AADLv2, a Domain Specific Language for the Modeling, the Analysis and the Generation of Real-Time Embedded Systems

Julien Delange, CMU/SEI, USA
Jérôme Hugues, ISAE, France

Software Engineering Institute
Carnegie Mellon
Copyright 2015 Carnegie Mellon University and IEEE

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

NO WARRANTY. THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN “AS-IS” BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

This material has been approved for public release and unlimited distribution.

This material may be reproduced in its entirety, without modification, and freely distributed in written or electronic form without requesting formal permission. Permission is required for any other use. Requests for permission should be directed to the Software Engineering Institute at permission@sei.cmu.edu.


DM-0002585
About the presenters

> Jérôme Hugues: from ISAE, leads the Ocarina project, a AADL tool chain, member of the steering committee of SAE AS-2C, 8+ years on AADL

Resources for this tutorial

> Information on AADL
  » http://www.aadl.info: updates on AADL standard
  » http://www.openaadl.org: resources around Ocarina

> Materials for this tutorial (slides and models) are on http://www.openaadl.org

> Feel free to contact us for more details
Real-time systems are defined as those systems in which the correctness of the system depends not only on the logical result of computation, but also on the time at which the results are produced”. Stankovic, 1988.

A safety-critical systems is a computer, electronic or electromechanical system whose failure may cause injury or death to human beings.”

Properties we look for:
- Functions must be predictable: the same data input will produce the same data output,
- Timing behavior must be predictable: must meet temporal constraints (e.g. deadline),
- Failure rates must be accounted for
Rely on models and domain-Specific system/software engineering methods, processes and tools to master quality and cost so as to

» Allow for early verifications at design step
» Reduce manual development efforts
» Ensure development consistency
Objectives of this tutorial

> Issues
  » How to model/design a real-time critical embedded system that conforms to requirements?
  » How to analyze the solution from a safety perspective
  » How to design and produce a good architecture?

> One solution among others: use an architecture description language
  » to model the system,
  » to run various verification,
  » and to automatically produce the system

Focus on the AADL2.1 SAE standard (2012)
Goal: to model an avionics-related system (ADIRU)

Let us suppose we have the following requirements

» System implementation is composed by physical devices (i.e. Hardware entities) and software entities: running processes and threads + operating system functionalities (scheduling) implemented in the processor that represent a part of execution platform and physical devices in the same time.

» The main process is responsible for signals processing:
  • General pattern is sensing/processing/actuating

» This system has to operate in an avionic platform, and demonstrate it is safe and secure, extends to both scheduling and safety analysis

» The system is to be run on an ARINC653 OS

» The system must be easy to design, understand and maintain
Goal: use AADLv2 for safety analysis and code generation

Part 1: Introduction to AADLv2 core (~ 50’)
  » Syntax, semantics of the language

Part 2: introducing a case study (~ 20’)
  » A avionics case study (ADIRU)

Part 3: ARINC653 modeling with Code Generation (~ 45’)
  » Modeling ARINC653 systems with specific requirements
  » Generate Module and Partition Code for ARINC653 OS

Part 4: Reducing Architecture Complexity with AADL (~ 20’)
  » Detecting and avoiding complex architecture patterns using AADLv2

Part 5 : Error-Model Modeling (~ 45’)
  » Specify errors and faults propagations policy
  » Generate safety analysis documents (FHA, FMEA, FTA, etc.)