Design and validation process of in-vehicle embedded electronic systems

Françoise Simonot-Lion, Ye-Qiong Song

Design and validation process of in-vehicle embedded electronic systems

Françoise Simonot-Lion
LORIA UMR 7503 – Institut National Polytechnique de Lorraine
Campus Scientifique - BP 239 - 54506 VANDOEUVRE-lès-NANCY CEDEX
Fax: +33 3 83 27 83 19
Phone : +33 3 83 58 17 62
simonot@loria.fr

YeQiong Song
LORIA UMR 7503 – Université Henri Poincaré Nancy I
Campus Scientifique - BP 239 - 54506 VANDOEUVRE-lès-NANCY CEDEX
Fax: +33 3 83 58 17 01
Phone : +33 3 83 58 17 64
song@loria.fr

CONTENTS

1 In-vehicle embedded applications: characteristics and specific constraints 2
1.1 Economic and social context 2
1.2 Several domains and specific problems 3
1.2.1 Power train 3
1.2.2 Chassis 4
1.2.3 Body 5
1.2.4 Telematic and HMI (Human Machine Interface) 5
1.3 Automotive technological standards 6
1.3.1 Networks and protocols 6
1.3.2 Operating Systems 11
1.4 A cooperative development process 12
2 Abstraction levels for in-vehicle embedded system description 13
2.1 Architecture Description Languages 13
2.2 EAST-ADL for in-vehicle embedded system modelling 14
3 Validation and verification techniques 15
3.1 General view of validation techniques 16
3.2 Validation by performance evaluation 17
3.2.1 Case study 18
3.2.2 Simulation approach 24
3.2.3 Deterministic timing analysis 27
3.2.4 Comments on results 29
3.2.5 Automatic generation of models for simulation purpose 29
4 Conclusions and future trends 31
5 Appendix: In-vehicle electronic system development projects 32
6 References 34

1 In-vehicle embedded applications: characteristics and specific constraints

1.1 Economic and social context

While automobile production is likely to increase slowly in the coming years (42 millions cars produced in 1999 and only 60 millions planned in 2010), the part of embedded electronics and more precisely embedded software is growing. The cost of electronic systems was $37bn in 1995 and $60bn in 2000, with an annual growth rate of 10%. In 2006, the electronic embedded system will represent at least 25% of the total cost of a car and more than 35% for a high-end model [1].

The reasons for this evolution are technological as well as economical. On the one hand, the cost of hardware components is decreasing while their performance and reliability are increasing. The emergence of automotive embedded networks such as LIN, CAN, TTP/C, FlexRay, MOST and IDB-1394 leads to a significant reduction of the wiring cost as well. On the other hand, software technology facilitates the introduction of new functions whose development would be costly or even not feasible if using only mechanical or hydraulic technology and allows therefore satisfying the end user requirements in terms of safety and comfort. Well known examples are Electronic Engine control, ABS, ESP, Active suspension, etc. In short, thanks to these technologies, the customers can buy a safe, efficient and personalised vehicle while the carmakers are able to master the differentiation of product variants and the innovation (analysts stated that more than 80% of innovation, and therefore of added value, will be obtained thanks to electronic systems [2]). Another new factor is emerging. A vehicle includes already some electronic equipments like hand free phones, audio / radio devices and navigation systems. For the passengers, a lot of entertainment devices, such as video equipments, and communication with outside world will be available in the very near future. Even if these kinds of applications have little to do with the vehicle's operation itself, they increase significantly the part of software embedded in a car.

Who is concerned by this evolution? First the *vehicle customer*, for which the requirements are on the one hand, the increase of performance, comfort, assistance for mobility efficiency (navigation), … and on the other hand, the reduction of vehicle fuel consumption and cost. Furthermore he requires a reliable embedded electronic system that ensures safety properties. Secondly, the stakeholders, *carmakers* and *suppliers*, who are interested in the reduction of time to market, development cost, production and maintenance cost. Finally this evolution has a strong *impact on the society*: legal restrictions on exhaust emission, protection of the natural resources and of the environment, …

The example of electronic systems formerly presented does not have to meet the same level of dependability. So their designs are relevant of different techniques. Nevertheless, common characteristics are their distributed nature and the fact that they have to provide a level of quality of service fixed by the market, the safety requirements, the cost requirements. Therefore their development and their production have to be based on a suitable methodology including their modelling, validation, optimisation and test.

1.2 Several domains and specific problems

In-vehicle embedded systems are usually classified in four domains that correspond to different functionalities, constraints and models [3], [4]. Two of them are concerned specifically with safety: “power train” and “chassis” domain. The third one, “body”, is emerging and presently integrated in major of cars. And finally, “telematic”, “multimedia” and “Human Machine Interface” domains take benefit of continuous progress in the field of multimedia, wireless communications and Internet.

1.2.1 Power train

This domain represents the system that controls the motor according to, on the one hand, requests of the driver, that can be explicit orders (speeding up, slowing down, …) or implicit constraints (driving facilities, driving comfort, fuel consumption, …) and, on the other hand, environmental constraints (exhaust pollution, noise, …). Moreover, this control has to take into account requirements from other parts of the embedded system as climate control or ESP (Electronic Stability Program).

In this domain, the main characteristics are:

- at a functional point of view: the power train control takes into account different working modes of the motor (slow running, partial load, full load, …); this correspond to different and complex control laws (multi-variables) with different sampling periods (classical sampling periods for signals provided by other systems are 1ms, 2ms or 5ms while the sampling of signals on the motor itself is in phase with the motor times,
- at a hardware point of view: this domain requires sensors whose specification has to consider the minimisation of the criteria “cost / resolution”, and micro-controllers providing high computation power, thanks to their multiprocessors architecture and dedicated coprocessors (floating point computations), and high storage capacities,

- at an implementation point of view: the specified function are implemented as several tasks with different activation rules according to the sampling rules, stringent time constraints imposed to task scheduling, mastering safe communications with other systems and with local sensors / actuators.

In this domain, systems are relevant of continuous systems, sampled systems and discrete systems. Traditional tools for their functional design and modelling are, for example, Matlab / Simulink, Matlab / Stateflow. Currently the validations of these systems are mainly done by simulation and, for their integration, by emulation methods and / or test. Last, as illustrated formerly, the power train domain includes hard real time systems; so performance evaluation and timing analysis activities have to be proceeded on their implementation models.

1.2.2 Chassis

Chassis domain gathers all the systems that control the interaction of the vehicle with the road and the chassis components (wheel, suspension, …) according to the request of the driver (steering, braking or speed up orders), the road profile, the environmental conditions (wind, …). These systems have to ensure the comfort of driver and passengers (suspension) as well as their safety. This domains includes systems as ABS (Anti-lock Braking System), ESP (Electronic Stability Program), ASC (Automatic Stability Control), 4WD (4 Wheel Drive), … Note that chassis is the critical domain contributing to the safety of the passengers and of the vehicle itself. Furthermore, X-by-Wire technology, currently applied in avionic systems, is emerging in automotive industry. X-by-Wire is a generic term used when mechanical and / or hydraulic systems are replaced by “electronic” ones (intelligent devices, networks, computers supporting software components that implement filtering, control, diagnosis, … functionalities). For example, we can cite brake-by-wire, steer-by-wire, that will be shortly integrated in cars for the implementation of critical and safety relevant functions. The characteristics of the chassis domain and the underlying models are similar to those presented for power train domain, that is multivariable control laws, different sampling periods and stringent time constraints. Regarding the power train domain, systems controlling chassis components are fully distributed. Therefore the development of such systems must define an feasible system, i.e. satisfying performance, dependability and safety constraints. Conventional mechanical and hydraulic systems have stood the test of time and have proved to be reliable; it is not the same for critical software based systems. In aerospace / avionic industries, X-by-Wire technology is currently employed; but, for ensuring safety properties, are used specific hardware and software components, specific fault tolerant solutions (heavy and costly redundancies of networks, sensors and computers) and certified design and validation methods. Now there is a challenge to adapt these solutions to

automotive industries that impose stringent constraints on component cost, electronic architecture cost (minimisation of redundancies) and development time length.

1.2.3 Body

Wipers, lights, doors, windows, seats, mirrors are more and more controlled by software based systems. This kind of functions makes up the body domain. They are not subject to stringent performance constraints but however involve globally many communications between them and consequently a complex distributed architecture. There is an emergence of the notion of sub-system or sub-cluster based on low cost sensor-actuator level networks as, for example, LIN that connect modules realized as integrated mechatronic systems. On another side, the body domain integrates a central subsystem, termed the "central body electronic" whose main functionality is to ensure message transfers between different systems or domains. This system is recognized to be a central critical entity.

Body domain implies mainly discrete event applications. Their design and validation rely on state transition models (as SDL, Statecharts, UML state transition diagrams, Synchronous models). These models allow, mainly by simulation, the validation of a functional specification. Their implementation implies a distribution over complex hierarchical hardware architecture. High computation power for the central body electronic entity, fault tolerance and reliability properties are imposed to the body domain systems. A challenge in this context is first to be able to develop exhaustive analysis of state transition diagrams and second, to ensure that the implementation respects the fault tolerance and safety constraints. The problem here is to achieve a good balance between time triggered approach and flexibility.

1.2.4 Telematic and HMI (Human Machine Interface)

Next generation of telematic devices provides new sophisticated Human Machine Interfaces to the driver and the other occupants of a vehicle. They enable not only to communicate with other systems inside the vehicle but also to exchange information with the external world. Such devices will be in the future upgradeable and for this domain, a "plug and play" approach has to be favoured. These applications have to be portable and the services furnished by the platform (operating system and / or middleware) have to offer generic interfaces and downloading facilities. The main challenge here is to preserve the security of the information from, to or inside the vehicle. Sizing and validation do not relies on the same methods than for the other domains. Here we shift from considering messages, tasks and deadline constraints to fluid data streams, bandwidth sharing and multimedia quality of service and from safety and

hard-real time constraints to security on information and soft real time constraints. Note that if this domain is more related to entertainment activities, some interactions exist with other domains. For example, the telematic framework offers a support for future remote diagnostic services. In particular, the standard OBD-3, currently under development, extends OBD-2 (Enhanced On Board Diagnosis) by adding telemetry. As its predecessor, it defines the protocol for collecting measures on the power train physical equipments and alerting, if necessary, the driver and a protocol for the exchanges with a scan tool. Thanks to a technology similar to that which is already being used for automatic electronic toll collection systems, an OBD-3-equipped vehicle would be able to report the vehicle identification number and any emissions problems directly to a regulatory agency.

1.3 Automotive technological standards

A way for ensuring some level of interoperability between components developed by different partners is brought at first by the standardisation of services sharing the hardware resources between the application processes. For this reason, in the current section, we provide an outline of the main standards used in automotive industry, in particular the networks and their protocols and the operating systems. Then, we introduce some works in progress for the definition of a middleware that will be a solution for portability and flexibility purpose.

1.3.1 Networks and protocols

Due to the stringent cost, real-time and reliability constraints, specific communication protocols and networks have been developed to fulfil the needs of the ECU (Electronic Control Unit) multiplexing. SAE has defined three distinct protocol classes named class A, B and C. Class A protocol is defined for interconnecting actuators and sensors with a low bit rate (about 10Kbps). An example is LIN. Class B protocol supports a data rate as high as 100Kbps and is designed for supporting non real-time control and inter ECU communication. J1850 and low speed CAN are examples of SAE class B protocol. Class C protocol is designed for supporting real-time and critical applications. Networks like high speed CAN, TTP/C belong to the class C, which support data rates as high as 1 or several mega bit per second. This section intends to outline the most known of them.

1.3.1.1 CAN

CAN (Controller Area Network) [5], [6] is without any doubt the mostly used in-vehicle network. CAN is initially designed by “Robert Bosch” company at the beginning of the 1980’s for multiplexing the increasing number of

ECUs in a car. It has become an OSI standard in 1994 and is now a de facto standard for data transmission in automotive applications due to its low cost, robustness and bounded communication delay. CAN is mainly used in power train, chassis and body domains. Further information on CAN related protocols and development, including TTCAN, can be found in http://can-cia.org/.

CAN is a priority-based bus which allows to provide a bounded communication delay for each message priority. The MAC (Medium Access Control) protocol of CAN uses CSMA with bit by bit non-destructive arbitration over the ID field (Identifier). The identifier is coded using 11 bits (*CAN2.0A*) or 29 bits (*CAN2.0B*) and it also serves as priority. Up to 8 bytes of data can be carried by one CAN frame and a CRC of 16 bits is used for transmission error detection. CAN uses a NRZ bit encoding scheme for making feasible the bit by bit arbitration with a logical AND operation. However the use of bit-wise arbitration scheme intrinsically limits the bit rate of CAN as the bit time must be long enough to cover the propagation delay on the whole network. A maximum of 1Mbps is specified to a CAN bus not exceeding 40m.

The maximum message transmission time should include the worst-case bit stuffing number. This length is given by:

$$C_i = \left(44 + 8 \cdot DLC + \left\lfloor \frac{34 + 8 \cdot DLC}{4} \right\rfloor \right) \cdot \tau_{bit} \quad (1)$$

where DLC is the data length in bytes and τ_{bit} the bit time; the fraction represents the overhead due to the *bit stuffing*, a technique implemented by CAN for bit synchronization which consists in inserting an opposite bit every time five consecutive bits of the same polarity are encountered.

Frame format is given in Table 1. We will not here detail field signification; note however that the *Inter Frame Space* (**IFS**) has to be considered when calculating the bus occupation time of a CAN message.

1.3.1.2 VAN

VAN (Vehicle Area Network) [7], [8] is quite similar to CAN. It was used by the French carmaker PSA Peugeot-Citroën for the body domain. Although VAN has some more interesting technical features than CAN, it is not largely adopted by the market and has now been abandoned in favour of CAN. Its MAC (Medium Access Control) protocol is also CSMA with bit by bit non-destructive arbitration over the ID field (Identifier), coded with 12 bits. Up to 28 bytes of data can be carried by one VAN frame and a CRC of 15 bits is used. The bit rate can reach 1 Mbps. One of the main differences between CAN and VAN is that CAN uses NRZ code while VAN uses a so-called E-Manchester (Enhanced Manchester) code: a binary sequence is divided into blocks of 4 bits and the first three bits are encoded

using NRZ code (whose duration is defined as one Time Slot per bit) while the fourth one is encoded using Manchester code (two Time Slots per bit). It means that 4 bits of data are encoded using 5 Time Slots (TS). Thanks to E-Manchester coding, VAN, unlike CAN, doesn't need bit stuffing for bit synchronisation. This coding is sometimes denoted by 4B/5B.

The format of VAN frame is given in Table 1. The calculation of the transmission duration (or equivalent frame length) of a VAN frame is given by:

$$C_i = (60 + 10 \cdot DLC) \cdot TS \quad (2)$$

Note however that the *Inter Frame Gap* (**IFG**), fixed to 4 TS, has to be considered when calculating the total bus occupation time of a VAN message. Finally, VAN has one feature which is not present in CAN: the in-frame response capability. The same single frame can include the remote message request of the consumer (identifier and command fields) and the immediate response of the producer (data and CRC fields).

Table 1

CAN AND VAN FRAME FORMAT

		SOF	ID	RTR	(reserved)	DLC	Data	CRC	ACK	EOF	IFS
CAN	bit	1	11/29	1	2	4	0-64	16	2	7	3

		SOF	ID	Command	Data	CRC	EOD	ACK	EOF	IFG
VAN	bit	---	12	4	(0-28) · 8	15	--	--	--	--
	time slot	10	15	5	(0-28) · 10	15+3	2	2	8	4

1.3.1.3 J1850

SAE J1850 [9] is developed in north America and has been used by carmakers such as Ford, GM and DaimlerChrysler. The MAC protocol follows the same principle than CAN and VAN, i.e. it uses CSMA with bit by bit arbitration for collision resolution. J1850 supports two data rates: 41.6Kbps for PWM (Pulse Width Modulation) and 10.4Kbps for VPW (Variable Pulse Width). The maximum data length is 11 bytes. The typical applications are SAE class B ones such as instrumentation/diagnostics and data sharing in engine, transmission, ABS.

1.3.1.4 TTP/C

TTP/C (Time Triggered Protocol) [10] has been developed at Vienna University of Technology. Hardware implementations of the TTP/C protocol, as well as software tools for the design of the application, are commercialized by TTTech (www.tttech.com).

At the MAC layer, the TTP/C protocol implements the synchronous TDMA scheme: the stations (or nodes) have access to the bus in a strict deterministic sequential order. Each station possesses the bus for a constant time duration called a slot during which it has to transmit one frame. The sequence of slots such that all stations have access once to the bus is called a TDMA round.

TTP/C is suitable for SAE class C applications with strong emphasize on fault tolerant and deterministic real-time feature. It is now one of the two candidates for X-by-Wire applications. The bit rate is not limited in TTP/C specification. The today's available controllers (TTP/C C2 chips) support data rates as high as 5 Mbps in asynchronous mode and 5 to 25 Mbps in synchronous mode.

1.3.1.5 FlexRay

The FlexRay protocol (www.flexray.com) is currently being developed by a consortium of major companies from the automotive field. The purpose of FlexRay is, as like as TTP/C, to provide for X-by-Wire applications with deterministic real-time and reliability communication. The specification of the FlexRay protocol is however not yet publicly available nor finalized at the time of writing of this chapter.

The FlexRay network is very flexible with regard to topology and transmission support redundancy. It can be configured as a bus, a star or multi-stars and it is not mandatory that each station possesses replicated channels even though it should be the case for X-by-Wire applications.

At the MAC level, FlexRay defines a communication cycle as the concatenation of a time triggered (or static) window and an event triggered (or dynamic) window. To each communication window, whose size is set at design time, a different protocol applies. The communication cycles are executed periodically. The time triggered window uses a TDMA protocol. In the event triggered part of the communication cycle, the protocol is FTDMA (Flexible Time Division Multiple Access): the time is divided into so called mini-slots, each station possesses a given number of mini-slots (not necessarily consecutive) and it can start the transmission of a frame inside each of its own mini-slot. A mini-slot remains idle if the station has nothing to transmit.

1.3.1.6 LIN

LIN (Local Interconnect Network) (www.lin-subbus.org) is a low cost serial communication system intended to be used for SAE class A applications, where the use of other automotive multiplex networks such as CAN is too expensive. Typical applications are in body domain for controlling door, window, seat, proof and climate.

Besides the cost consideration, LIN is also a sub network solution to reduce the total traffic load on the main network (CAN for example) by building a hierarchical multiplex system. For this purpose, many gateways exist allowing for example to interconnect a LIN subnet to CAN.

The protocol of LIN is based on the master/slave model. A slave node must wait for being polled by the master to transmit data. The data length can be 1/2/4/8 bytes. A master can handle at most 15 slaves (there are 16 identifiers by class of data length). LIN supports data rates up to 20Kbps (limited for EMI-reasons).

1.3.1.7 MOST

MOST (Media Oriented System Transport) (http://mostnet.de/) is a multimedia fibre optic network developed in 1998 by MOST coorperation (a kind of consortium composed of carmakers, set makers, system architects and key component suppliers). The basic application blocks supported by MOST are audio and video transfer, based on which end-user applications like radios, GPS navigation, video displays and amplifiers and entertainment systems can be built.

The MOST protocol defines data channels and control channels. The control channels are used to set up what data channels the sender and receiver use. Once the connection is established, data can flow continuously for delivering streaming data (Audio/Video). The MOST network proposes a data rate of 24.8 Mbps.

1.3.1.8 IDB-1394

IDB-1394 is an automotive version of IEEE-1394 for in-vehicle multimedia and telematic applications jointly developed by the IDB Forum (www.idbforum.org) and the 1394 Trade Association (www.1394ta.org). IDB-1394 defines a system architecture/topology that permits existing IEEE-1394 consumer electronics devices to interoperate with embedded automotive grade devices. The system topology consists of an automotive grade embedded Plastic Optical Fiber network including cable and connectors, Embedded Network Devices, one or more Consumer Convenience Port interfaces, and the ability to attach hot-pluggable portable devices.

The IDB-1394 Embedded Network will support data rates of 100Mbps, 200Mbps, and 400Mbps. The maximum number of Embedded Devices is limited to 63 nodes.

From both data rate and interoperability with existing IEEE-1394 consumer electronic devices point of view, IDB-1394 is a serious competitor of the MOST technology.

1.3.2 Operating Systems

OSEK/VDX (Offene Systeme und deren schnittstellen für die Elektronik im Kraft-fahrzeug) [11] is a multi-task operating system that becomes a standard in European automotive industry. Two types of task are supported by OSEK/VDX, basic tasks without blocking point and extended tasks that can include blocking points. This Operating System does not allow the dynamic creation / destruction of tasks. It implements a Fixed Priority (FP) scheduling policy combined with Priority Ceiling Protocol (PCP) [12] to avoid priority inversion or deadlock due to exclusive resource access. OSEK/VDX offers a synchronisation mechanism through private events and alarms. A task can be pre-emptive or non pre-emptive. An implementation of OSEK/VDX has to be compliant to one of the four conformance classes –BCC1, BCC2, ECC1, ECC2- defined according to the supported tasks (basic only or basic and extended), the number of tasks on each priority level (only one or possibly several) and the limit of the reactivation counter (only one or possibly several). The MODISTARC project (Methods and tools for the validation of OSEK/VDX - based DISTributed ARChitectures) [13] aims to provide the relevant test methods and tools to assess the conformance of OSEK/VDX implementations. OSEK/VDX Com and OSEK/VDX NM are complementary to OSEK/VDX for communication and network management services. Furthermore, a language OSEK/OIL (OSEK Implementation Language) is a basis both for the configuration of an application and the tuning of the required operating system. In order to ensure dependability and fault tolerance for critical applications, the time-triggered operating system OSEKtime [11] was proposed. It supports static scheduling and offers interrupt handling, dispatching, system time and clock synchronisation, local message handling, and error detection mechanisms and offers predictability and dependability through fault detection and fault tolerance mechanisms. It is compatible to OSEK/VDX and is completed by FTCom layer (Fault Tolerant Communication) for communication services.

Rubus is another operating system tailored for automotive industry. It is developed by Arcticus systems [14], with support from the research community, and is, e.g., used by Volvo Construction Equipment. Rubus OS consists of three parts achieving an optimum solution: the Red Kernel, which manages execution of off-line scheduled time-

triggered tasks, the Blue Kernel dedicated for execution of event-triggered tasks and the Green Kernel in charge of the external interrupts. These three operating systems are well suited to power train, chassis and body domain because the number of tasks integrated in these applications is known off-line. On the other hand, they don't fit to the requirements of telematic applications. In this last domain, are available, for example Window CE for Automotive that extends the classical operating system Windows CE with telematic-oriented features.

Finally, an important issue for the multi-partners development and the flexibility requirement is the portability of software components. For this purpose, several projects aim to specify an embedded middleware, which has to hide the specific communication system (portability) and to support fault tolerance (see Titus Project [15], ITEA EAST EEA Project [24], DECOS Project [16] or Volcano [17]). Note that these projects as well as Rubus Concept [14] provide not only a middleware or an operating system but also a way for a Component Based Approach for designing a real time distributed embedded application.

1.4 A cooperative development process

Strong co-operation between suppliers and carmakers in the design process implies the development of a specific concurrent engineering approach. For example, in Europe or Japan, carmakers provide the specification of subsystems to suppliers, which are, then, in charge of the design and realization of these subsystems including the software and hardware components and possibly the mechanical or hydraulic parts. The results are furnished to the carmakers that have to integrate them on the car and to test them. The last step consists in calibration activities that is in tuning some control and regulation parameters for meeting the required performances of the controlled systems. This activity is closely related to testing activities. In USA, the process is slightly different, as the suppliers cannot be really considered as independent of carmakers. Nevertheless, the sub-system integration and calibration activities are always to be done and, obviously, any error detected during this integration leads to a costly feedback on the specification or design steps. So, in order to improve the quality of the development process, new design methodologies are emerging. In particular, the different actors of a system development apply more and more methods and techniques ensuring the correctness of subsystems as early as possible in the design stages and a new trend is to consider the integration of subsystems at a virtual level [18]. This means that carmakers as well as suppliers will be able to design, prove and validate the models of each subsystem and of their integration at each level of the development in a cooperative way. This new practice will reduce significantly the cost of the

development and production of new electronic embedded systems while increasing flexibility for the design of variants.

2 Abstraction levels for in-vehicle embedded system description

As shown in section 1.4, the way to improve the quality and the flexibility of an embedded electronic system while decreasing the development and production cost is to design and validate this system at a virtual level. So, the problem is, first, to identify the abstraction level at which the components and the whole system are to be represented. In order to apply validation and verification techniques on the models, the second problem consist in specifying which validation and verification activities have to be applied and, consequently, which formalisms support the identified models.

2.1 Architecture Description Languages

Two main key words were introduced formerly: architectures, that refer to the concept of Architecture Description Language (ADL), well known in computer science and, components, that leads to modularity principles and object approach. An ADL is a formal approach for software and system architecture specification [19]. In the avionic context for which the development of embedded systems is relating to the same problems, MetaH [20] has been developed at Honeywell and, in 2001, has been chosen as the basis of a standardization effort aiming to defining an Avionics Architecture Description Language (AADL) standard under the authority of SAE. This language can describe standard control and data flow mechanisms used in avionic systems, and important non-functional aspects such as timing requirements, fault and error behaviours, time and space partitioning, and safety and certification properties. In automotive industry, some recent efforts brought a solution for mastering the design, modelling and validation of in-vehicle electronic embedded systems. The first result was obtained by the French project AEE (Architecture embedded electronic) [21] and more specifically through the definition of AIL_Transport (Architecture Implementation Language for Transport). This language, based on UML, allows to specify in the same framework, electronic embedded architectures, from the highest level of abstraction, for the capture of requirements and the functional views, to the lowest level, for the modelling of an implementation taking into account services and performances of hardware supports and the distribution of software components [22], [23].

2.2 EAST-ADL for in-vehicle embedded system modelling

Taking AIL_Transport as one of the entry points for the European project ITEA EAST-EEA [24], (July 2001- June 2004), a new language named EAST-ADL was defined. As AIL_Transport, EAST-ADL offers a support for the unambiguous description of in-vehicle embedded electronic systems at each level of their development. It provides a framework for the modelling of such systems through 7 views (see Figure 1) [25]:

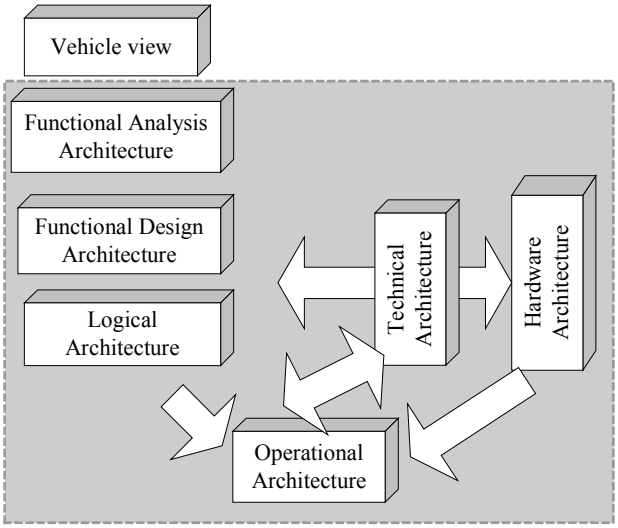


Figure 1 - The abstraction layers of the EAST-ADL.

- *Vehicle View* describing user visible features such as anti-lock braking or windscreen wipers.
- *Functional Analysis Architecture* level represents the functions realizing the features, their behaviour and their exchanges. There is an n-to-n mapping between Vehicle View entities and Functional Analysis Architecture entities, i.e. one or several functions may realize one or several features.
- *Functional Design Architecture* level models a decomposition or refinement of functions described at *Functional Analysis Architecture* level in order to meet constraints regarding allocation, efficiency, re-use, supplier concerns, etc. Again, there is an n-to-n mapping between entities on Functional Design Architecture and Functional Analysis Architecture.
- *Logical Architecture* level where the class representation of the Functional Design Architecture has been instantiated to a flat SW structure suitable for allocation. This level provides an abstraction of the software components to implement on an hardware architecture. The logical architecture contains the leaf functions

of the Functional Design Architecture. From the Logical Architecture point of view, the code could automatically generated in many cases.

In parallel to the application functionality, the execution environment is modelled from three views:

- The *Hardware Architecture* level includes the description of the ECUs (Electronic Component Units) and more precisely those of the used micro-controller, the sensors and actuators, the communication links (serial links, networks) and their connections.
- At *Technical Architecture* level, is given the model of the operating system or Middleware API and the services provided (behaviour of the middleware services, schedulers, frame packing and memory management, in particular).
- The *Operational Architecture* models the tasks, managed by the operating systems and frames, managed by the protocols. On this lowest abstraction level, all implementation details are captured.

A system described on the Functional Analysis level may be loosely coupled to hardware based on intuition, various known constraints or as a back annotation from more detailed analysis on lower levels. Furthermore, the structure of the Functional Design architecture and of the Logical Architecture is aware of the Technical architecture. Finally, this EAST-ADL provides the consistency within and between artefacts belonging to the different levels, at a syntactic and semantic point of view. This leads to make a EAST-ADL based model a strong and non ambiguous support for building automatically models suited to optimal configuration and / or validation and verification activities. For each of these identified objectives (simulation or formal analysis at functional level, optimal distribution, frame packing, round building for TDMA-based networks, formal test sequences generation, timing analysis, performance evaluation, dependability evaluation, …), a software, specific to the activity, to the related formalism and to the EAST-ADL, extracts the convenient data from the EAST-ADL repository and translates into the adequate formalism. Then, the concerned activity can run thanks to the adequate tools.

3 Validation and verification techniques

In this section, we introduce, briefly in 3.1, the validation issues in automotive industry and the place of these activities in the development process and detail in 3.2 a specific validation technique that aims to prove that an operational architecture meets its performance properties.

3.1 General view of validation techniques

The validation of an embedded system consists of proving, on the one hand, that this system implements all the required functionalities and, on the other hand, that it ensures functional and extra-functional properties such as performance, safety properties. From an industrial point of view, validation and verification activities address two complementary objectives:

- Validation and verification of all or parts of a system at a functional level without taking into account the implementation characteristics (e.g. hardware performance). For this purpose, simulation or formal analysis techniques can be used.
- Verification of properties of all or parts of a system at operational level. These activities integrate the performances of both the hardware and technical architectures and the load that is due to a given allocation of the logical Architecture. This objective can, also, be reached through simulation and formal analysis techniques. Furthermore, according to the level of guarantee required for the system under verification, a designer can need deterministic guarantees or simply probabilistic ones, involving different approaches.

The expression “formal analysis” is employed when mathematic techniques can be applied on an abstraction of the system while “simulation” represents the possibility to execute a virtual abstraction of it. Obviously, formal analysis leads to an exhaustive analysis of the system, more precisely of the model that abstracts it. It provides a precise and definitive verdict. Nevertheless the level of abstraction or the accuracy of a model is inverse ratio to its capacity to be treated in a bounded time. So this technique is generally not suitable to large systems at fine grain abstraction level as required, for example, for the verification of performance properties of a wide distributed operational architecture; in fact, in this case, the system is modelled thanks to timed automata or queuing systems whose complexity can make their analysis impossible. To solve this problem, simulation techniques can be applied. They accept models at almost any level of detail. But the drawback is that it is merely impossible to guarantee all the feasible execution can be simulated. So the pertinence of these results is linked to the scenario and the simulation duration and therefore we can only ensure that a system is correct for a set of scenarios but this doesn’t imply that the system will stay correct for any scenario. In fact, in automotive industry, simulation techniques are more largely used than formal analysis. An exception to this can be found in the context of the verification of properties to be respected by frames sharing a network. Well-known formal approach, usually named timing analysis are available for this purpose. Finally, note that some tools are of course of general interest for the design and validation of electronic embedded systems as, for

example Matlab / Simulink or Stateflow [26], Ascet [27], Statemate [28], SCADE [29]… In some cases, an interface encapsulates these tools in order to suit the tool to the automotive context.

Moreover, these techniques that work on virtual platforms are completed by test techniques in order to assume that a realisation is correct: test of software components, of logical architectures, test of an implemented embedded system. Note that the test activities as well as the simulation ones consist of providing a scenario of events and/or data that stimulate the system under test or stimulate an executable model of the system; then, in both techniques we have to look which events and/or data are produced by the system. The input scenario can be manually built or formally generated. In this last case the test or simulation activity is closely linked to a formal analysis technique [30].

At last, one of the main targets of validation and verification activities is the dependability aspect of the electronic embedded systems. As seen in the first section, some of these systems are said safety-critical. This fact is enhanced in chassis domain through the emergence of X-by-Wire applications. In this case, high dependability level is required: the system has to be compliant to a number of failures by hour less than 10^{-9} (this means that the system has to work 115 000 years without a failure). By now, it's a challenge because it's impossible to ensure this property only through the actual reliability of the electronic devices. Moreover, as the application may be sensitive to electromagnetic perturbations, its behaviour can be not entirely predictable. So the required safety properties can be reached by introducing fault tolerant strategies.

3.2 Validation by performance evaluation

The validation of a distributed embedded system requires, at least, proving that all the timing properties are respected. These properties are generally expressed as timing constraints applied to the occurrences of specific events, as, for example, a bounded jitter on a frame emission, a deadline on a task, a bounded end-to-end response time between two events, … The first way of doing this is analytically, but this means one should be able to establish a model that furnishes the temporal behaviour of the system and that can be mathematically analysed. Considering the complexity of an actual electronic embedded system, such a model has necessarily to be strongly simplified and generally provides only oversized solutions. For instance, the holistic scheduling approach introduced by K. Tindell and J. Clark [31] allows just the evaluation of the worst-case end-to-end response time for the periodic activities of a distributed embedded application. Using this holistic scheduling approach, Y.Q. Song et al., in [32], studied the end-to-end task response time for architecture composed of several Electronic Control Units, interconnected by CAN.

Face to the complexity of this mathematical approach, the simulation of a distributed application is therefore a complementary technique. It allows to take into account a more detailed model as well as the unavoidable perturbations that may affect the foreseen behaviour. For example, simulation-based analysis [33] of the system in [32] gave more realistic performance measures than those obtained Analytically in [32].

An outline on these two approaches is illustrated in sections 3.2.3 and 3.2.2 by the mean of a common case study presented in section 3.2.1 ; then, in paragraph 3.2.4, the respective obtained results are compared. Finally, we show how a formal architecture description language, as introduced in section 2, is a strong factor for promoting validation and verification on virtual platform in automotive industry.

3.2.1 Case study

Figure 2 shows the electronic embedded system [36], used in the two following sections as a basis for both mathematical and simulation approaches. In fact, this system is derived from an actual one presently embedded in a vehicle manufactured by PSA Peugeot-Citroën Automobile Co. [34]. It includes functions related to powertrain, chassis and body domains.

Hardware Architecture level (Figure 2)

- We consider 9 nodes (**ECU** or **E**lectronic **C**ontrol **U**nit) that are interconnected by means of one CAN and one VAN network. The naming of these nodes recall the global function that they support. Engine controller, AGB (Automatic Gear Box), ABS/VDC (Anti-lock Brake System / Vehicle Dynamic Control), WAS/DHC (Wheel Angle Sensor/Dynamic Headlamp Corrector), Suspension controller refer nodes connected on CAN, while X, Y and Z (named so for confidentiality reason) refer nodes connected on VAN. At last, the ISU (Intelligent Service Unit) node ensures the gateway function between CAN and VAN.
- The communication is supported by two networks: *CAN 2.0A* (bit rate equal to 250kbps) and VAN (*time slot rate* fixed to 62.5kTS/s).
- The different ECUs are connected to these networks by means of network controllers. For this case study are considered the *Intel 82527* CAN network controller (14 transmission buffers), the *Philips PCC1008T* VAN network controllers (one transmission buffer and one FIFO reception queue with two places) and the *MHS 29C461* VAN network controllers (handling up to 14 messages in parallel).

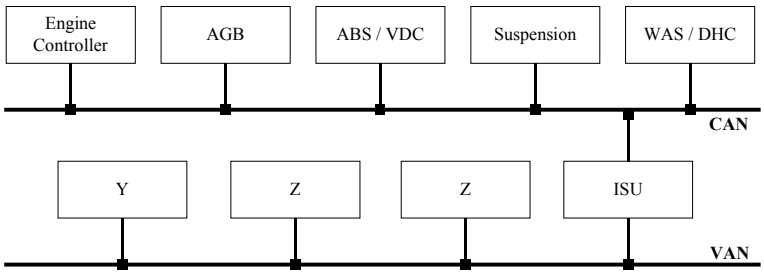


Figure 2 - Hardware Architecture.

Technical level

- The operating system OSEK [11] runs on each ECU. The scheduling policy is Fixed Priority Protocol. Each OS task is a *basic task*, at OSEK sense. In the actual embedded system, pre-emption is not permitted for tasks. In the presented study, analytical method is applied strictly to this system, while simulations are run for different configurations among which, two accept preemptible tasks.

Operational level

The entities that are considered at this level are **tasks** and **messages** (frames); they are summarized in Figure 3, Figure 4, Figure 5and Figure 7. The mapping of the Logical Architecture (not presented here) onto the Technical / Hardware ones produces 44 OSEK OS tasks (in short, *tasks*, in the following) and 19 messages exchanged between these tasks. Furthermore, we assume that a task OS consumes (resp. produces) possibly a message simultaneously to the beginning (resp. the end) of its execution. In the case study, two kinds of task can be identified according to their activation law:

- Tasks activated by occurrence of the event "reception of a message" (**event triggered tasks**) as, for example, *T_Engine6* and *T_ISU2*.
- Tasks that are activated periodically (**time triggered tasks**), as *T_AGB2*.

Each ***task*** is characterized by its name and, on the ECU, named *k*, on which it is mapped by (see Figure 3, Figure 4, Figure 5):

- T_i^k , its **activation period** in ms.(for time triggered tasks) or the name, M_n, of the **message** whose reception activates it (for event triggered tasks),

- C_i^k , its **WCET** (Worst Case Execution Time) on this ECU (disregarding possible pre-emption); in the case study, we assume that this WCET is equal to 2 ms for each task,
- D_i^k , its **relative deadline** in ms.,
- P_i^k , its **priority**
- M_i , its possibly produced message (we assume, in this case study, that at most, one message is produced by one task; note the method can be applied even if a task produces more than one message).

For notation convenience, we assume that, on each ECU named *k*, priority P_i^k is higher than priority P_{i+1}^k. In the following section, a task is simply denoted τ_i if its priority is P_i^k on an ECU, named *k*.

ECU: Engine_Ctrl

	P_i	T_i	*input*	*output*	D_i
T_Engine1	1	10		M1	10
T_Engine2	4	20		M3	20
T_Engine3	7	20		M10	100
T_Engine4	3		M4		15
T_Engine5	2		M2		14
T_Engine6	6		M8		50
T_Engine7	5		M6		40

ECU:AGB

	P_i	T_i	*input*	*output*	D_i
T_AGB1	2	15		M4	15
T_AGB2	3	50		M11	50
T_AGB3	4		M8		50
T_AGB4	1		M2		14

ECU: ABS/VDC

	P_i	T_i	*input*	*output*	D_i
T_ABS1	2	20		M5	20
T_ABS2	5	40		M6	40
T_ABS3	1	15		M7	15
T_ABS4	6	100		M12	100
T_ABS5	3		M3		20
T_ABS6	4		M9		20

ECU: Suspension

	P_i	T_i	*input*	*output*	D_i
T_SUS1	4	20		M9	20
T_SUS2	5		M5		20
T_SUS3	1		M1		10
T_SUS4	2		M2		14
T_SUS5	3		M7		15

ECU: WAS/DHC

	P_i	T_i	*input*	C_i	*output*	D_i
T_WAS1	1	14		2	M2	14
T_WAS2	2		M9	2		20

Figure 3 - OS Tasks on nodes connected to CAN

ECU: X

	P_i	T_i	*input*	*output*	D_i
T_X1	2	150		M16	150
T_X2	4	200		M17	200
T_X3	1		M15		50
T_X4	3		M19		150

ECU: Y

	P_i	T_i	*input*	*output*	D_i
T_Y1	2	50		M15	50
T_Y2	3		M13		50
T_Y3	1		M14		10
T_Y4	4		M18		100
T_Y5	5		M16		150

ECU: Z

	P_i	T_i	*input*	*output*	D_i
T_Z1	2	100		M18	100
T_Z2	3	150		M19	150
T_Z3	4		M17		200
T_Z4	1		M15		50

Figure 4 - OS Tasks on nodes connected to VAN

	ECU: ISU				
	P_i	T_i	*input*	*output*	D_i
T_ISU1	4	50		M8	50
T_ISU2	5		M11	M13	50
T_ISU3	1		M1	M14	10
T_ISU4	6		M10		100
T_ISU5	3		M6		40
T_ISU6	2		M9		20
T_ISU7	7		M12		100

Figure 5 - OS Tasks distributed on the gateway ECU

The *Task Response Time* is classically defined as the time interval between the activation of a given task and the end of its execution (Figure 6). We denoted R_i^j the *Task Response Time* of the instance *j* of a task τ_i.

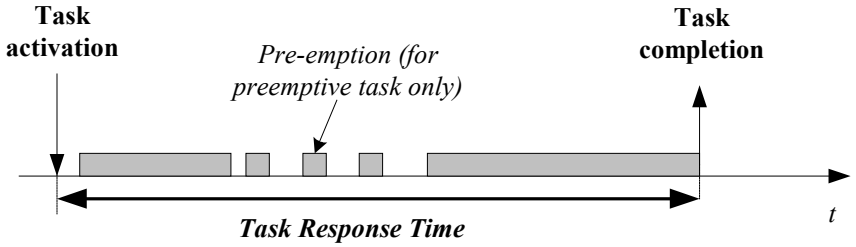


Figure 6 - Task Response Time

Each ***message*** (frame) is characterized by its name and (Figure 7):

- DLC_i, its **size**, in byte
- C_i, its transmission duration; ; this duration is computed thanks to the formulae given in (1) and (2) (see sections 1.3.1.1 and 1.3.1.2),
- $Task_i$ the name of the task that produces it,
- T_i, its **inherited period** (for time triggered tasks), assumed in [31] and [32], to be equal to the activation period of its producer task,
- P_i, its **priority**.

A message will also be denoted by τ_i if its priority is P_i.

τ_i Message (Frame)	$Task_i$ Producer Task	DLC_i (bytes)	T_i Inherited period
M1	T_Engine1	8	10
M2	T_WAS1	3	14
M3	T_Engine2	3	20
M4	T_AGB1	2	15
M5	T_ABS1	5	20
M6	T_ABS2	5	40
M7	T_ABS3	4	15
M8	T_ISU1	5	50
M9	T_SUS1	4	20
M10	T_Engine3	7	100
M11	T_AGB2	5	50
M12	T_ABS4	1	100

(a) – Messages on CAN

τ_i Message (Frame)	$Task_i$ Producer Task	DLC_i (bytes)	T_i Inherited period
M13	T_ISU2	8	50
M14	T_ISU3	10	10
M15	T_Y1	16	50
M16	T_X1	4	150
M17	T_X2	4	200
M18	T_Z1	20	100
M19	T_Z2	2	150

(b) – Messages on VAN

Figure 7 - Messages exchanged over networks

The *Message Response Time* is the time interval between the production of a specific message and its reception by a consumer task (Figure 8). We denoted R_i^j the *Message Response Time* of the instance *j* of a message τ_i.

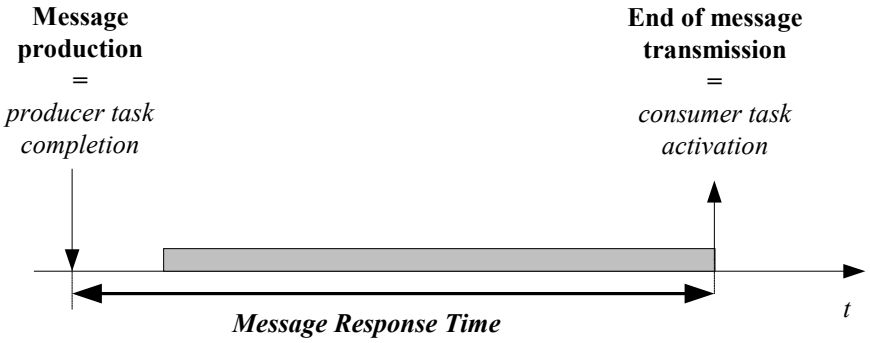


Figure 8 - Message response Time

Finally, in this system, from Figure 3, Figure 4, Figure 5 and Figure 7, we identify some "*logical chains*" that are causal sequences of tasks and messages. In the case study, the most complex logical chains that can be identified are:

lc1 : <T_Engine1 - M1 - T_ISU3 - M14 - T_Y3> and

lc2 : <T_AGB2 - M11 - T_ISU2 - M13 - T_Y2>.

Here, the task *T_Y2* (resp. *T_Y3)*, running on a VAN connected node, depends on the message M14 (resp. M13) supported by the VAN bus; M14 (resp. M13) is produced by task *T_ISU3* (resp. *T_ISU2*) running on the ISU node;

T_ISU3 (resp. *T_ISU2*) is activated by the message *M1* (resp. *M11*) that is produced by *T_Engine1* (resp. *T_AGB2) running* on CAN connected nodes.

The *Logical Chain Response Time*, more generally named *End-to-End Response Time*, is defined for *lc1* (resp. *lc2*) as the time interval between the activation of *T_Engine1* (resp. *T_AGB2)* and the completion of *T_Y2* (resp. *T_Y3)* (Figure 9). We note R_{lci}^{j} the *Logical Chain Response Time* of the instance *j* of the logical chain *lci*.

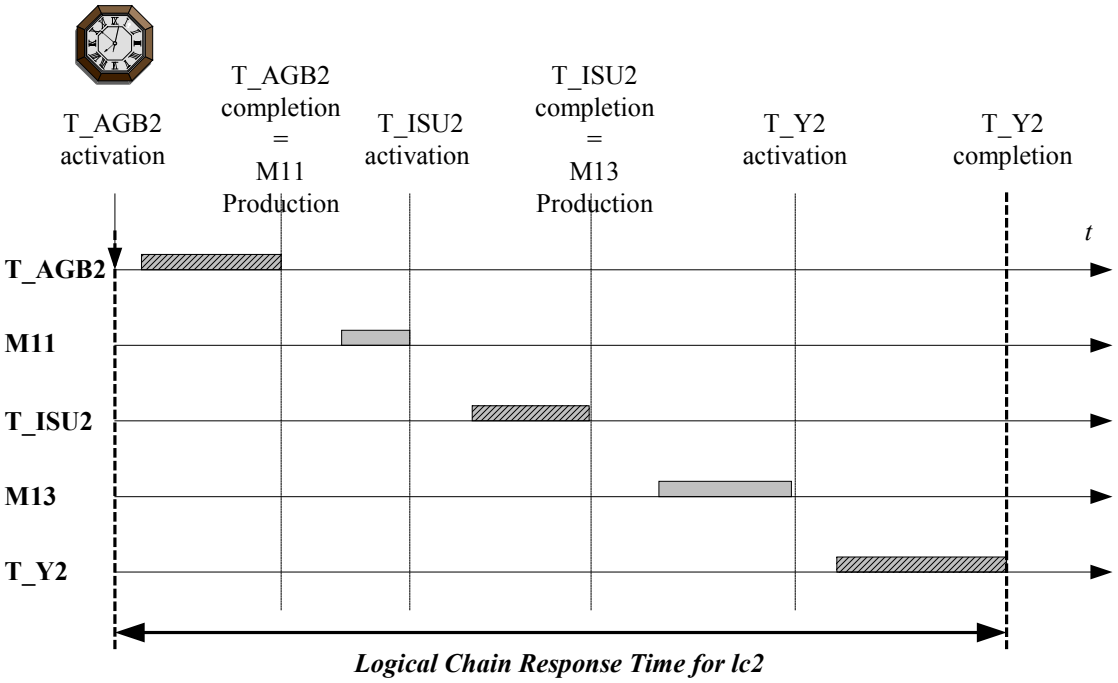


Figure 9 - Example of Logical Chain

Performance properties

As presented in Figure 3, Figure 4 and Figure 5, relative deadline constraints are imposed on each task in this application. Furthermore, for the given application, some other performance properties were required. Among these properties, we focus on two specific ones:

- **Property A**: no message, transmitted on CAN or VAN, is lost.

  This means that no message can be overwritten in network controller buffers or, more formally, that each message is transmitted before its inherited period T_i, considered as the worst case.

- **Property B**: this property is expressed on the two logical chain *lc1* and *lc2* presented above. The *Logical Chain Response Time* for *lc1* (resp. *lc2)* is as regular as possible for each instance of *lc1* (resp. *lc2*). More

formally, if R1 is the set of *Logical Chain Response Times* obtained for each instance *j* of *lc1* (resp. *lc2*), the property is: $\forall j, \left| R_{lc1}^{j} - E\left[\mathrm{R1}\right] \right|$.

This kind of property is commonly required in embedded automatic control applications where the command elaborated through a *Logical Chain* has to be applied to an actuator as regularly as possible.

An embedded system is correct if, at least, it meets the above-mentioned properties. Well, the task scheduling policy on each node and the MAC protocol of VAN and CAN lead unavoidably to **jitters** on task terminations. So, a mathematical approach as well as a simulation one were applied in order to prove that the proposed operational architecture meets all its constraints. Thanks to a mathematical approach, related to general techniques named *timing analysis*, for each entity (task or message) and for each logical chain, we find out lower and upper bounds of their respective response times. These values represent the best and worst cases. In order to handle more detailed and more realistic models, we use a simulation method, which furnishes minimum, maximum and mean values of the same response times. Furthermore, several simulations, with different parameter configuration, were performed in order to obtain an architecture meeting the constraints. In fact, we use the mathematical approach for validating the results obtained by simulation.

3.2.2 Simulation approach

We model four different configurations of the presented operational architecture according to the formalism supported by SES Workbench tool. For each configuration, we use this tool in order to run a simulation and obtain different results. Furthermore, as we want to analyse specific response times, we introduce adequate probes in the model. Thanks to this, the log file obtained throughout the simulation process can be easily analysed by applying an elementary filter that furnishes the results in a readable way.

Three kinds of parameters are considered and can be different from one configuration to another :

- networks controllers, specifically the VAN ones,
- the fact that tasks can be pre-empted or no,
- the task priority.

Rather than describing a simulation campaign that should include exhaustively each possible case among these possibilities, we prefer to present it by following an intuitive reasoning, starting from a given configuration

(configuration 1) and, by modifying one kind of parameter at a time, leads successively to better configuration (configuration 2, then configuration 3) for, finally, reaching a correct configuration that verifies the required properties A and B.

Configuration 1

As a first simulation attempt:

- As given in the description of the actual embedded system (see 3.2.1), all the tasks are considered as being OSEK basic tasks and are characterised by their local priority. Moreover, their execution is done without pre-emption
- We assign the *Intel 82527 controller* to each node connected to CAN bus and the *Philips PCC1008T* controller for those are connected on VAN network. Note that the ISU ECU integrates these two network controllers.

In this case, a probe is introduced in the model; it observes the occurrences of message production and of message transmission and detects the fact that a given instance of a message is stored in the buffer of a network controller before the instance of the previous produced message was transmitted through the network. For each of these detected events, it writes a specific string in the log file. The filter consists then in extracting the lines containing this string from the log file. A screenshot is given in Figure 10 where it can be seen that some messages are overwritten in the single transmission buffer of the VAN controller that was chosen for this configuration. So, we conclude that the property A is not verified.

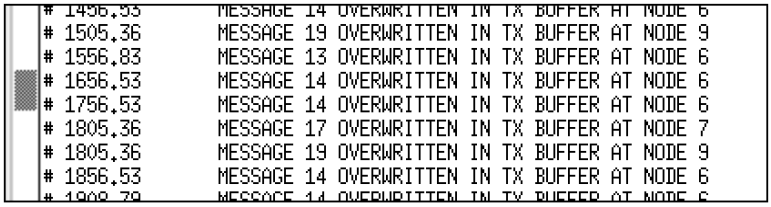


Figure 10 - Log file filtered for verification of property A

Configuration 2

One of the possible causes for the non verification of property A by the previous configuration could be that the VAN controller *PCC1008T*, providing only one single buffer, is not suitable for the required performance property. So, we assign the full VAN controller *MHS 29C461* to all nodes transmitting message on the VAN bus (*ISU computer*, *X*, *Y* and *Z*). We modify the SES Workbench model and we re-launch the simulation. This time, probes and filters proposed for the configuration 1 provides an empty list. So we can conclude that messages are correctly transmitted and that property A is verified. Furthermore SES Workbench gives additional results such as the network load; for this configuration the load of CAN bus is less than 21.5% and of VAN bus less than 41%.

On the same configuration, we study property B. For this purpose, probes are introduced for observing the occurrences of the first task activation (*T_Engine1* or *T_AGB2*) and the occurrences of the last task completion (*T_Y3* or *T_Y2*). A filter is developed for the evaluation of the minimum, mean and maximum *Logical Chain Response Times* of *lc1* and *lc2* as well as their standard deviation. The obtain results are given in Figure 11. Under this configuration, none of the chains meet the required property.

Configuration 3

In the two last configurations, pre-emption is not allowed for any task. We change this characteristic and allow the pre-emption; as *T_Engine1* and *T_AGB2* have the highest local priority and considering that they are basic tasks, they will never waiting for the processor. One more, we model the operational architecture by modifying the scheduling policy on nodes *Engine_ctlr* and *AGB* without changing the other parameters. The same probes and filters are used; the results obtained by simulation of configuration 3 are shown in Figure 11. We can conclude that property B is verified only for the logical chain *lc1*. So, this configuration doesn't correspond to a correct operational architecture.

Configuration 4

Further log file analysis points out the problem: priority of *T_ISU2* is probably too weak. After modifying the priority of this task (2 in place of 5), by using always the same probes and filters and simulating the new model, we obtain the results presented in Figure 11. The property B is verified for lc1 and lc2.

3.2.3 Deterministic timing analysis

In order to validate these results, we apply analytic formulas of [32] to the case study. The main purpose of this analysis is to obtain the lower (best-case) and the upper (worst-case) bounds on the response times. It is worth noting that in practice neither the best-case nor the worst-case can necessarily be achieved, but they provide some deterministic bounds.

As time-triggered design approach is adopted, both tasks and messages are hoped to be "periodic" although in practice jitters exist on event whose occurrences are supposed to be periodic. In the following, a non pre-emptive task or a message τ_i whose priority is P_i can be indifferently characterized by (C_i , T_i) as defined previously.

As introduced previously, we are interested in evaluating:

- the response time R_i of such a task or message of priority P_i,
- the Logical Chain Response Time of *lc1* and *lc2* obtained by summing these individual response times.

Best case evaluation

The best case corresponds to the situation where a task τ_i (resp. a message τ_i), whose priority is P_i, is executed (resp. transmitted) without any waiting time. In this case,

$$R_i = C_i \tag{3}$$

The best case of the *Logical Chain Response Time* is the sum of best-case response times of all entities (tasks and messages) involved in the chain. Applying it to the two logical chains, we obtain (see in Figure 11):

$$R\_best_{lcx} = \sum_{y \in lcx} C_y \tag{4}$$

Worst case evaluation

We distinguish the evaluation of the worst case for task and message response time.

Messages. For a message τ_i of priority P_i the worst-case response time can be calculated as:

$$R_i = C_i + I_i \tag{5}$$

where I_i is the interference period during which the transmission medium is occupied by other higher priority messages and by one lower priority message (because of non pre-emption). Take notice to the fact that the message

response time is defined here in way different from those specified by K. Tindell and A. Burns in [35], so, the jitter J_i is not include in the formulae (5).

The following recurrence relation can calculate the interference period I_i:

$$I_i^{n+1} = \max_{i+1 \le j \le N}(C_j) + \sum_{j=1}^{i-1}\left(\left\lfloor \frac{I_i^n + J_j}{T_j} \right\rfloor + 1 \right) C_j \quad (6)$$

where N is the number of messages and $\max_{i+1 \le j \le N}(C_j)$ is the blocking factor due to the non pre-emption. A suitable initial value could be $I_i^0 = 0$. Equation (6) converges to a value as long as the transmission medium's utilization is less than or equal to 100%. We also notice that the jitters should be taken into account for the calculation of the worst case interference period as the higher priority messages are considered periodic with jitters.

Tasks. For a task τ_i whose priority is P_i, the same arguments lead to formulae similar to those obtained for messages. However, we must distinguish two cases for the *Task Response Time* evaluation. For the non pre-emptive fixed priority scheduling, the equations (5) and (6) are directly applicable while, if the basic tasks are scheduled thanks to a pre-emptive fixed priority policy, the factor $\max_{i+1 \le j \le N}(C_j)$ has not to be considered (the possibility of pre-emption ensures that a task at a given priority level cannot be pre-empted by a task at a lower level). Therefore, the following recurrence relation allows to calculate the response time of a basic pre-emptive task

$$R_i^{n+1} = C_i + \sum_{j<i} \left\lceil \frac{J_j + R_i^n}{T_j} \right\rceil C_j \quad (7)$$

Again, equation (7) converges to a value as long as the processor utilization is less than or equal to 100%. And a suitable initial value for computing could be $R_i^0 = 0$.

Logical Chains. Finally, we can apply equation (5) for non pre-emptive case (respectively equation (7) for pre-emptive case) to calculate the worst-case response time of the two logical chains:

$$R\_worst_{lcx} = \sum_{y \in lcx} R_y \quad (8)$$

Figure 11 presents the bounds (minimum and the maximum response time) obtained thanks to this mathematical timing analysis for the both logical chains according to the equations (4) and (8). Note that the maximum response

time in Figure 11 corresponds to the non pre-emptive case for the configuration 2 while the two other configuration are based on pre-emptive assumption.

3.2.4 Comments on results

Configurations	Logical Chain	Logical Chain Response Times					
		Simulation results				Analytic results	
		minimum	mean	maximum	***Standard deviation***	***minimum***	***maximum***
Configuration 2	*lc1*	9.09	11.03	16.67	***1.775***	***8.992***	***22.116***
	lc2	9.82	12.82	16.67	***1.458***	***8.576***	***41.172***
Configuration 3	*lc1*	9.09	9.45	12.62	***0.667***	***8.992***	***16.116***
	lc2	12.45	14.16	20	***2.101***	***8.576***	***35.172***
Configuration 4	*lc1*	9.09	9.45	12.62	***0.667***	***8.992***	***16.116***
	lc2	12.45	12.61	14.11	***0.490***	***8.576***	***27.172***

Figure 11 - Response time evaluation

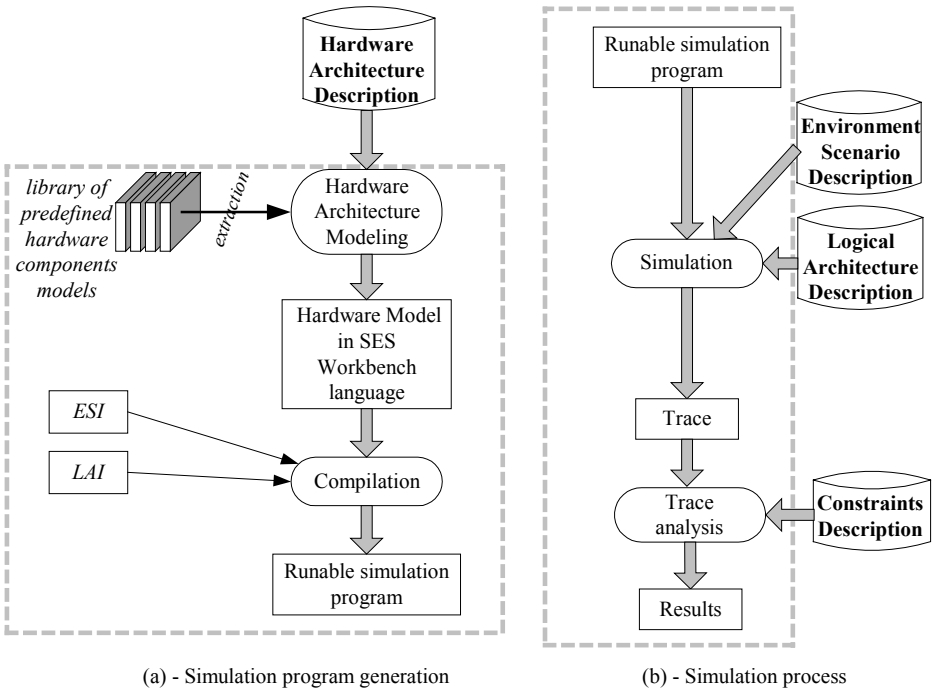
First of all, we notice that simulation results remain within bounds given by the analytic method of section 3.2.3. However, it can be seen that analytic bounds for the worst case are never reached during simulation. Maximum values obtained by simulation vary from 40% to 70% of analytic calculated worst cases while mean values vary from 30% to 60%. The importance of the simulation to obtain more realistic results becomes obvious when evaluating performances of an embedded system. From these tables we can also see that, compared to non pre-emptive scheduling, pre-emptive scheduling logically results in shorter response time for high priority tasks and longer response time for low priority tasks. Note however that this fact seems to be in contrast with analytic method results, where the worst-case bound gets better for pre-emptive policies than for non pre-emptive ones, irrespective of task priority! This is perfectly normal while results from the two methods have not to be interpreted in the same way: analytic results can be used as bounds to validate simulation results, but they have different meanings and they are rather complementary.

3.2.5 Automatic generation of models for simulation purpose

Usually, the direct use of a general-purpose simulation platform is not judged suitable by in-vehicle embedded system designers since too much effort must be made in building the simulation model. Thanks to a non-ambiguous description of embedded systems, as seen in section 2, it is possible to generate automatically a model that can be run

on a specific discrete simulation tool. For example, in [36], is proposed a modelling methodology, developed in collaboration with the French carmaker PSA Peugeot-Citroën, based on a component approach. This methodology has been implemented through the development of a simulation tool called Carosse-Perf and based on SES Workbench simulation platform [37]. It is composed, on the one hand, of a library of pre-built component modelled in SES Workbench formalism and, on the other hand, of a *constructor* that uses these models and the description of the embedded distributed architecture in order to obtain the whole model that will be simulated. The *constructor* extracts the pertinent information from the static description of the system at Logical Architecture level (tasks, data exchanged between tasks, behaviour), from the Technical and Hardware Architectures (policies for the access to resources – scheduler policy and network protocols-, performances of the hardware components) and, finally, from the description of how the Logical Architecture is mapped onto the Technical one. Technical and Hardware Architecture components are modelled once and for all in SES Workbench formalism. The principle of the model building is presented in Figure 12-(a). As at the simulation step, the behaviour of the logical architecture entities (tasks and messages) and the environment signal occurrences animate the simulation, the constructor has to include two generic modules in the model that will be executed by the simulator: an logical architecture interpreter and an environment scenario interpreter (named respectively LAI and ESI in Figure 12) whose role is to extract, during the simulation, the current event from logical architecture entities or environment signals that is to be managed by the discrete event simulator.

This kind of tool allows designers to easily build a simulation model of their new in-vehicle embedded systems (operational architecture) and then to simulate the model. More details about the underlying principles can be found in [36]. Carosse-Perf was used to construct automatically the models corresponding to the four configurations of the case study and to simulate them.



(a) - Simulation program generation (b) - Simulation process

Figure 12 – Simulator generation and simulation process.

4 Conclusions and future trends

The part of embedded electronics and especially embedded software takes more and more importance within a car, in terms of both the functionality and cost. Due to the cost, real-time and dependability constraints in automotive industry, many automotive specific networks (e.g. CAN, LIN, FlexRay) and operating systems (e.g. OSEK/VDX) have been or still being developed, most of them are within the SAE standardization process.

Today's in-vehicle embedded system is a complex distributed system mainly composed of four different domains: power train, chassis, body and telematic. Functions of the different domains are under quite different constraints. SAE has classified the automotive applications into classes A, B and C with increasing order of criticality on real-time and dependability constraints. For the design and validation of such a complex system, an integrated design methodology as well as the validation tools are therefore necessary.

After introduced the specificity of the automotive application requirements in terms of time-to-market, design cost, variant handling, real-time and dependability, and multi-partner involvement (carmakers and suppliers) during the development phases, in this chapter, we have described the approach proposed by EAST-ADL which is a promising design and development framework tailored to fit the specific needs of the embedded automotive applications.

Concerning the validation of the meeting of the application constraints by an implementation of the designed embedded system, we have reviewed the possible ways and illustrated the use of simulation for validating the real-time performance. This illustration is done through a case study drawn from a PSA Peugeot-Citroën application. The obtained results have shown that the use of simulation approach, combined with the timing analysis method (especially the holistic scheduling method), permits to efficiently validate the designed embedded architecture.

If we can consider that the power train and body domains begin to achieve their maturity, the chassis domain and especially the X-by-Wire systems are however still in their early developing phase. The finalization of the new protocol FlexRay as well as the development of the 42V power supply will certainly push forward the X-by-Wire system development. The main challenge for X-by-Wire systems is to prove that their dependability is at least as high as that of the traditional mechanical/hydraulic systems.

Portability of embedded software is another main preoccupation of the automotive embedded application developers, and consists of the another main challenge. For this purpose, carmakers and suppliers established AUTOSAR consortium (http://www.autosar.org/) to propose an open standard for automotive embedded electronic architecture. It will serve as a basic infrastructure for the management of functions within both future applications and standard software modules. The goals include the standardization of basic system functions and functional interfaces, the ability to integrate and transfer functions and to substantially improve software updates and upgrades over the vehicle lifetime.

5 Appendix: In-vehicle electronic system development projects

SETTA (System Engineering of Time Triggered Architectures). This project (January 2000 - December 2001) was partly funded by the European Commission under the Information Society Technologies program. The overall goal of the SETTA project is to push time-triggered architecture - an innovative European-funded technology for safety-critical, distributed, real-time applications such as fly-by-wire or drive-by-wire - to future vehicles, aircraft, and to train systems. The consortium is led by DaimlerChrysler AG. DaimlerChrysler and the partners Alcatel (A), EADS (D), Renault (F), and Siemens VDO Automotive (D) act as the application providers and technology validators. The technology providers are Decomsys (A) and TTTech (A). The academic research component is provided by the University of York (GB), and the Vienna University of Technology (A). http://www.setta.org/

EAST-EEA (Embedded Electronic Architecture) - ITEA-Project-No. 00009. The major goal of EAST-EEA (july 2001 – June 2004) is to enable a proper electronic integration through definition of an open architecture. This will allow to reach hardware and software interoperability and re-use for mostly distributed hardware. The partners are AUDI AG (D), BMW AG (D), DaimlerChrysler AG (D), Centro Ricerche Fiat (I), Opel Powertrain GmbH (D), PSA Peugeot Citroen (F), Renault (F), Volvo Technology AB (S), Finmek Magneti Marelli Sistemi Elettronici (I), Robert Bosch GmbH (D), Siemens VDO Automotive AG (D), Siemens VDO Automotive S.A.S. (F), Valeo (F), ZF Friedrichshafen AG (D), ETAS GmbH (D), Siemens SBS – C-LAB (D), VECTOR Informatik (D), CEA-LIST (F); IRCCyN (F),·INRIA (F), Linköping University of Technology (S), LORIA (F), Mälardalen University (S), Paderborn University – C-LAB (D), Royal Institute of Technology (S),Technical University of Darmstadt (D). www.east-eea.net/docs.

AEE project (Embedded Electronic Architecture). This project (November 1999 – December 2001) was granted by the French Ministry for Industry. It involved French carmakers (PSA, RENAULT), OEM suppliers (SAGEM, SIEMENS, VALEO), EADS LV company and research centers (INRIA, IRCCyN, LORIA) It aimed to specify new solutions for in-vehicle embedded system development. The Architecture Implementation Language (AIL_Transport) has been defined to specify and describe precisely any vehicle electronic architecture. http://aee.inria.fr/en/index.html

EASIS (Electronic Architecture and System Engineering for Integrated Safety Systems). The goal of the EASIS project (January 2004 – December 2006) is to define and develop a platform for software-based functionality in vehicle electronic systems providing common services upon which future applications can be built; a vehicle on-board electronic hardware infrastructure which supports the requirements of integrated safety systems in a cost effective manner; a set of methods and techniques for handling critical dependability-related parts of the development lifecycle and an engineering process enabling the application of integrated safety systems. This project is funded by the European Community (6th FWP). Partners are Kuratorium Offis E. V.(G), DAF Trucks N.V. (N), Centro Richerche FIAT, Societa Consortile per Azioni (I), Universitaet Duisburg – Essen, Standort - Essen (G), Dspace GMBH (G), Valéo Electronique et Systèmes de Liaison (F), Motorola GMBH (G), Peugeot-Citroën Automobiles SA (F), Mira Limited (UK), Philips GMBH Forschungslaboratorien (G), ZF Friedrichshafen AG (G), Adam Opel Aktiengesellschaft (G), ETAS (G), Volvo Technology AB (S), Lear Automotive (S), S.L. (S), Vector

Informatik GMBH (G), Continental Teves AG & CO. OHG (G), Decomsys GMBH (A), Regienov (F), Robert Bosch GMBH (G).

AUTOSAR (Automotive Open System Architecture). The objective of the partnership involved in AUTOSAR (May 2003 – August 2006) is the establishment of an open standard for automotive E/E architecture. It will serve as a basic infrastructure for the management of functions within both future applications and standard software modules. The goals include the standardization of basic system functions and functional interfaces, the ability to integrate and transfer functions and to substantially improve software updates and upgrades over the vehicle lifetime. The AUTOSAR scope includes all vehicle domains. A three tier structure, proven in similar initiatives, is implemented for the development partnership. Appropriate rights and duties are allocated to the various tiers: Premium Members, Associate Members, Development Member, and Attendees. http://www.autosar.org/

6 References

1. www.sae.org
2. Gabriel Leen, Donald Heffernan, *Expanding Automotive Electronic Systems, Computer*, Vol.35, N°1, pp. 88-93, IEEE Computer Society, January 2002.
3. Alberto Sangiovanni-Vincentelli, *Automotive Electronics: Trends and Challenges*, Convergence 2000, Detroit (MI), USA, October 2000.
4. Françoise Simonot-Lion, *In Car Embedded Electronic Architectures: how to ensure their safety*, Proceedings of the 4th IFAC Conference on Fieldbus Systems and their Applications - FET03, pp.1-8, Aveiro, Portugal, July 2003.
5. ISO, *Road vehicles – Interchange of digital information – Controller area network for high-speed communication*, ISO 11898, International Organization for Standardization (ISO), 1994.
6. ISO, *Road vehicles – Low-speed serial data communication – Part 2: Low-speed controller area network*, ISO 11519-2, International Organization for Standardization (ISO), 1994.
7. ISO, *Road vehicles – Low-speed serial data communication – Part 3: Vehicle area network*, ISO 11519-3, International Organization for Standardization (ISO), 1994.

8. Bruno Abou, Joël Malville, *Le bus VAN Vehicle Area Network: Fondements du protocole*, Dunod, Paris, 1997, ISBN 2 10 03160 0.
9. SAE, *Class B data communications network interface, J1850*, Society of Automotive Engineers (SAE), May 2001.
10. TTTech, *Specification of the TTP/C Protocol, version 0.5*, TTTech Computertechnik GmbH, July 1999.
11. OSEK, *OSEK/VDX operating system, version 2.2*, 2001. http://www.osek-vdx.org
12. John B. Goodenough, Lui Sha, The Priority Ceiling Protocol: A Method for Minimizing the Blocking of High Priority Tasks, Proceedings of th 2nd International Workshop on Real-Time Ada Issues, Ada Letters 8, 7 fall 1988, pp. 20-31.
13. Modistarc Project, http://www.osek-vdx.org/whats_modistarc.htm.
14 http://www.arcticus.se/.
15. Ulrich Freund, Michael von der Beeck, Peter Braun, Martin Rappl, Architecture Centric Modeling of Automotive Control Software,SAE Technical Paper Series 2003-01-0856.
16. DECOS Project, http://www.decos.at/.
17 A. Rajnak, K. Tindell and L. Casparsson, "Volcano Communications Concept", Volcano Communications Technologies AB, Gothenburg (Sweden), 1998. Available: http://www.vct.se
18. Paolo Giusto, Jean-Yves Brunel, Alberto Ferrari, Eliane Fourgeau, Luciano Lavagno, Alberto Sangiovanni-Vincentelli , "Automotive Virtual Integration Platforms: Why's, What's, and How's" , 2002 IEEE International Conference on Computer Design: VLSI in Computers and Processors (ICCD'02), pp370-378, Freiburg (Germany), September 16 - 18, 2002.
19. Medvidovic Nenad and Taylor Richard N., A Framework for Classifying and Comparing Architecture Description Languages, Technical Report, Department of Information and Computer Science, University of California, Irvine, 1997.
20. Vestal Steve, *MetaH User's Manual*, Honeywell Technology Center, 1995. http://www.htc.honeywell.com/metah/uguide.pdf
21. AEE, Architecture Electronique Embarquée, http://aee.inria.fr, 1999.

22. Elloy Jean-Pierre., Simonot-Lion Françoise, *An Architecture Description Language for In-Vehicle Embedded System development*, Proceedings of the 15th IFAC World Congress, IFAC B'02, Barcelona, Spain, 21-26 juillet 2002.

23. Migge Jörn, Elloy Jean-Pierre, *Embedded electronic architecture*, Proceedings of 3rd International Workshop on Open Systems in Automotive Networks, Bad Homburg, Germany, 2-3 February, 2000.

24. ITEA – EAST EEA Project, www.east-eea.net/docs.

25. U. Freund, O. Gurrieri, J. Küster, H. Lonn, J. Migge, M.O. Reiser, T. Wierczoch and M. Weber, "An architecture description language for developing automotive ECU-software", INCOSE'2004, pp101-112, Toulouse (France), 20-24 June, 2004.

26 www.mathworks.com/

27 Ascet SupplyChain, www.**ascet**.com/

28 Ilogic – Statemate, www.ilogix.com/

29 Esterel Technologies – SCADE Suite™ for Safety-Critical Software, www.esterel-technologies.com

30. C. Jard. *Automatic Test Generation Methods for Reactive Systems*. CIRM Summer School, Marseille, 1998.

31. Tindell Ken, Clark John, *Holistic schedulability analysis for distributed hard real-time systems*, Microprocessor and Microprogramming, vol. 40, pp. 117-134, 1994.

32. Y.Q. Song, F. Simonot-Lion and N. Navet, "De l'évaluation de performances du système de communication à la validation de l'architecture opérationnelle - cas du système embarqué dans l'automobile", Ecole d'été temps réel 1999, Poitiers (France), C.N.R.S., Poitiers (France), Ed. LISI-ENSMA, ISBN 2-9514541-1-2, 1999.

33. Song Ye-Qiong, Simonot-Lion Françoise, Bélissent Pierre, *VACANS - A tool for the validation of CAN-based applications*, Proceedings WFCS'97, Barcelona (Spain), Oct. 1997.

34. Coustre Alain, *The electrical electronic architecture of PSA Peugeot Citroen vehicles: current situation and future trends*, Presentation at Networking and Communication in the Automobile, Munich (Germany), March 2000.

35. Tindell K., Burns A., "Guaranteed message latencies on controller area network (CAN)", 1st international CAN conference, ICC'94, 1994.

36. Castelpietra Paolo, Song Ye-Qiong, Simonot-Lion Françoise, Attia Mondher, *Analysis and simulation methods for performance evaluation of a multiple networked embedded architecture*, IEEE Transactions on Industrial Electronics, 49-6, pp. 1251-1264, December 2002.

37. SES Workbench, HyPerformix Inc. http://www.hyperformix.com.