AVIONICS SYSTEM DESIGN USING AN ARCHITECTURE DESIGN LANGUAGE

M. Gangkofer, M. Bastl
ESG, Elektroniksystem- und Logistik GmbH
Einsteinstraße 174, D-81675 München

ABSTRACT

The Architecture Analysis and Design Language (AADL) that is defined by the SAE Standard AS5506, allows performing a graphical system design based on a standardised Syntax. For the design of an avionics system in accordance with the principles of Integrated Modular Avionics, AADL proves to be an appropriate tool for the construction of a consistent system representation integrating hardware as well as software components. The detail of specification is determined by the interfaces and components that are defined by the respective IMA standards. “Property Sets”, which are part of the AADL language, allow for application specific extensions, which are used for the characterisation of system components in terms of assigned properties and performance respectively resource specific data. In this manner, AADL can be used for the design of military as well as civil avionics systems. Adding generation of system or configuration data from the XML (eXtensible Mark-up Language) representation of AADL, the system design data become the basis for the system configuration of an IMA system and AADL proves to be an important tool for building an avionics system.

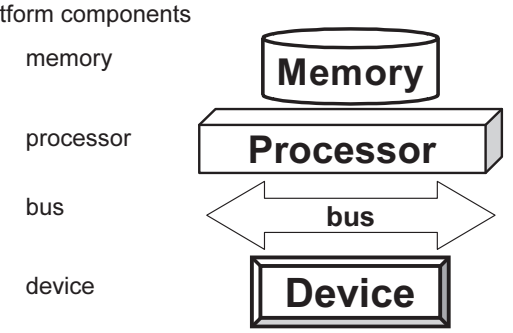
1. THE AADL LANGUAGE

AADL [3] is a textual and graphical language for the analysis and design of the architecture of (embedded) real time systems. It covers both, hardware and software aspects of such systems. The system structure is described as a hierarchical arrangement of Hardware and Software components. AADL further allows describing the functional interfaces and the interaction of components with each other.

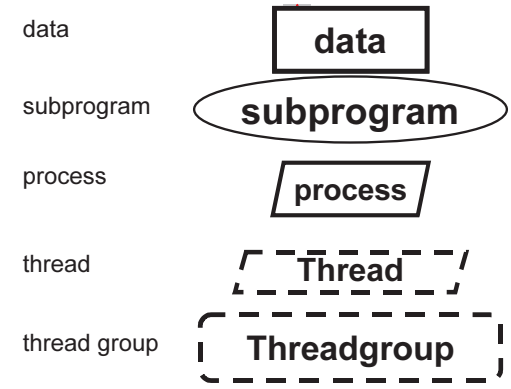
The principal structure of AADL discriminates between the *type* and the *implementation* of components. A component type describes the outside view of the component in terms of its interfaces. A component implementation identifies the structure of the component in terms of its parts, implementation specific modes and the interactions of its parts with each other as well as the mapping of the component’s external interfaces to them. Each component type may have several implementations and it is possible to inherit either the properties of a component type to another component type or from a component implementation to another implementation. This allows to base a system description on templates of (hardware) modules and to describe multiple configurations of both hardware and software components and to treat the transitions between the different modes of the system consistently in a model. The complete system is represented by a complete, hierarchical breakdown of a single system implementation.

The system breakdown describes the principle structure of the entire system in terms of hardware resources and software components that are assigned to those resources. It does not describe the actual construction of the components itself. The AADL standard defines the three categories of components together with their graphical representation:

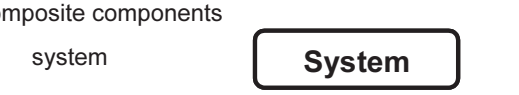
1) Platform components



2) Software components



3) Composite components



Each component may be assigned with specific properties that provide a further description of the components. AADL defines a set of properties, which may be user-defined as needed for appropriate component description.

Further AADL elements are several kinds of ports and connections. Like the components, also the port and connection objects can be described by means of properties.

2. AADL TOOL ENVIRONMENT

At the Software Engineering Institute (SEI) that has been involved in the definition of the AADL standard, an AADL tool named “OSATE” that is integrated into the eclipse platform has been developed. This tool provides

– A syntax sensitive AADL text editor,
– An AADL XML (aaxl) editor,
– A graphical AADL editor,
– Converters between XML and text representation and
– A number of analysis plug-ins.

The OSATE tool is available at an SAE web site [4]. It has recently integrated the graphical AADL editor that has been developed by the TOPCASED programme [5], which is developing an integrated development environment based on AADL and UML and other modelling approaches.

3. IMA SYSTEMS

Integrated Modular Avionics (IMA) is an architectural concept for avionics systems that provides a strict separation of hardware and application software layers for core processing functions in an avionics system. This separation is provided by means of a set of standardised interfaces. The effect of the separation is independence of the application software functions from the hardware resources, which facilitates a modularisation of both software and hardware functions and the definition of standardised hardware modules providing processing resources for application software functions that can be allocated quite freely to the hardware resources.

There are two main representatives for IMA standards, the ARINC 6xx set of standards, especially the ARINC 653 standard [1] and the ASAAC / MoAA set of standards [2]. The ARINC 6xx standards are mainly used for civilian avionics systems; the ASAAC / MoAA standards are designed for military avionics systems. The ARINC 653 standard makes use of modularity as application software partitions that guarantee processing resources in terms of memory and time are allocated to processing resources by means of configuration tables. The allocation is performed in a static manner and the freedom of allocation is usually limited to a single computer (Line Replaceable Unit – LRU), which may provide special interfaces, which application functions may depend on. Therefore, the configuration data are limited to the configuration of a single processing resource or a single LRU. Due to specific military requirements like high availability also under combat damage and prolonged maintenance periods, the ASAAC standards support a more comprehensive approach of a modular system that provides a system wide configurability for the allocation of application functions to processing resources.

For both kinds of IMA standards, the generation of configuration data is elementary for the correct operation of the IMA system and the configuration must be correct and consistent. Therefore, the most obvious approach for generating correct and consistent configuration data is to base these on a complete design model representing the entire system. Such model can be queried for completeness and checked for consistency. Therefore, a system design in AADL provides a model that can be verified and used further in the development process to generate configuration data in a consistent manner.

4. DESIGN OF IMA SYSTEMS USING AADL

The design of a system in general consists of the following activities:

– The system functions are broken down into components.
– The functions allocated to the system components are derived from the system allocated functions.
– Components are separated into software and hardware functions, respectively components.
– The structure of the system, its interfaces and component interactions are determined.

This is an iterative process, which is finished when a level of buildable hardware and software components is reached.

For an IMA system, this means to define the available resource components with the associated network structures and the available application software units and the interfaces between the software modules. In the following sections, the description of the hardware and the software aspects of an IMA system are described in separate models. Then it is shown how the separate descriptions of hardware and software aspects of the systems can be integrated into a single model.

5. SAMPLE SYSTEM

The following example is based on the AADL model of a mission management application that is implemented on an ASAAC / MoAA system platform, which is named the ASAAC IMA Core Processing Platform (CPP).

5.1. Resource Model

The resource model of an IMA system on the highest level consists of AADL system components that represent the LRUs or the modules, the system consists of. Each of these units has network interfaces and the connection of these specifies the structure of the network. The network interfaces are mapped to data access ports and the network connections to data access connections. The IMA CPP consists of five modules and a network switch is illustrated in FIG 1.

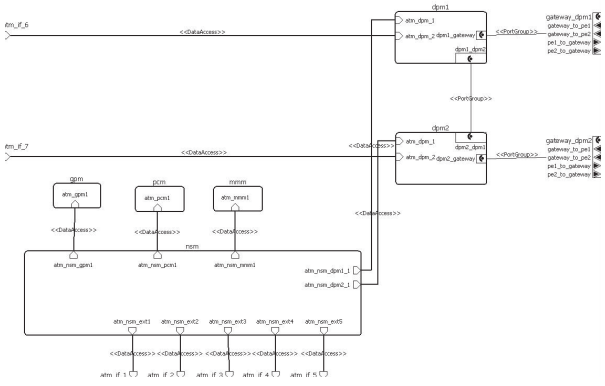


FIG 1. AADL Model of an IMA platform

In addition to the network connections in FIG 1 the data links between the modules are specified as event data

ports and the associated connections. Because there are multiple of such connections between two modules, these are summarised to port groups and the associated port groups connections.

From this level that shows the System components, i.e. the modules are further broken down into a more detailed picture of the structure of the modules that is shown in FIG 2.

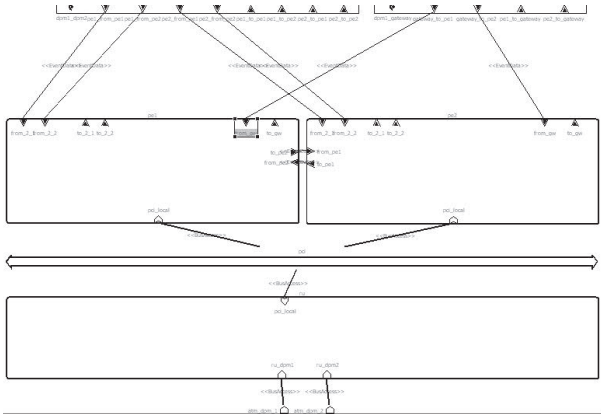


FIG 2. AADL Model of a Data Processing Module

In this example, a Data Processing Module (DPM) with two processing elements is modelled. The port groups are now resolved into individual ports and the connections are associated with actual data links between processing elements. As the data links, which represent a link of two processing elements over the network, are objects, which are relevant for the system configuration and are therefore specified by means of properties that are summarised into an AADL property set.

In a third and final step the processing elements are modelled as shown in FIG 3.

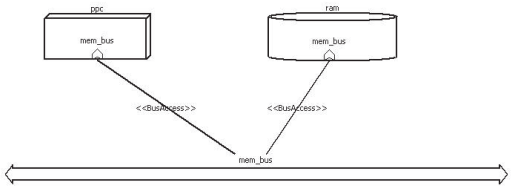


FIG 3. AADL Model of a Processing Element

At this level finally the processing resources in terms of memory space and processor capacity are identified and can be described by means of property sets, if for the configuration of a system the resource consumption of processes (memory) and threads (processor capacity) shall be matched with the resources provided by the system hardware platform.

By means of the AADL modes being assigned to the resource model, different configurations of the system resources that may result from failure or damage scenarios can be integrated. A failed module and its associated connections would then be available in one mode and not available in another mode.

5.2. Application Model

The top level application model is shown in FIG 4. This model shows the processes, which provide the application functions and their interconnection with communication channels. Within the frame of the ASAAC / MoAA standards process is the notion of an encapsulated, functional unit that communicate with each other only by means of message based communication channels, which are called “Virtual Channel” (VC). In the context of ASAAC / MoAA processes represent a protected memory area. Therefore the properties assigned to processes are:

– Process identifier
– Process memory size
– Executable name

In the context of ARINC 653, the structure of the model would be identical. In the ARINC 653 context instead of “process” the term “partition” is used and the meaning of a Virtual Channel is an interconnection of partition ports. However in the ARINC context partitions does not only represent a memory partition, but also a time partition, therefore timing information, such as WCET and scheduling frequency have to be added as partition properties.

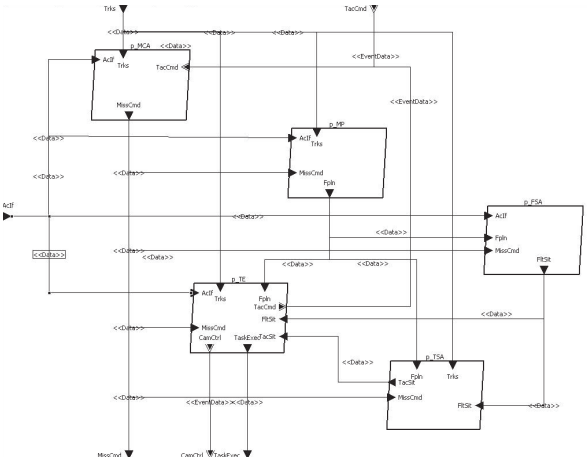


FIG 4. Model of a Mission Management Application

The properties that are assigned to the communication channels of the application model that is illustrated in FIG 4, split into properties of the data type that is associated with the connections, and the resource properties that are associated with the terminals of the communication channel. i.e. the sending, respectively receiving ports, which form the interface of the application processes.

The connection properties are:

– A Virtual Channel identifier,
– A reference to an AADL data type, and
– The maximum message size.

These VC properties refer to the message data type that is associated with the resource. AADL allows several options to specify the connection. It can be specified either as a data connection or a event data connection and it can further be specified either as immediate connection or as a delayed connection.

The port properties are:

– A process local VC identifier that is used as reference by the application functions that are implemented by the process,
– The required buffer size (which should be identical with the maximum message size),
– The number of buffers required for the port,
– Queuing options, and
– Optionally message frequency information that defines communication bandwidth requirements.

These local VC properties specify the resource requirements of an individual process, which are needed to attach this process to the communication channel. This means that at this level the application communication requirements are already specified completely, because the IMA standards restrict the communication of processes with each other to exchange of message across a Virtual Channel.

In the context of the ARINC 653 standard, the partition definition already specifies also the timing requirements of a partition. The scheduling of ARINC 653 processes, inside a time partition has no more influence on the split-up of the processor time resource, which is already fixed in the properties of the partition.

In the context of the ASAAC / MoAA standards, the processor time resource is mapped to threads, where threads are scheduled individually independent of the process in which they are contained. Therefore, here a further step is required to complete the modelling of the resources, which are required by the application function functions. In this step, as illustrated in FIG 5, the threads that are part of a process are identified together with the communication channel requirements of the individual threads. The communication channel specifications are only required in case a process would be split between several processors. The threads are uniquely mapped to a process, because they are defined in a process model, which means a further breakdown in the AADL structure.

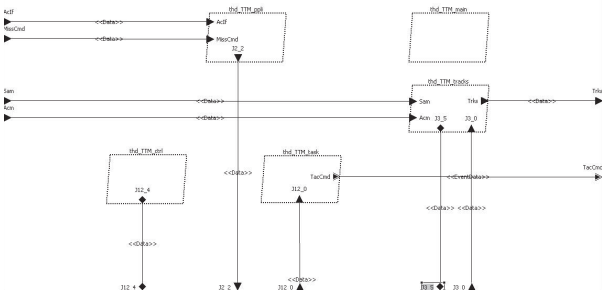


FIG 5. Model of an Application Process

AADL does allow further breakdown of threads into subprograms, data items and programming interfaces. At the level of breakdown that has been described here, all resource requirements matching the resources that are provided by the resource model are already sufficiently described. Therefore in the context of the referenced IMA standards a further breakdown of the model does not make sense within the AADL model and should be performed as part of software design activities for the development of the application processes using appropriate tools like the UML.

The application functions should be separated into appropriate functional units, that match with the system modes and the system hierarchy definitions of the System Configurations model that is described in the next section.

5.3. System Configuration Model

The ASAAC / MoAA architecture standard also specifies system modes, which represent different functional system states, which are associated with sets of application functions. The ARINC 6xx standards do not provide an analogous concept.

AADL provides also a mode concept. AADL modes are always specified for individual components. Therefore the system modes can be specified within the top level system implementation as illustrated in FIG 6 and be propagated into the breakdown of system components, that is shown in FIG 7. At the top level the system modes represent the functional states of the system. In the shown example this are only an initial state and an operational state, but within a more complex system for flight phases like take-off, cruising or combat can be mapped to different functional configurations. As already mentioned above, the resource model can also provide a set of resource configurations.

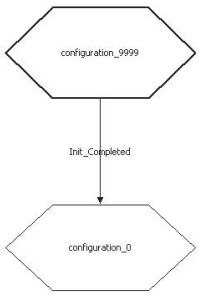


FIG 6. Top Level System Modes

A further step in the definition of the system is the hierarchical breakdown of the system into functional units, i.e. subsystems, respectively in the terminology of the ASAAC / MoAA standards, “Integration Areas” (IA). In the AADL structure, this means a breakdown of the top level system structure into system components as shown in FIG 7. Each of these system components defines a set of its own system modes, which have to match with the system modes of the parent system and may also define additional modes for instance to match different resource configurations.

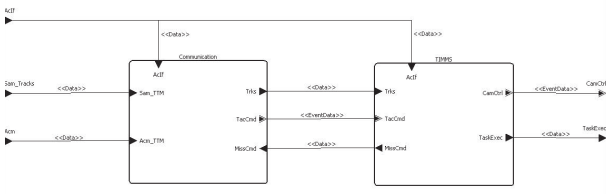


FIG 7. System Hierarchy

In accordance with the ASAAC / MoAA standards each integration area, as well as every processing element as the terminal components in the system hierarchy, is associated with a (standardised) set of system management functions. The term used for these function

is “Generic System Management” (GSM). The implementation of the GSM functions does consume resources like the applications function do, therefore, in order to get to a complete system model, the GSM functions, that are associated with the elements of the defined system hierarchy have to be specified analogous with the definition of the application model. The model itself is not generic, but depends on the design details of the GSM functions. FIG 7 refers to an implementation that has been used within the ASAAC standardisation programme.

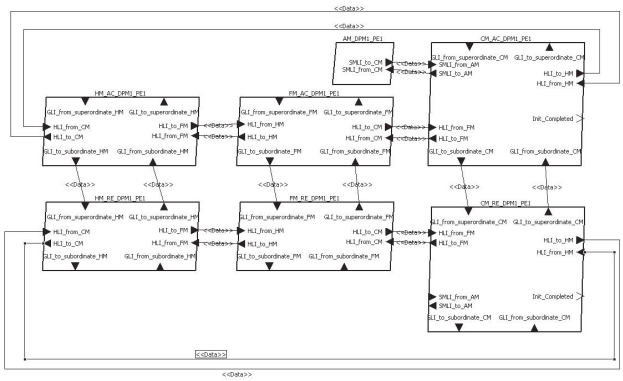


FIG 8. System Management Functions

The system configuration model defining

– A system modes and transitions model,
– A system hierarchy, and
– A model of the system management functions

provides an AADL framework into which both resource model and application models can be integrated in order to get a complete AADL System Model as described in the next section.

5.4. System Model

On the basis of the Systems Configuration model, the resource model and the application model, an overall System model as sketched in FIG 9 can be integrated in several steps.

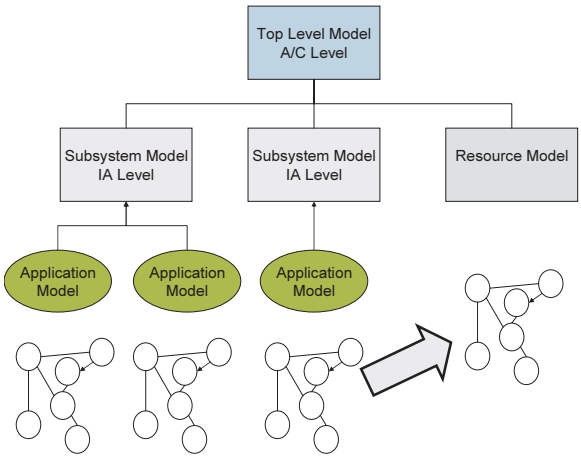


FIG 9. Parts of the overall system model

The Resource model is integrated as part of the top-level model. Application models are integrated into the subsystem / Integration Area models in accordance with the functional specifications.

In the next step processes are allocated to processors and Virtual Channels are allocated to the appropriate Transfer Connections, i.e. the allocation of communication channels to data connections of processors. The allocation of processes is provided by standard AADL properties. The allocation of Virtual Channels is performed by means of a user-defined property specifying the transfer connection (data connection), a VC is mapped to. This mapping has only to be performed for communication across processor borders.

Finally the allocation has to be verified in terms of consistency of the resource consumption by the allocated components with the resources provided by the components of the resource model, like processors, memory and network connections.. This may be an iterative process, which has to be repeated until consumption matches the actual resources.

6. EXPLOITATION OF THE SYSTEM MODEL

After this has been done a complete and consistent AADL model of the system is available. As AADL has a standardised XML representation. This model can further be exploited within the system development process for:

– Verification of the system model,
– Analysis of system properties, for instance end-to-end data latencies, or
– Generation of Configuration tables for LRUs.

Especially the generation of configuration tables is a very important application, because the reference for these configuration, whether applied for ARINC 653 processor configuration or ASAAC / MoAA Run-Time Blueprints, is a single, consistent System Model.

7. CONCLUSION

This article shows how AADL can be used for the modelling of an avionics system, which is based on IMA standards such as the ASAAC / MoAA or ARINC 653.

This shows that AADL has the power to express the structure of a complete avionics system within a single model. This itself is already a very important step in managing the complexity of an avionics system, as such a model in excess of the syntax and semantic requirements of AADL itself, can be queried for the verification of completeness, correctness and consistency of the model. Especially the need to generate correct and consistent configuration data will push the application of AADL in the system development process.

8. REFERENCES

[1] Avionics Application Software Standard Interface ARINC Specification 653-1, published by AERONAUTICAL RADIO, INC, October 16, 2003
[2] Modular and open Avionics Architectures, PART II: Software
STANAG 4626 (DRAFT 1), North Atlantic Treaty Organisation
[3] Society of Automotive Engineers (SAE) International:

Architecture Analysis & Design Language (AADL) SAE AS5506, Nov. 2004
[4] AADL web site des Software Engineering Institute http://www.aadl.info
[5] TOPCASED web site http://www.topcased.org