Early Timing, Schedulability and Performance Analysis of Embedded Electronics Architectures

F Corbier, P Dissaux

Early Timing, Schedulability and Performance Analysis of Embedded Electronics Architectures

F. Corbier[1], P. Dissaux[2]

1: Dassault Systèmes, 35 rue Haroun Tazieff, 54320 Maxéville, France
2: Ellidiss Technologies, 24, quai de la douane, 29200 Brest, France

Keywords

AADL, E/E Requirement, Embedded Electronic Architecture, IMA, Integrated Modular Avionic, Timing analysis, Execution performance, ARINC 563 distribution, Memory segmentation, Virtual processor partitioning, Safety and Security analysis, Code generation.

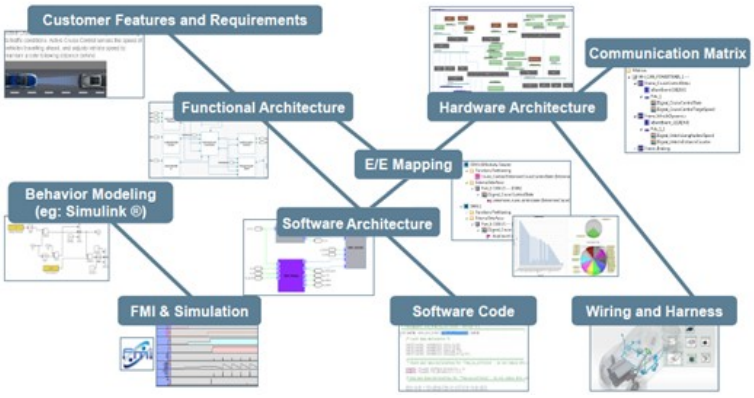
Introduction

Advanced model-centric engineering environments need to comply at the same time with domain specific modelling paradigms for the end user and with specialised languages required by high value added analysis tools. Frequently, designers achieved this by interconnecting existing modelling and model processing tools to minimize development and maintenance costs while using the state-of-the-art solution for each link of the tool-chain. Even then, it is difficult and time-consuming to achieve pre-implementation.

This paper describes an example of such a heterogeneous coupling between a System Engineering platform for embedded electronics (E/E) architectures and a software architecture timing analysis framework.

1. Embedded Electronics Architecture

The growing demand for new functionalities including safety or comfort in modern transportation systems is rapidly increasing the complexity of Electrical/Electronic (E/E) architectures such as Integrated Modular Avionics (IMA) that become the main driver of innovation. Hence, the design of such architectures is turned out to be more and more difficult, labour-intensive, and error-prone using manual workflow rather necessitating a sophisticated tool-chain, for modelling, early-stage verification, validation, and testing of the system.



Dassault Systèmes provides a platform (called 3DEXPERIENCE) that support the data authored by internal applications as well as external tools. The platform also natively provides features allowing engineers to manage versions, traceability, life cycle and diversity (options and variants) of the system (projects and platform/architecture oriented product) and its components / sub-systems.

The E/E architecture definition takes the requirements and functions coming from the system engineers as inputs, and delivers the hardware topology (controllers, devices, actuators & sensors and networks & wires), the software architecture (including partitions, tasks, virtual links…), the mapping of function on Software, the distribution of SW on HW, and also the communication matrixes. The platform allows E/E engineer to integrate analysis framework to cover the standards of industries: for example, it is the case with AUTOSAR for SW engineering in automotive and, as we will see in this paper, AADL for avionic.

2. AADL - Architecture Analysis and Design Language

AADL is an international standard of the SAE, Aerospace Division, under reference AS-5506C, January 2017. This standard help describing software intensive real-time systems and embed a sufficient level of semantics to enable the use of early (pre-implementation) analysis or production tools. AADL propose a textual representation that fully describes models and architecture. This makes it very scalable and appropriate for version and configuration management.

As opposed to UML based modelling solutions, customizing AADL to a particular domain or processing feature consists in reducing the scope of its meta-model (AADL subsets), instead of enlarging it by the definition of new entities (UML profiles). This approach ensures that the various tools of the tool-chain always use the AADL standard semantic.

AADL is a native modelling language for description of the software architecture and the hardware execution platform, and broader integrated system-level. In addition, it can also serve as a pivot language between domain specific modelling solutions – typically system engineering approaches – and analysis or implementation modelling or programming solutions.

Several realizations, in particular from a TASTE, Capella or SCADE System models experimented the use of AADL as a pivot language. This paper describes the use of AADL to connect E/E architecture definition models with timing analysis tools.

3. Early exploration of E/E Architectures

3.1. Workflow

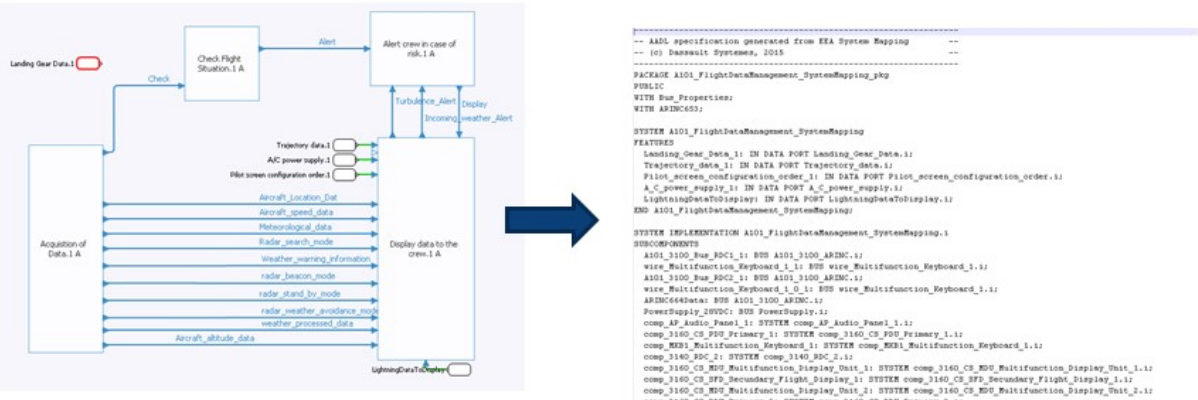
As a single source of truth, the 3DExperience platform hosts Requirements, Functions, and Software and Hardware architectures. The target is to verify early as possible that the defined EE architectures match with requirements, process allocation, network configuration and execution performance constraints.
The workflow is then to generate AADL files from the 3DExperience platform and to use tools based AADL to evaluate static and dynamic performances of the E/S architecture (including scheduling, partitioning, multi-tasking, communication…).

3.2. E/E Architecture (EEA/EEW) to AADL

The 3DEXPERIENCE platform provide applications for the design of high level IMA, including functional and HW architecture, function mappings, communication matrix and signal mappings. We developed connectors to extract data from the 3DEXPERIENCE platform and to generate AADL files for timing and scheduling analysis:

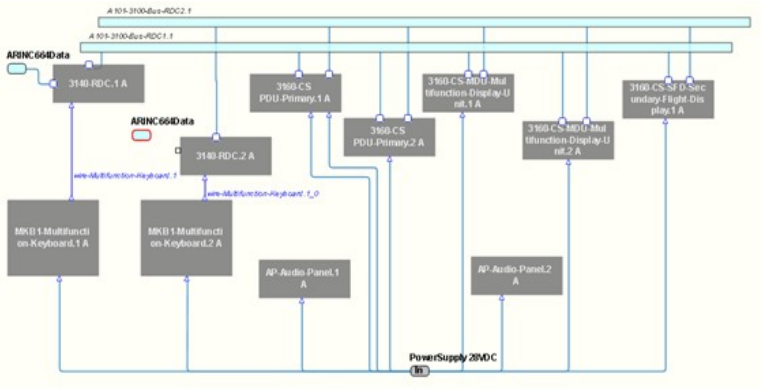
- Functional architecture to AADL
- Hardware topology to AADL
- Software architecture to AADL
- System mapping SW to HHW (including communication matrix and distribution on partition)

Functional architecture to AADL



This layer features the concepts of Function, Instance, Flow, Port (which is technically an instance of a flow), and connections between ports, arranged hierarchically. The translator creates the SYSTEM, IMPLEMENTATION, IN/OUT DATA PORT and CONNECTION elements of the AADL description.

Hardware topology to AADL




```
PROCESSOR processor_3140_RDC
FEATURES
  req_ARINC664: REQUIRES BUS ACCESS bus_A101_3100;
END processor_3140_RDC;

PROCESSOR processor_3160_CS_PDU_Primary
FEATURES
  req_ARINC664: REQUIRES BUS ACCESS bus_A101_3100;
END processor_3160_CS_PDU_Primary;

PROCESSOR processor_3160_CS_MDU_Multifunction_Display_Unit
FEATURES
  req_ARINC664: REQUIRES BUS ACCESS bus_A101_3100;
END processor_3160_CS_MDU_Multifunction_Display_Unit;

PROCESSOR IMPLEMENTATION processor_3140_RDC.i3140_RDC_2
SUBCOMPONENTS
  VP1 : VIRTUAL PROCESSOR VP;
  VP2 : VIRTUAL PROCESSOR VP;
PROPERTIES
  Scheduling_Protocol => (ARINC653);
  Scheduling_Protocol => (RATE_MONOTONIC_PROTOCOL) APPLIES TO VP1,VP2;
  ARINC653::Module_Major_Frame => 20ms;
  ARINC653::Module_Schedule => (
    [Partition=>REFERENCE(VP1);Duration=>3ms;Periodic_Processing_Start=>FALSE;],
    [Partition=>REFERENCE(VP2);Duration=>17ms;Periodic_Processing_Start=>FALSE;]
  );
    Scheduling_Protocol => (RATE_MONOTONIC_PROTOCOL);
END processor_3140_RDC.i3140_RDC_2;

PROCESSOR IMPLEMENTATION processor_3160_CS_PDU_Primary.i3160_CS_PDU_Primary_1
PROPERTIES
  Scheduling_Protocol => (RATE_MONOTONIC_PROTOCOL);
END processor_3160_CS_PDU_Primary.i3160_CS_PDU_Primary_1;

PROCESSOR IMPLEMENTATION processor_3160_CS_PDU_Primary.i3160_CS_PDU_Primary_2
PROPERTIES
  Scheduling_Protocol => (RATE_MONOTONIC_PROTOCOL);
END processor_3160_CS_PDU_Primary.i3160_CS_PDU_Primary_2;
```

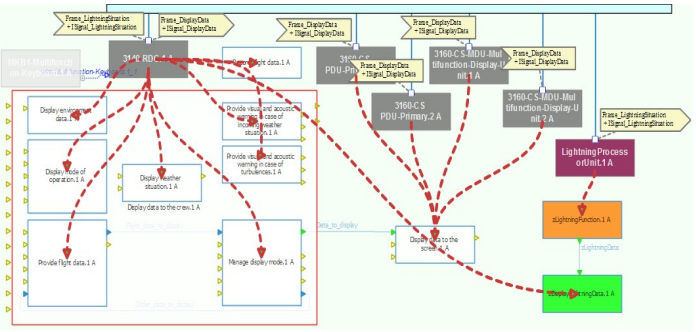
The 3DEXPERIENCE platform supports the design of Hardware Topologies to describe the structure of the E/E components including involved communication systems (buses & wires) to interconnect control units, gateways, sensors and actuators

Software architecture to AADL

The application software implements the dynamic subsystem behaviour. The translation of the software architecture provides PROCESS, THREAD, PORTS and CONNECTIONS. The translator also generates the properties of each threads.

System mapping SW to HW

A system mapping is an artefact describing the allocation of a function/software sub-network (a part of the global network, typically corresponding to the implementation of a customer function) in the final system. It links a Function Network or a Software Network to a given Hardware topology, and contains a series of “Mapper” objects of various categories:




```
PROCESSOR processor_3160_CS_PDU_Primary
FEATURES
  req_ARINC664: REQUIRES BUS ACCESS bus_A101_3100;
END processor_3160_CS_PDU_Primary;

PROCESSOR processor_3160_CS_MDU_Multifunction_Display_Unit
FEATURES
  req_ARINC664: REQUIRES BUS ACCESS bus_A101_3100;
END processor_3160_CS_MDU_Multifunction_Display_Unit;

PROCESSOR IMPLEMENTATION processor_3140_RDC.i3140_RDC_2
SUBCOMPONENTS
  VP1 : VIRTUAL PROCESSOR VP;
  VP2 : VIRTUAL PROCESSOR VP;
PROPERTIES
  Scheduling_Protocol => (ARINC653);
  Scheduling_Protocol => (RATE_MONOTONIC_PROTOCOL) APPLIES TO VP1,VP2;
  ARINC653::Module_Major_Frame => 20ms;
  ARINC653::Module_Schedule => (
    [Partition=>REFERENCE(VP1);Duration=>3ms;Periodic_Processing_Start=>FALSE;],
    [Partition=>REFERENCE(VP2);Duration=>17ms;Periodic_Processing_Start=>FALSE;]
  );
    Scheduling_Protocol => (RATE_MONOTONIC_PROTOCOL);
END processor_3140_RDC.i3140_RDC_2;
```

The translation of the System mapping creates SYSTEMS (electronic components) that contains PROCESS and PROCESSOR (and access to communication BUS). Depending of the configuration, a processor support the execution of full software or provide virtual processors (partitions) as defined in the ARINC 563 standard.

3.3. AADL Inspector

AADL Inspector is a Model Processing framework that can either parse native AADL models or translate foreign models into AADL and then connect them to a variety of verification and generation tools, such as Cheddar for schedulability analysis, Marzhin for event based simulation and Ocarina for source code generation.
The LMP toolbox implement all the processing features (model navigation, queries, constraints or transformations).

3.4. Logic Model Processing

LMP applies the benefits of Logic Programming to Model Driven Engineering. The main idea consists in defining any meta-model under the form of a set of Prolog facts and to process corresponding models with Prolog rules.
The current LMP toolbox is composed of a set of parsers (AADL, XML/XMI, programming languages), generic processing libraries (e.g. AADL instance model generation) and a Prolog engine.

3.5. Timing analysis

The Cheddar tool provides an implementation of a variety of schedulability tests as defined by the real-time scheduling theory, as well as a static simulator that produces a schedule table over the hyper period of the system.
As every formal analysis tool, Cheddar is based on its own meta-model, called Cheddar-ADL. The model transformation (between AADL and Cheddar-ADL) is implemented with LMP and is embedded into AADL Inspector.

3.6. Real-Time simulation

Timing analysis using the real-time scheduling theory is restricted to periodic activities and provides usually pessimistic results. To extend the scope of timing analysis, use of time based or event based simulation is often required.
This is the purpose of the Marzhin tool that implements the AADL run-time to virtually execute AADL models. It thus supports the various scheduling and communication protocols specified by the AADL standard, including asynchronous dependencies.
Marzhin also defines its own set of entities that needs to be created from the original AADL model thanks to a LMP transformation.

3.7. Safety and Security analysis

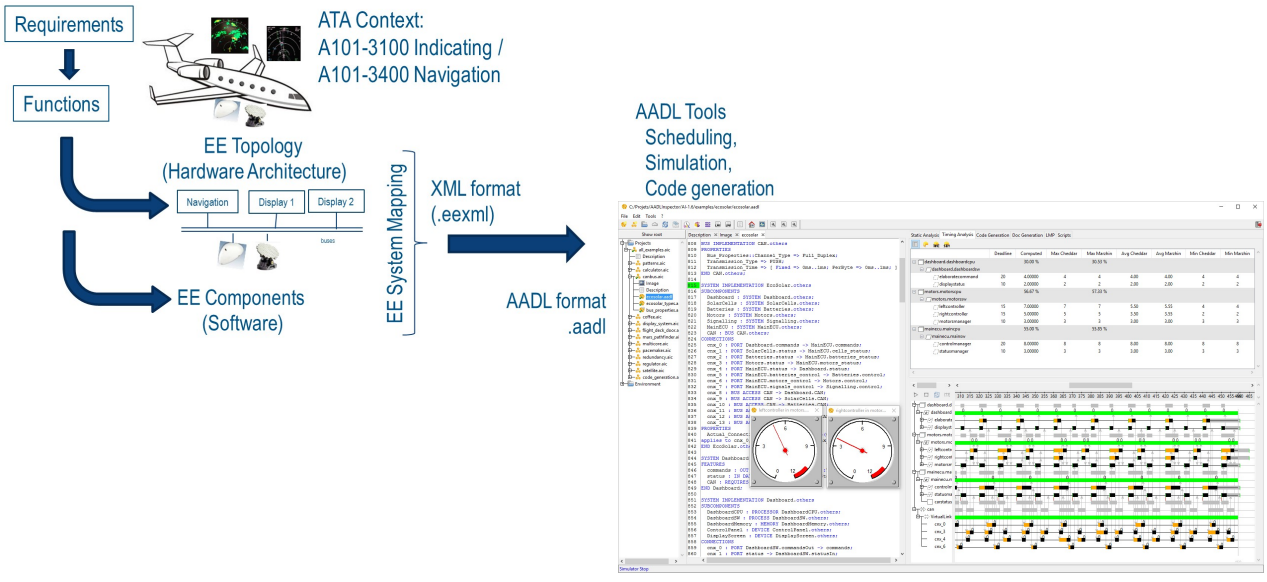
In addition to timing analysis, we are developing additional plugins to add Safety and Security analysis capabilities to AADL Inspector.
Here again the LMP technology will play an important role, either as a direct solution to build new analysis features, such as architectural reasoning in Prolog, or to implement model transformations to specialized languages like SLIM or Alta-Rica.

3.8. Code generation

After the desired level of confidence in the architectural model has been achieved by running early timing, safety and security analysis, following step may be to generate source code in programming language syntax.
Many technical solutions can be foreseen to implement code generation rules for an AADL model and LMP again appears to be a valid option. It can be used either for the direct implementation of the coding rules in Prolog or to check that the input model complies with the requirements of an external code generation tool such as Ocarina.

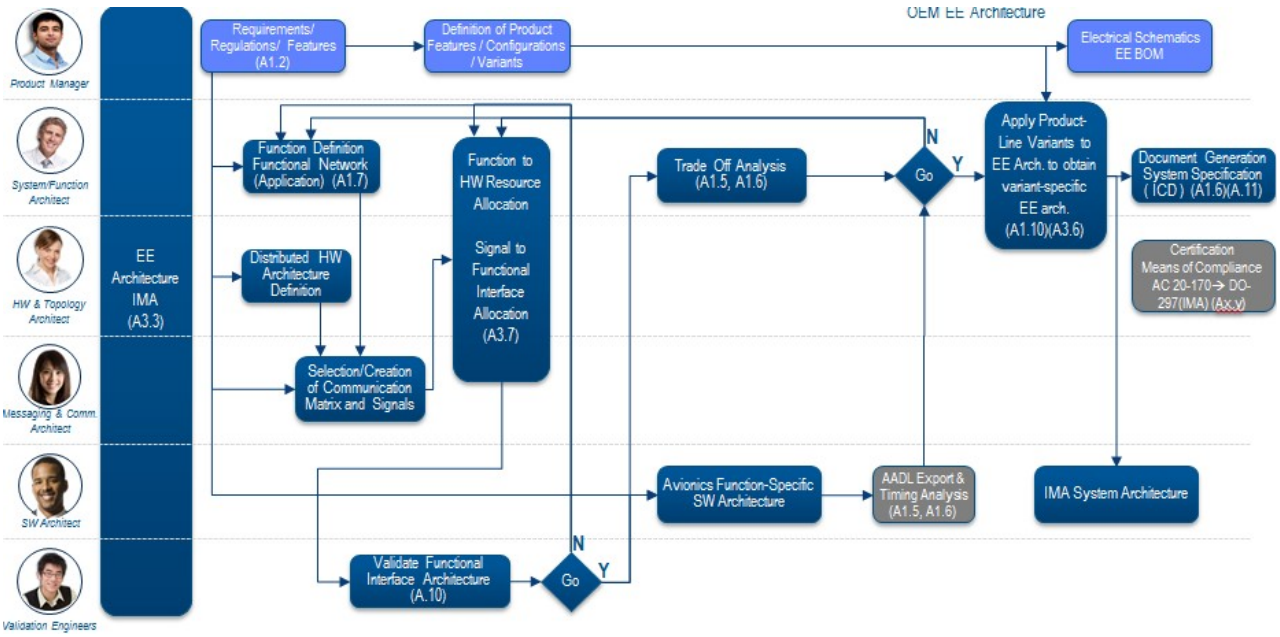
4. IMA case study

The E/E design to AADL analysis tool chain has been deployed on a generic avionic program starting with the ATA definition of the product breakdown structure. Objective was to fully support the design cycle from the product requirement to the software architecture specification through well-defined hardware architecture. At this stage, we generate the associated AADL files for the analysis of space and time partitioning in safety-critical avionics real-time operating systems giving the token to the AADL Inspector framework.



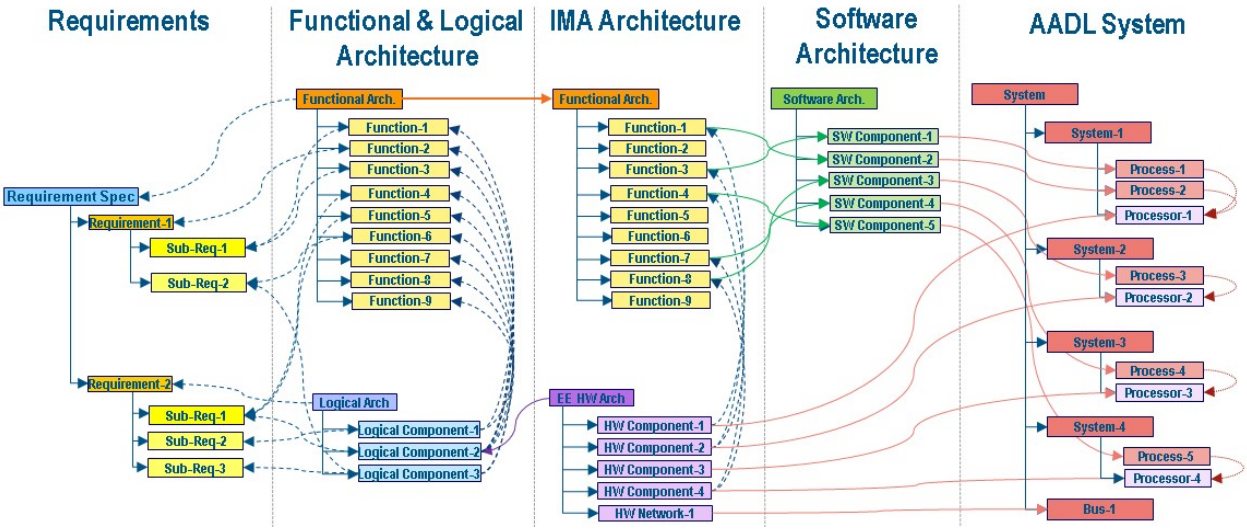
Conclusion

The implementation of operational solutions requires the construction of heterogeneous development environments that maximize the benefit of each link of the tool chain according to its context of use.



This is particularly the case for the articulation between system engineering and software engineering.

The first one has to offer a rich power of expression enough to describe complex systems and a language close to the users. The second must have a sufficiently powerful semantic rigor to allow the use of upstream verification techniques (static and dynamic) and automatic code generation.



It was with this view that an experiment was set up combining E/E modelling for system engineering using the 3DEXPERIENCE platform and AADL for software engineering and implementing real-time performance analysis techniques.

Today, we are exploring additional work such as extending the perimeter of the analyses to safety and security, as well as the feedback of Software Architecture Verification results to the System Engineering model.