POK – System validation and certification

Julien Delange <julien@gunnm.org>

Forewords

- **The POK project**
  - **Design and implement safe and secure system**
  - **Complete development process with model-based engineering**
- **Now, focus on system validation & certification**
  - **How AADL helps you in system validation**
  - **How can we certify a system against its requirements ?**
  - **What are the benefits, the limits ?**

Outline

- Introduction, remainder of AADL modeling
- Specification validation
- System certification
- Conclusion

Outline

- **Introduction, remainder of AADL modeling**
- **Specification validation**
- **System certification**
- **Conclusion**

Introduction

- **AADL modeling for partitioned architectures**
  - **ARINC653 dedicated modeling patterns**
  - **MILS compliance**
- **Code generation**
  - **Generate both code and partitions**
  - **ARINC653 compliance**
  - **Integration with other development processes**

Toolset

- **Eclipse/TOPCASED/OSATE**
  - **Modeling framework**
  - **Efficient for end-user**
- **Ocarina**
  - **Code generation functionalities**
  - **Efficient for tool-processing**

Toolset for validation/certification

- **Ocarina/REAL**
  - **Requirements enforcement**
- **Cheddar**
  - **Scheduling simulation & validation**
- **SPOQ**
  - **Application behavior validation**

Outline

- Introduction, remainder of AADL modeling
- **Specification validation**
- System certification
- Conclusion

Specification validation

- **REAL, theorem-based language**
  - **Implementation in Ocarina**
  - **Verify requirements at model-level**
- **ARINC653 & MILS dedicated theorems**
  - **Integrate in the POK toolchain**
  - **Automatically invoked**
  - **Verify ARINC653 requirements**
  - **Safety and security validation**

REAL overview

- **Theorem-based language**
  - **Math approach**

- **Operate on components set**
  - **Predefined sets: processor set, virtual processor set …**
  - **You can build your own constrained set**

- **Manipulate model properties**
  - **Access to components property (ex : Period, Bus rate, ...)**

REAL theorem example

a partitioned pok operating system

Analyze each thread of the AADL components hierarchy under the *t* variable

Check that ***Source_Data_Size*** property is defined on thread *t*

```
theorem Check_Thr_Memory_Properties

 foreach t in thread_set do
   check
     (Property_Exists (t, "Source_Data_Size") and
      Property_Exists (t, "Source_Code_Size") and
      Property_Exists (t, "Source_Stack_Size")
     );
end Check_Thread_Memory_Properties;
```

verify that threads specify properties related to memory requirements

REAL example – cont'd

Analyze each process of the AADL components hierarchy under the ***prs*** variable

Get all threads contained in the ***prs*** process under the ***Thrs*** variable

```
theorem check_threads_processes
  foreach prs in process_set do

    Thrs := {x in Thread_Set | is_subcomponent_of (x, prs)};
    check
      (sum (property (thrs, "Source_Data_Size"))
       <
       property (prs, "Source_Data_Size"));
end check_threads_processes;
```

verify threads data sizes are less than process data size

REAL and ARINC653

- **Validate system consistency**
  - **Modeling patterns enforcement**
  - **All properties are declared**
- **Verify major time frame compliance**
  - **Major time frame $\doteq \sum$ partitions slots**
- **Validate space isolation**
  - **Partition isolation in memory segments**
- **Check for recovery policy trade-off**
  - **Can a partition at a given criticality others at higher criticality ?**

REAL & ARINC653 - example

a partitioned pok operating system

Analyze each ARINC653 module under the ***cpu*** variable

Get the Major Frame value of the current ***cpu***

Get the sum of frames Allocated on the current ***cpu***

```
theorem scheduling_major_frame
  foreach cpu in processor_set do
    check
    (float (property (cpu, "ARINC653::Module_Major_Frame"))
     =
     sum (property (cpu, "ARINC653::Partition_Slots")));
end scheduling_major_frame;
```

REAL and security

- **Validate security policies verification**
  - **Bell-Lapadula**
  - **BIBA**
- **Check security levels isolation (MILS)**
  - **Check security isolation enforcement**
- **Ensure security mechanisms implementation**
  - **Cipher algorithms**
  - **Data protection across a distributed system**

REAL & security - example

```
theorem bell_lapadula
  foreach p_src in process_set do
    VP1     := {x in Virtual_Processor_Set  |
                is_bound_to (p_src, x)};
    B_Src   := {x in Virtual_Bus_Set        |
                is_provided_class (VP1, x)};
    P_Dest  := {x in Process_Set            |
                is_Connected_To (p_src, x)};
    VP2     := {x in Virtual_Processor_Set  |
                is_bound_to (P_Dest, x)};
    B_Dst   := {x in Virtual_Bus_Set        |
                is_provided_class (VP2, x)};

    Check ( Cardinal (P_Dest) = 0 or
     (max (property (B_Src, "POK::Security_Level"))
      <=
      min (property (B_Dst, "POK::Security_Level"))));
end bell_lapadula;
```

REAL - Summary

- **System validation at specification level**
  - **ARINC653 requirements enforcement**
  - **MILS & security validation**
- **Extensible approach**
  - **More theorems, dig into `misc/real.lib` file**
  - **Ability to define your own theorems**

Outline

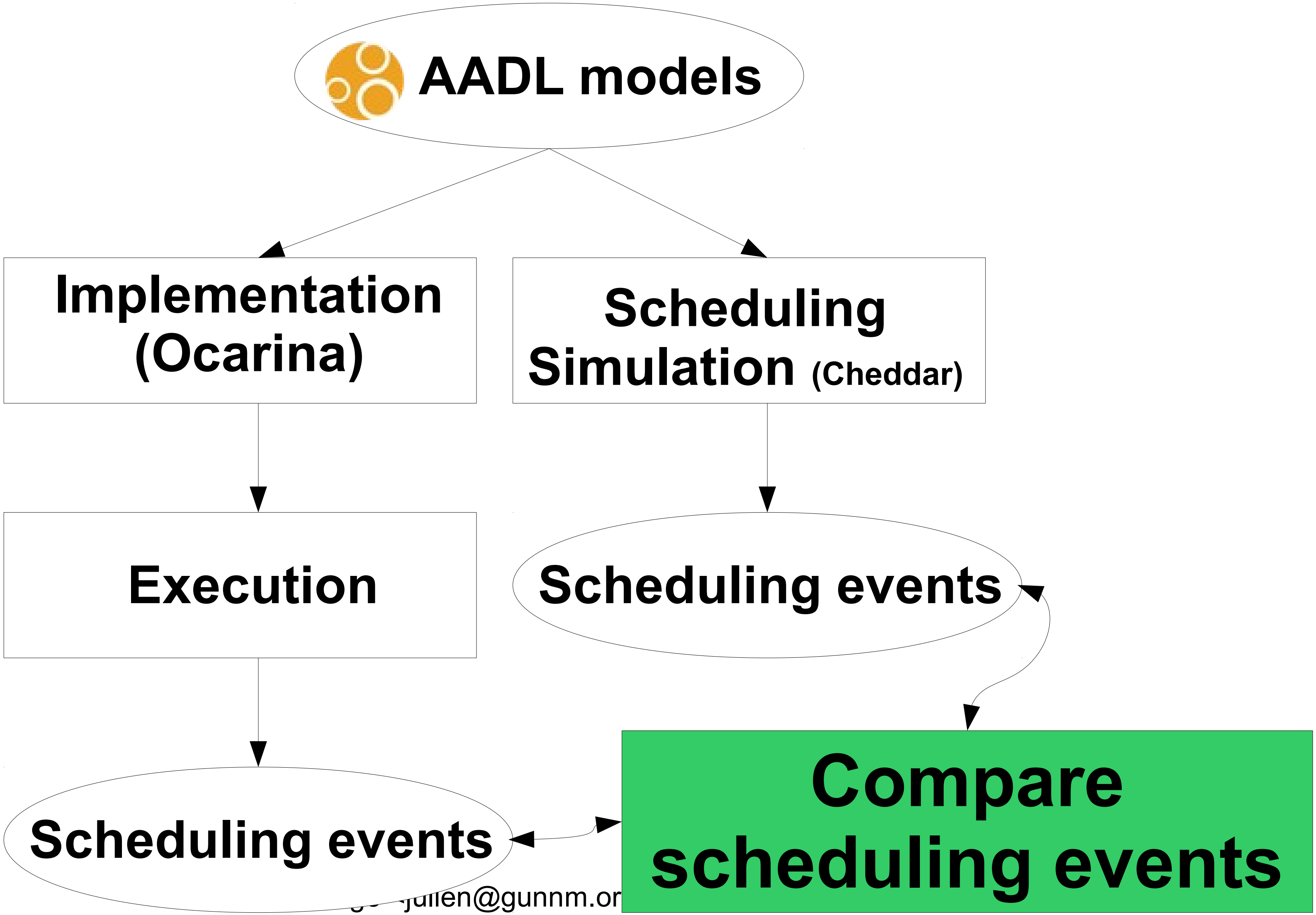
- Introduction, remainder of AADL modeling
- Specification validation
- **System certification**
- Conclusion

System certification

- **Scheduling validation**

- **Behavior checking**

Scheduling validation



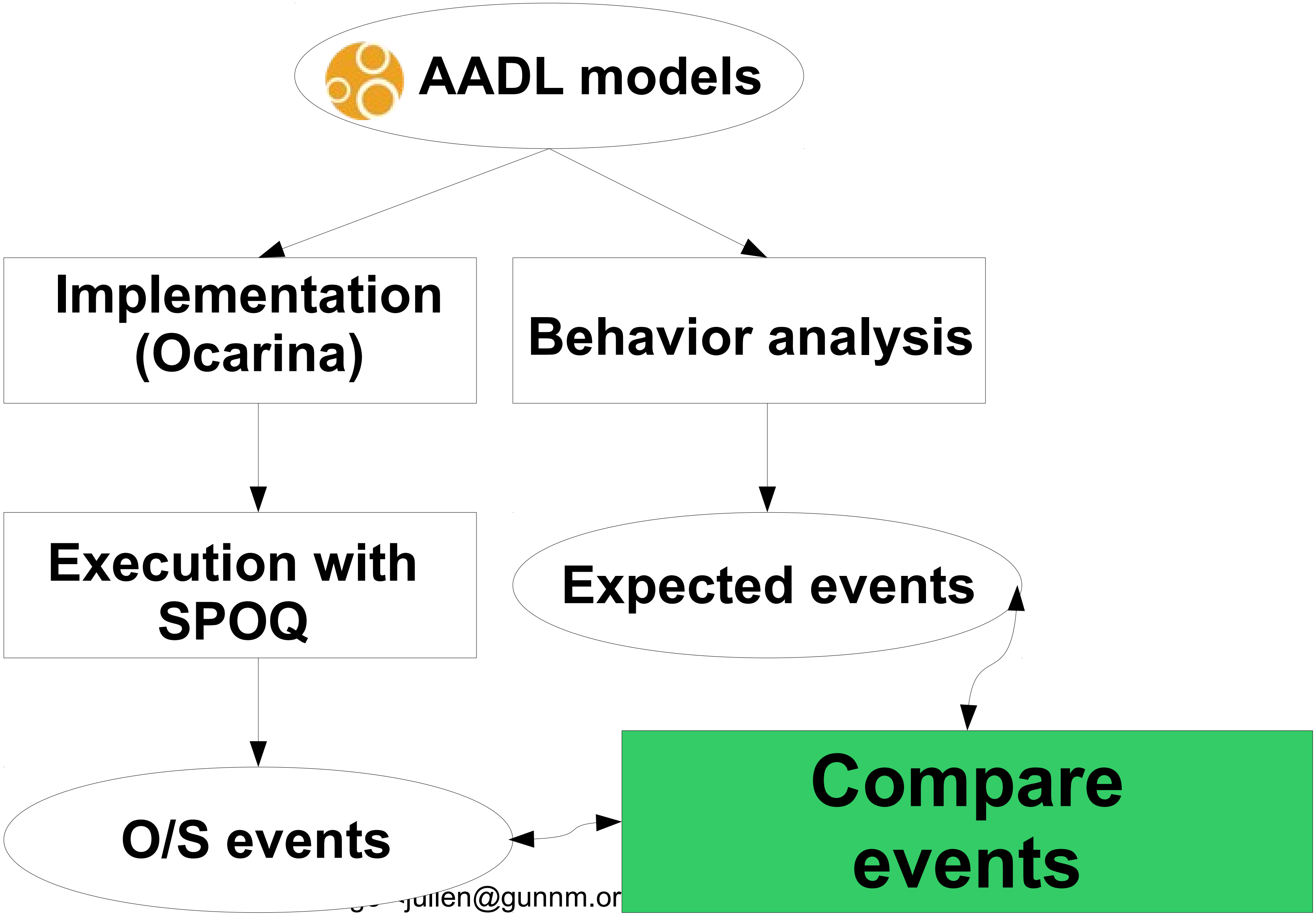
Under the wood

- **Cheddar**
  - **Support for hierarchical partitioning**
  - **Output scheduling as XML file (1)**
- **POK**
  - **Dedicated instrumentation mode**
  - **Output scheduling events compliant with Cheddar/XML notation (2)**
- **Scheduling events comparator**
  - **Check (1) and (2)**
  - **Available in `misc/compare-scheduling-traces.pl`**

Results

- **Engineering efforts required**
  - **Still not an automatic process**
- **Scheduling comparator**
  - **Functional**
  - **Scheduling at runtime faster**
  - **Simulation only cares about WCET**
- **Assists engineers in the certification process**

Behavior validation



Under the wood

- **POK**
  - **Dedicated instrumentation mode**
- **SPOQ**
  - **Dedicated version of QEMU**
  - **Trace syscall, memory segments, etc.**
  - **Support for x86 target**

Results

- **Engineering efforts required**
  - **Still not an automatic process**

- **Behavior verification**
  - **Still manual**
  - **Syscall analysis with time information**

- **Assists engineers in the certification process**

Outline

- Introduction, remainder of AADL modeling
- Specification validation
- System certification
- **Conclusion**

Conclusion

- **AADL for system validation**
  - **Pre and post-implementation valiation**
  - **Reduce errors, costs, improve development reliability**
- **Existing fundation & toolset**
  - **Ocarina/REAL**
  - **SPOQ**
  - **POK**
- **Improvements needed**
  - **Should be considered as a first step**
  - **Further work required, design new certification tools**

Questions ?