AIL: description of a global electronic architecture at the vehicle scale

Arjun Panday – Damien Couderc – Simon Marichalar

Abstract

This paper introduces the Architecture Implementation Language; a description language that allows for an internal representation of the architecture and acts as a connection with tools to simplify the construction, planning, verification, capitalisation, and documentation of an architecture. The objective of AIL is to describe a vehicle architecture from the level of the desired services down to the level of physical implementation, rendered concrete in one or more resulting operational architectures. The proposed methodology introduces the concepts of high level component based architectures to the highly constrained automotive world.

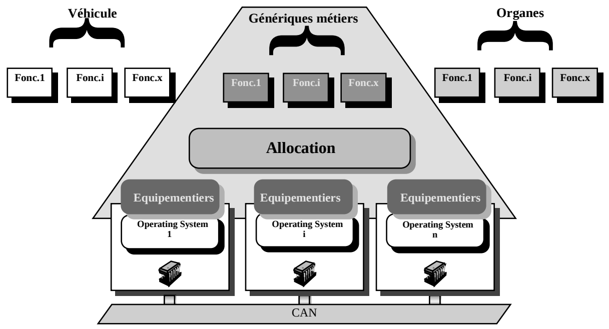
Introduction

The Architecture Implementation Language intends to describe the complete electronic architecture of a vehicle, taking into account all inter dependencies. Being firstly a conception tool, it distinguishes between three levels of description :

- the **functional architecture** describes all vehicle level functions and features perceived by the customer and decomposes them into smaller sub functions from a purely functional point of view,
- the **software architecture** describes the software implementation of the functions defined in the functional architecture. These implementations are developed upon a virtual environment giving access to all vehicle services and hardware devices,
- the **hardware architecture** describes the physical placement of hardware ECUs on one or more networks, the bridges and gateways between networks, as well as low level device and network drivers.

The functional architecture allows designers to describe, without any hardware or implementation related constraints, the whole set of high level features and services at the vehicle scale and their interactions in the case of complex functions (ESP + ABS + navigation info…). This description allows a fine grained decomposition of the services into sub functions, eventually reaching elementary functional modules for which well defined interfaces are published (brake pressure regulation, injection control, vehicle positioning…). The set of these elementary modules forms the virtual environment the software architecture relies upon.

A fourth level of description results of the marriage of the software architecture with the hardware architecture (both designed independently with their respective constraints). The **operational architecture** thus obtained describes the distribution of the software modules amongst the various ECUs. Elementary components can be placed close to their source of information whereas composite services can be placed where room is available and possibly distributed amongst various ECUs.



Definition of the software components

AS Component :

An Application Software Component is a reusable software component used in the realisation of the Elementary Functions in association with other ASC and I/O BSC.
The functional content of the ASC component and its interfaces with the other components, ASC and I/O BSC, are standardised.

This software component is the smallest entity of the reused AIL Architecture. It is a coherent and independent object, which must be portable on several ECU.

It must be developed in a portage language which preserves the independence with the hardware. The

ASC interface can communicate with an other ASC or BSCs by call of ICEM services.

I/O BS Component

Basic Software Component does the conversion from the raw data coming out of the drivers to the format needed by the application. It is usually attached to one particular physical component, sensor or actuator.
An I/O BSC is a software entity, depending of the hardware and written in a portage language. I/O BSCs communicate with the Interface Component Exchange Manager (ICEM) by exchange of software data or directly with the ASC by call of libraries services and with Driver BSC.

COM BS Component
This component manages the communication services for the transfer of application messages between several ECUs. The ICEM communicate with the Driver COM BSC through this layer, using application program interface of OSEK COM.
This software depends of hardware features.

Driver COM BSC
This component provides services for the communication over the physical layer.
Each COM Driver Component manage the communication on its own network :
- between ECUs
- between ECUs and a multiplexed peripheral.

COM Driver depends of hardware features.

Driver BSC
A driver component is a basic software in charge of the management of a peripheral device.
The generic peripherals are :
- Input Output functions such as Analog input, Digital I/O, Pulse Width Modulation or serial interface
- Specific device : SPI for EEPROM, Flash Programming or ASIC and so on

Driver BSC depends of hardware features.

Library BSC
The Library BSC is a set of standard services or routines to do arithmetic and mathematical operations, available on a given platform.

There are two kinds of libraries :
- written in a portage language : independent of the hardware.
- written for a particular target : dependent of the hardware but with a standard interface

The libraries are only called by ASC and BSC components but not by driver components.

OS BSC
The Operating System BSC provides an execution support for the different components of the architecture.
It allows :
- tasks management and scheduling
- synchronisation mechanism such as resource or event management.
- monitoring

The OSEK OS defines a standard for an architecture of distributed embedded control units.

ICEM BSC
This component provides a generic software layer suitable for all hardware architectures. It delivers and also updates data to the application and the receive entities.
The ICEM allows :
- broadcasting of sharing software data.
- communication of messages and synchronisation between components.
- to ensure data consistency

Software Data

A Software Data is an information used and shared by different component ASC and I/O BSC. The ICEM BSC allows the use of this data.
A Vehicle Data is a set of Software Data.

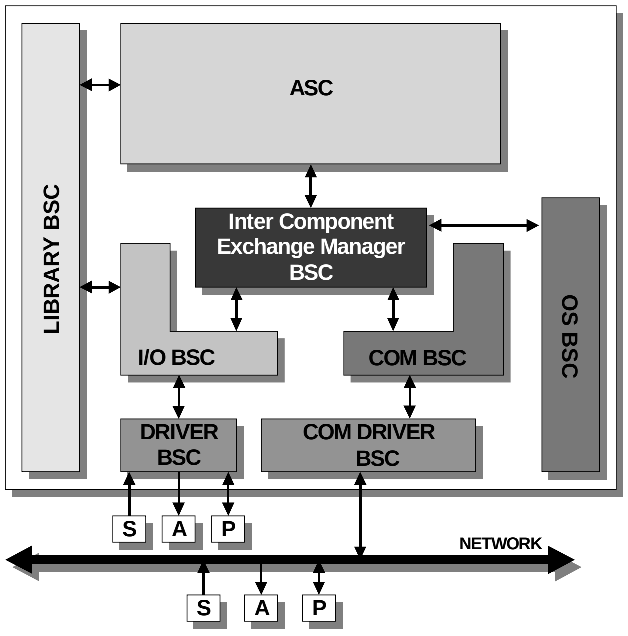
ICEM : a middle ware layer configured for a given architecture

At this stage, the components are distributed according to a specific architecture and able to communicate via a middle ware layer that represent all services available on the vehicle. Unique visible interface for the application components, the ICEM (Inter Component Exchange Manager) is in charge for dispatching the application modules messages to the appropriate sensor, device, operating system or application component.

We understand however that a classical request broker solution is not acceptable in the world of real time embedded systems. In classical models a request broker is either on each platform and therefore processing power and memory intensive, or on a dedicated platform and therefore requires numerous network transactions and introduces important delays. Because we cannot afford to cope with these issues an alternative solution was adopted.

Unlike constantly changing environments in the Internet world, embedded systems have well defined boundaries and a fairly static organisation. Because all dependencies are well described at design time, automatic configuration and optimisation is possible. Once the architecture for a given vehicle is established, fully described and physically placed, the compilation chain for each ECU freezes application software and middleware layer into a statically bound system allowing only the transactions needed by those components present on the ECU.

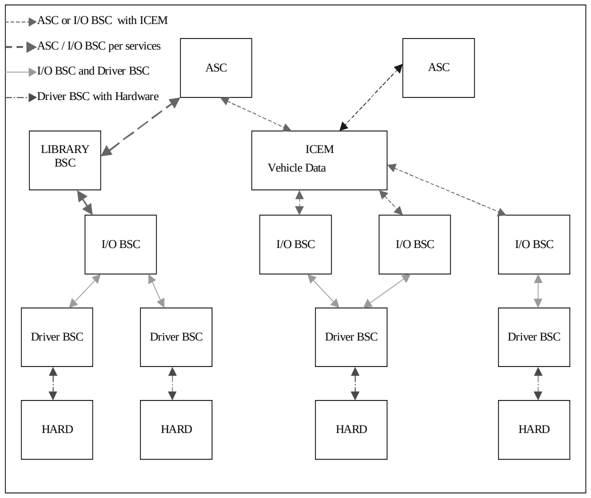
Static Architecture :



A: Actuator
S : Sensor
P : Peripheral access write/read

Sample of communication between components

This second diagram shows the possible hierarchy between the different components of the architecture.



Conclusion

We have described the Architecture Implementation Language and presented the structure of its main components. This framework will :

- ensure the portability of certain elements making up an application,
- ensure the reuse of certain elements making up an application,
- ensure the validation of the architecture during the different phases of the development process.

The formalisms, methods, and tools associated with AIL will allow the creation of an operational architecture validated a priori and implemented in an optimised manner.

References

AEE project deliverables – PSA, Renault, Sagem, Siemens France, Valeo.

“Automotive electronics : trends and challenges” – Alberto Sangiovanni-Vincentelli – University of Berkeley California

“What’s ahead for embedded software ?” – Edward A. Lee – University of Berkeley California

“Vehicle electrical/electronic system design considerations” – Gary Rushton, Viren Merchant – Auto Neural Systems