Formal Contracts for Runtime Verification Support in the Ada Programming Language

CISTER - Research Center in
Real-Time & Embedded Computing Systems

André Pedro, David Pereira, Luís Miguel Pinho, and Jorge Sousa Pinto
{anmap,dmrpe,lmp}@isep.ipp.pt, jsp@di.uminho.pt

Motivation

- Static Verification is not sufficient to cope with many of the challenges of modern and future generation real-time embedded systems:
  - state-explosion problem of model-checking;
  - limited automation in deductive reasoning, even with recent advances in SAT and SMT solvers.
- Most of the data important to certify a real-time embedded system is related to extra-functional properties:
  - Duration of tasks;
  - Energy consumption;
  - Temperature management;
  - Other cyber-physical properties.
- Unfortunately, most of the extra functional data is only available and verifiable during execution time.

Runtime Verification

- Runtime Verification is the discipline that studies formal theories and that proposes methods to generate monitors capable of observing and verifying formal specification during execution time:
  1. Formal specifications determine the property of interest that must be verified;
  2. Monitors are generated from that specification and are instrumented into the system.
- Typical contracts establish properties about the program that are verified via static approaches
- Runtime Verification behavior should follow the same principles:
  - Users define contracts about properties that he wishes to see verified upon execution;
  - The system is responsible for generating the monitors from those contracts.

Ada 2012 and Contracts

- Contracts enhance trust in the system by establishing a compromise between requirements and implementation
- Ada 2012 provides a sub-language for specifying contracts:
  - Checked at runtime via asserts, or;
  - Statically verified using the SPARK toolset.
- Contract language provides the ideal environment to specify properties that we need to be checked upon run-time (e.g., timed behavior of tasks)
- Runtime Verification contracts can be pre-processed to generate the monitors, and afterward removed, thus preserving the standard Ada 2012 contracts

Underlying Architecture

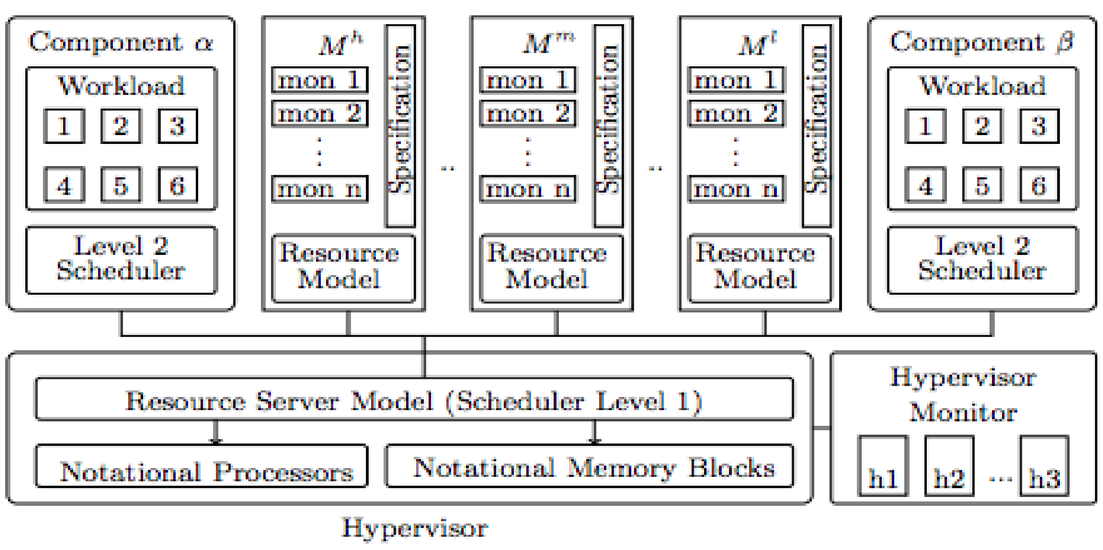

Fig. 1: Component-based Monitoring Architecture (CMA)

Pluggable Formal Theories

Timed Regular Expressions

```
task type T_Simulation (period: integer; deadline: integer)
 with
   Monitor_Mode => Event_Triggered,
   Monitor_Case => ( RMTLD ,
      T_Simulation'Event(Task_Release) next implies
      duration[T_Simulation'Time(period)]
      T_Simulation'Event(ANY) < T_Simulation'Time(wcet)
    );
```

Metric Temporal Logic with Durations

```
protected type Protected_Environment
 with
   Monitor_Mode => Time_Triggered
   Monitor_Case => ( TRE,
     ( Protected_Environment.read_CH4'Event(pre) .
       <(Protected_Environment'Event(ANY))*>[0..20] .
       Protected_Environment.read_CO2'Event(post))*
     ) ,
 is
   function read_CO2 return CO2_Level_State;
   function read_CH4 return CH4_Level_State;
   function read_Air_Flow return Air_Exhaust_State;
   function read_WaterPipe_Flow return WaterPipe_Flow_State;
end;
```

References

[1] Pedro, A., Pereira, D., Pinho, L.M., Pinto, J.S. “Towards a Runtime Verification Framework for the Ada Programming Language”. Reliable Software Technologies – Ada Europe 2014, LNCS 8454, pp. 58-73, Paris, France, 2014.

[2] Pedro, A., Pereira, D., Pinho, L.M., Pinto, J.S. “A Compositional Monitoring Framework for Hard Real-Time Systems”. NASA Formal Methods Symposium 2014, LNCS 8430, pp. 16-30, Houston, Texas, USA, 2014.

ARTEMIS/0001/2013 - JU grant nr. 621429 (EMC2). Co-financed by:

CISTER Research Centre/INESC-TEC
ISEP, Polytechnic Institute of Porto
Rua Dr. Antº Bernardino de Almeida, 431
4200-072 PORTO Portugal
tel: +351-228340502
fax: +351-228340509
http://www.cister.isep.ipp.pt
cister-info@isep.ipp.pt