Local Object

Receiver_Thread

Read

Watcher_Thread

Watch

Read

Watcher_Thread

SC_3

Data_Sink: in event data port

Lehigh Wire

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

SpaceWire

LEON TSIM

LEON TSIM

LEON TSIM

J. Hugues, INFRES
The ASSERT ESA demonstrator (2008)

- Stood + Ocarina + ASN.1 tools demo
- Seamless integration of SDL, SCADE, Simulink, C, Ada, ASN.1 and AADL
AADL vs. manual coding (2008)

- Example from the “Guide for the use of the Ada Ravenscar Profile in high integrity systems”
  - Model a pump system, typical example for RT systems
  - 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 phase
Ocarina examples

A set of pre-built Ada generated examples available at [http://aadl.enst.fr](http://aadl.enst.fr)
- Examples from CMU/SEI, ASSERT, internal
- For Linux, LEON and ERC32 platforms
- Can be compiled for other native platforms

A set of educational material is available
- Build your own lab session using AADL
  - Then perform schedulability analysis, code generation, test
- For master degree, or in-house tutorials
Ocarina’s Eclipse plug-in

Better integration with OSATE

Status is alpha, mail to Ocarina-users@ if you are willing to test
Conclusion and Ongoing Work

- AADL proved it is interesting for our partners to build and generate code
  - IST-ASSERT, Flex-eWare, AdaCore, Thalès, SAGEM, MBDA

- Ocarina is now available as both source and binaries packages
  - Use it, test it, report bugs to Ocarina’s mailing lists

- Some case studies are available, need more
  - Do not hesitate to send us models!