Building a Generic (cross-domains) Basic Software on top of the XtratuM hypervisor

Jean-Jacques METGE[(1)], Julien GALIZZI[(1)], Paul ARBERET[(1)], Bernard SANCHEZ[(2)], Jonathan PATY[(2)], Gilles SAINT-AUBIN[(3)], Mikael DESCHAMPS[(3)]

[(1)] CNES - Centre Spatial de Toulouse – 18, avenue Edouard Belin - 31401 TOULOUSE Cedex 9 - France
[(2)] Continental Engineering Services France - 1 avenue Paul Ourliac - BP83649 - 31036 Toulouse Cedex 1 - France
[(3)] Intertechnique / Zodiac Aerospace – 61, rue Pierre Curie – BP1 – 78373 Plaisir Cedex - France

Keywords : hypervisor, time & space partitioning, middleware, service oriented architecture (SOA), quality of service (QoS), basic software, monitoring & control

1- INTRODUCTION

Based on the lessons learnt of the aeronautical domain inherited from the [ARINC653] standard deployment, the CNES is currently working, since 2008, on the definition and the development of a new type of framework for embedded software development.
This framework, which is known in the space domain as “LVCUGEN”, consists in :

- a set of generic software building blocks providing a set of standard general purpose (middleware) services, compatible with the monitoring & control operations expected by the applicable standard in space (refer to [PUS])
- an innovative middleware technology :
    - capable to ensure the quality of service expected by the client applications
    - highly taking benefit of time & space partitioning property, inherited from the [ARINC653] standard, and ensured by the XtratuM hypervisor
    - highly de-coupled from monitoring & control services and protocols
    - highly de-coupled from hardware targets implementation

An internal evaluation of the framework, performed in 2010, successfully demonstrated a TRL4/5 maturity level on a single core LEON architecture, fully compatible with an operational deployment on a science payload for a future space observation satellite. This evaluation phase demonstrated also that this framework meets the targeted end-users expectations, in terms of increased independence between the internal functions of any embedded software, and also between software and hardware components, with very positive impacts on development cycles and integration efforts.
But this evaluation showed also that the satellites market will not probably be able to support alone the industrialisation costs of this framework, this situation being engraved by the emergence of multi-cores architectures.

In this difficult funding context, the CNES decided in 2011 to impulse a long terms initiative with several major french suppliers of space, aerospace and automotive domains, with the primary objective to extend the business model of this framework, enabling the emergence of competitive and perennial technology providers. Directly inspiring from the [AUTOSAR] business model, this initiative consists in giving the opportunity to these new end-users to influence the current definition of the software building blocks constituting the framework, while enabling an open competition between providers of this new technology.
The roadmap currently envisioned for the development of this cross-domains framework relies upon four steps :

- assessment of the LVCUGEN framework by suppliers from other domains than space, in the context of single core architectures
- pre-development of a generic (cross-domains) framework by “hybridization” of LVCUGEN framework with inputs from other domains than space, in the context of single core architectures
- extension of the generic framework capabilities to multi-cores architectures
- industrialisation of the generic (cross-domains) framework and pre-qualification vs a cross-domains qualification baseline, in the context of single core and multi-cores architectures

The present article aims at presenting the preliminary results of feasibility studies launched in 2011 by Intertechnique / Zodiac Aerospace (aeronautical use case) and Continental Engineering Services (automotive use case). These activities, which contribute to the first step, have been covered by an internal CNES budget dedicated to LVCUGEN framework valorisation.

2- THE LVCUGEN FRAMEWORK (CNES)

The LVCUGEN framework is the result of the combined exploration of several key technologies described hereafter.

a. Time & space partitioning (TSP)

The TSP property is ensured by a dedicated software building block ensuring the fundamental [ARINC653] principles : the XtratuM hypervisor.
The technical description of the XtratuM hypervisor, which can be found in [XtratuM], is out of the scope of the present article, that intends to focus more on the rationales that lead CNES to push for the emergence of such an hypervisor technology for embedded applications and also on the consequences of this technology on any software architecture, in which it is deployed.

Taking profit of the lessons learnt of aeronautical domain, the CNES decided, in 2008, to fund the development of a new hypervisor based technology for single core LEON architectures, because this technology is well known as a very efficient solution enabling to optimize the balance between kernel adaptability to specific end-users needs (ie choice of the guest OS, possibility to add specific services, easy adaptation to hardware) and simplicity of the kernel design.
In this particular context, the systematic research of the most simple design has been considered as an essential objective to be met by the XtratuM hypervisor, this property enabling in particular to ease the kernel determinism demonstrations, optimize memory and CPU footprints of the kernel and reduce kernel qualification costs, which were considered very early as critical objectives of the LVCUGEN project.
But the side effects of this technological choice are, in comparison to a more traditional microkernel :

- the very low level of services provided by the kernel
- the rejection of I/Os management outside the kernel

These two side effects implicitly raise the question, at architecture level, of implementing medium level services (ie middleware services), managing or not I/Os, inside partitions, with a significant probability that several partitions need exactly the same set of these medium level services.

b. I/Os virtualisation

The need for an I/Os virtualisation technology is the first direct consequence of the previous point, in the sense that, in an hypervisor based architecture, most of the hardware components outside the CPU core are considered as pure I/O devices, and must be consequently managed at partition level. This applies for example to non volatile memory controllers, I/O controllers (eg SPW, 1553, …) and specialised hardware devices (eg temperature & inrush current monitoring probes, …).
In this context, decision has been made in 2009 to define a standard solution for I/Os virtualisation inside partitions, which consisted in defining a standard SW/SW interface (named SSIS in the following) between I/Os drivers and medium level services that are supposed to use them.
The second consequence of the previous point is the fact that, in a TSP based architecture, several partitions must be able to share the same I/Os with a full guarantee of QoS. In the frame of the LVCUGEN project, this raised the additional need for a deterministic solution for QoS management of shared I/Os, since LVCUGEN project hypothesis was that the framework should be fully compatible with I/O devices that wouldn't ensure intrinsically this QoS by themselves.
In this context, the other decision made in 2009 has been to develop a deterministic I/O Scheduler, enabling to manage shared I/Os with a guaranteed QoS.
The main role of this I/O Scheduler is to manage, in a deterministic way, a configurable set of RX and TX sequences, either in WAIT mode (for synchronous exchanges) or NO_WAIT mode (for asynchronous exchanges), and to ensure the routing of frames between the I/O devices and the local RX/TX buffers of an "I/O Server" partition, interface with the I/O devices being, as explained above, ensured through I/O drivers compatible with SSIS. In this context, I/O data managed by the I/O Server partition can be accessed by any client partition through the inter-partition communication service provided by the XtratuM hypervisor.
The scheduling plan executed by the I/O Scheduler has to be statically determined by the platform integrator, generally with the help of the Xoncrete tool (see [Xoncrete]), according to the need of the hosted applications in terms of I/O types, maximum frames sizes, maximum dataflow rates and maximum latencies.
Determinism of the I/Os scheduling plan, and consequently QoS guarantee to all the client partitions, is consequently ensured by the fact that the scheduling plan of the I/O Server partition is perfectly known (property ensured by XtratuM) and the duration of each sequence is set equal to its WCET (property to be ensured by the platform integrator) and that a sequence is not considered as finished until its duration has not elapsed (property ensured by the I/O scheduler).

c. General Purpose Middleware (Basic Software)

The fact that the LVCUGEN architecture is based upon a computing platform hosting several partitions leads to the need to define a set of Common Services operating at computing platform level :

- shared I/Os management

- power management
- ON/OFF management
- Resets management
- Modes management
- HW & SW monitoring management
- Built-in-tests management
- Warning management
- Failures & errors management
- Non Volatile Memory management
- Software uploads management
- Software configuration control management
- Instrumentation service management

In order to ensure portability of the framework between the projects, decision has been made in 2009 to define a services oriented architecture (SOA) enabling to implement these services independently of the high level protocols and services that enable the monitoring & control of the computing platform.
The rationale is that, even if the space domain is permanently willing to standardise the monitoring & control protocols applicable to satellites, a majority of satellites projects still continues to define their own instantiation of the standard, which has generally a direct impact on the on board software implementation.
The second key design choice of these services has been to cluster them into five specialized “Generic Engines”, implemented in such a manner that they could be developed and qualified independently 2 by 2 inside separate partitions, with very limited and simple interfaces between them :

- I/O Engine, for shared I/Os management
- MMDL Engine, for :
    - modes, ON/OFF, resets & modes management
    - dataloading operations (ie software uploads management, software configuration control management, memory dumps & memory checks management)
- HSEM Engine, for HW & SW events, built-in-tests (including EDAC management), warning, failures & errors management
- INSTR Engine, for instrumentation service management
- MMM Engine, for the management of operations in the Non Volatile Memory

Thanks to this implementation, the targeted goal was to provide capability to implement and qualify incrementally the whole set of Common Services, to have the possibility to fully characterize them, in terms of QoS, and to parallelize their developments, with the possibility to enable/disable a subset of them or to add a new subset of services, depending upon the specific needs of the end-user project, with a minimum impact of the remaining subset of services. In this context, remote invokation of the Common Services by external partitions imposed to define a standard communication protocol, implemented on top of the inter-partitions communication service provided by XtratuM, the portability of these services on the hardware platforms being ensured through basic drivers compatible with the SSIS interface.
In terms of implementation, each Generic Engine is developed as a single thread code, based upon a dedicated “actions scheduler” adapted to its exact functional needs and QoS constraints :

- deterministic scheduler for the I/O Engine (see above). The rationale is that a strict QoS must be ensured by the I/O Engine to any client partition, whatever the status of the other partitions sharing the I/O device
- basic events handler mechanism provided by XtratuM, for the MMDL Engine. The rationale is that modes changes and memory operations occur relatively rarely and that the expected QoS associated to these operations is generally not too stringent.
- flexible scheduler for the HSEM Engine, guaranteeing response times for a set of pre-defined actions, while providing a dynamic CPU reallocation capability to a background action (eg scrubbing) depending upon the effective execution times of the pre-defined actions. The rationale is that the duration of the actions linked to the management of events is highly dependant of the status of the observation points that are currently monitored

INSTR Engine and MMM Engine are not implemented yet, but, in a first approach, it is probable that they could be respectively implemented with the flexible scheduler of the HSEM Engine and the deterministic scheduler of the I/O Engine.
The second important characteristics of each Generic Engine consist in their high configurability, ensured by dedicated configuration tables, populated by the platform integrator, and implemented as pure binary code, generated independently of the binary code of the Generic Engine it is associated to.
A synthetic overview of the General Purpose Middleware architecture (Basic Software) is provided in the following figure :

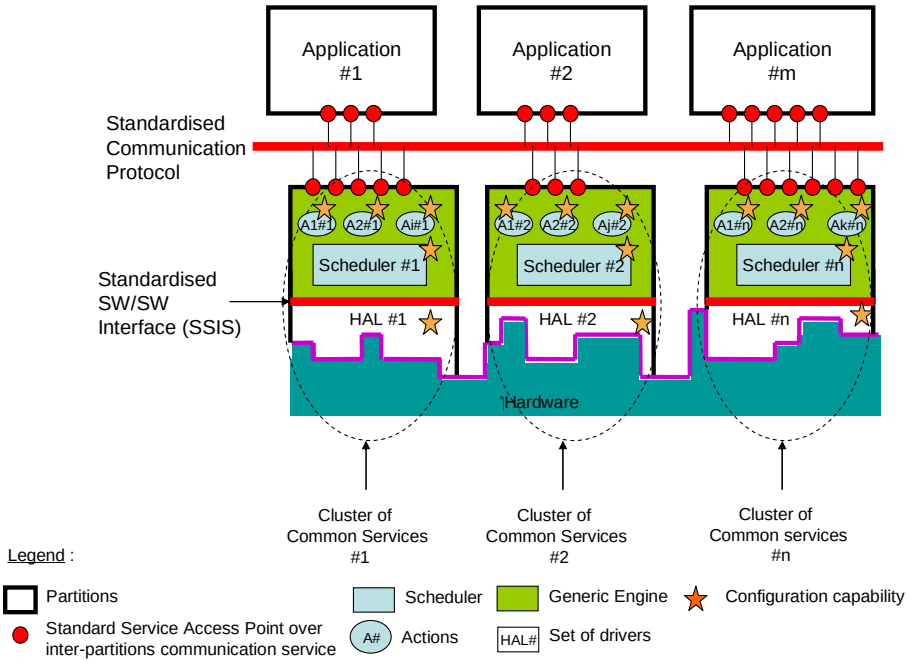

Figure 1 : static architecture of Basic Software

d. Qualification package

Based on the lessons learnt of the aeronautical domain derived from the [DO297] considerations, the last main objective targeted by the LVCUGEN project was the possibility to apply an incremental qualification process to the computing platform and to the hosted applications.
This firstly lead to the definition and development of a qualification kit applicable to each generic software component of the computing platform (ie XtratuM hypervisor and Generic Engines), enabling to demonstrate the functional validity of a component and to measure the WCETs of the Common Services they provide, on the final hardware target. This qualification kit, which is part of the LVCUGEN framework, must be executed each time a generic software component Engine is ported on a new hardware platform.
The need for incremental qualification process capability lead also to define a qualified toolchain (*) for :

- automatic validity checking of XtratuM configuration parameters
- automatic generation of configuration tables of XtratuM and Generic Engines
- system binary images production

(*) In the LVCUGEN process, the management of the CPU allocations between the partitions, the management of the precedences, dataflows and temporal constraints management across partitions and the offline system schedulability analyses, are directly managed by the platform integrator through the support of the Xoncrete tool, which will not be qualified. The description of the Xoncrete tool is out of the scope of the present article and can be found in [Xoncrete].

3- AUTOMOTIVE USE CASE (CONTINENTAL ENGINEERING SERVICES)

a. Feasibility study for automotive application

A feasibility study has been performed by Continental Engineering Services to evaluate the LVCUGEN framework in an automotive context. The first target of this study was an engine control application (ECU), but it quickly appeared that the static scheduling plan provided by Xtratum could not fit to the dynamic behavior of an ECU software. A part of it is triggered asynchronously by engine events (crank events, that appears with a so-called period that ranges from 4 ms to 60 ms), and the expected activation time is very short (a few µs). This requirement can't be reached with a static temporal design.
So the study turned toward a body controller module (BCM), which does not impose such dynamic constraint, as its functions are only triggered periodically.

b. Requirements of BCM system

Continental provides to a French OEM a hardware and software platform for a BCM application. This system manages body functionalities (wipers, electric windows, air conditioning…), centralized diagnosis and is a gateway between all other electronic equipment in the car. The hardware platform is based on a 32 bits microcontroller. It includes a Mbytes flash memory, [CAN] cells networks, LIN networks and hundreds of electric (discrete, analogical or PWM) inputs and outputs.

The software of this application, based on [AUTOSAR] release standard, is split in two parts :

- BSW developed and configured by Continental, and provided to the customer (source code is delivered),
- ASW developed and integrated by the customer into the BSW platform.

For this use case, the customer requires that the whole functionality of BSW is executed without consuming more than 40% of CPU load, and no task loss is allowed as inputs and outputs are not buffered. Moreover, I/O and LIN treatments have to be achieved with as minimum jitter as possible. The implemented functionalities are distributed into stacks, specified by the [AUTOSAR] standard:

- I/O Hardware Abstraction: writing and reading on discrete I/Os, ADC conversions and PWM driving,
- Communication Services: emission and reception on [CAN] and [LIN] networks, use of networks as wake-up sources, on-board diagnosis through [CAN] network,
- Memory Services: reading and writing of configured data blocks in EEPROM during system state transitions (wake-up, shutdown) or on-the-fly,
- System Services: error management and functions inhibition, ECU state management, watchdog management…

c. Use case feasibility study

The LVCUGEN framework faces with two main differences toward the current realization in the [AUTOSAR] standard:

- it enforces, by the usage of Xtratum, a static scheduling plan, whereas the actual OS has dynamic features such as preemption and management of tasks priorities,
- it provides generic engines to manage shared hardware resources, while [AUTOSAR] defines stacks aimed for the same goals, but with specific automotive functionalities.

The study will intend to show how far can the LVCUGEN framework can be reused in the respect of the customer requirements.

A preliminary study has shown that the LVCUGEN general purpose middleware could be partially reused to cover the [AUTOSAR] BSW requirements. Depending on the engine, reuse could be easy, or would need some adaptations:

- the I/O engine fits to the basic requirements of the I/O [AUTOSAR] stack, but does not offer extended functionalities, mostly required by the communication stack for [CAN] and [LIN] networks management. The access to I/O resources can be managed by I/O engine, and specific partitions must be developed to provide the specific [AUTOSAR] features,
- by an extension of its configuration capability, the MMDL engine could be able to manage ECU states as it is expected in automotive. The reprogramming functions are currently embedded in the boot software, but the use of the data loading capability of the MMDL engine can be useful to allow fast reprogramming (in order to download one specific partition for instance, which is not possible with the [AUTOSAR] architecture),
- the error management functionalities brought by the HSEM engine would not replace the DEM and FIM modules that fit to applicative automotive needs. But the engine will be used to manage events linked to the partitioned architecture,
- the INSTR engine provides measurement capability to the upper partitions. A deep study has to be done to determine the adaptations needed on the XCP module, that provide this feature for automotive needs, to fit to INSTR engine interfaces,
- the MMM engine for non-volatile memory management is not yet developed.

So, in order to migrate the BCM software into the LVCUGEN framework, the upper level of the [AUTOSAR] stacks will be reused in specific partitions, because they embed enhanced automotive functionalities, and some LVCUGEN engines will be used to cover the lower level requirements:

- the [AUTOSAR] OS will be removed, task management being fully handle by the Xtratum scheduler,
- the microcontrollers drivers have to be developed to fit both to hardware interfaces and to Xtratum interfaces,
- each [AUTOSAR] stack is integrated as a Xtratum partition. The lower level modules can be replaced by LVCUGEN engines,
- a specific partition has to be reserved for the customer’s application.

The resulting physical design of the application is described in the following figure :

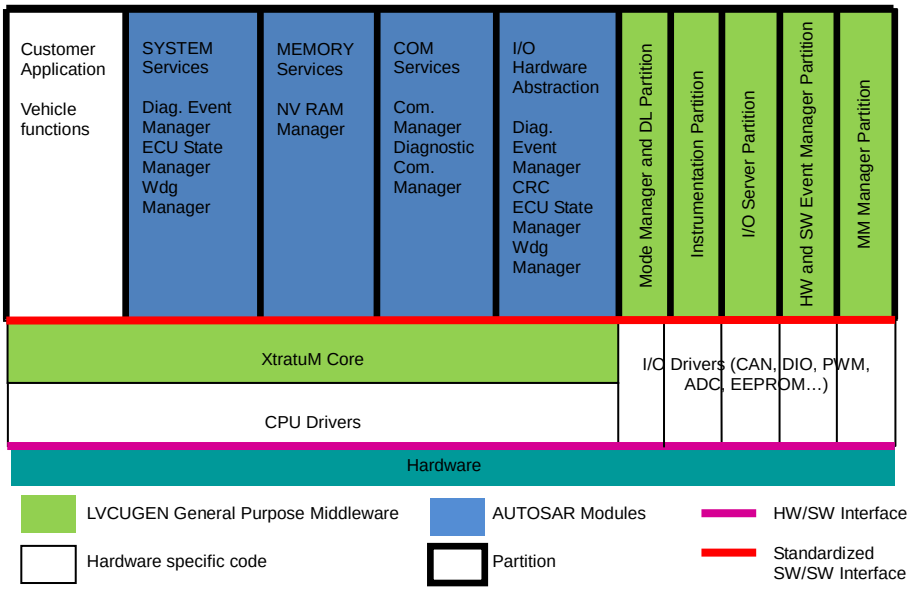

Figure 2 : static architecture of the BCM software

The use of LVCUGEN framework offers several benefits compared to the actual technical solution and process. First, the QoS of each partition is guaranteed by the static scheduling plan and the design of specific schedulers embedded in each partition. With the current way of working, it is only demonstrated experimentally, by running software for a long time to try to reach every critical software setpoint. It also improves the whole V-cycle: the specification and design duration is increased (around +10% for specification phase and +20% for the design phase), but the integration tests are less important (around -30%), and the integration of built and tested object files brings profits in case of reuse of software. In a long term view, and also in a reuse context, some time will be saved on the development of an application.
The quantity of generic engines that can be reused has to be specified with a more detailed study, but this first step has shown that there will be a mix between LVCUGEN BSW (use of generic engines for functions close to hardware drivers) and AUTOSAR modules (integrated into specific partitions to provide automotive requirements) :

- the I/O engine could be fully reused, and the dedicated partition would only contain enhanced functionalities,
- the MMM engine development could be based on the Memory Services stack,
- [CAN] and [LIN] drivers could also be integrated into the I/O engine.

In this first study, all the tasks execution time were specified with the measured WCET. As a result, the CPU load consumption would be close to 70%, whereas the customer allows only a 40% CPU load consumption. This way of designing the scheduling plan is not a good approach.
It would be necessary to create a scheduling plan that respects the resources consumption requirements. Then, it will be necessary to specify a accurate QoS need, to identify critical partitions that cannot afford to lose tasks, and less critical partitions that can support some jitter. As a result, the real QoS would be known, and the CPU load which is available for the customer application would be also guaranteed.

d. Conclusions and evaluation continuation

The study of this use case has shown that technically, it is feasible to migrate a BCM application into the LVCUGEN framework. The part of the middleware that can be reused has to be studied accurately, and some benefits can be expected on the development process. The biggest effort will have to be done to specify a new dynamic architecture, to propose a more deterministic approach of the scheduling, and then to guaranty a better QoS that the actual one. The static design will also have to be studied deeper: at this time, each AUTOSAR stack is mapped in a single partition. Communication between stacks has to be identified: a better mapping of the AUTOSAR module into partitions could avoid some hypercalls and save execution time.

4- AERONAUTICAL USE CASE (INTERTECHNIQUE / ZODIAC AEROSPACE)

a. Evaluation context

The purpose of this evaluation was to determine the ability of the LVCUGEN framework to provide a consistent alternative to existing COST or proprietary solutions in aerospace domain, from several viewpoints, among which technical, safety/certification and costs ones are the most important.

This evaluation was based on the furnished documentation: user and reference manuals, interfaces descriptions, implementation examples of use cases extracted from the spatial domain and a trial version of the associated scheduling configuration and analysis tool (Xoncrete).
This evaluation was performed considering the porting of a real use case on a subset of an Integrated Modular Avionics architecture (fully distributed architecture).
The resulting hardware architecture of the use case is depicted in the following figure :

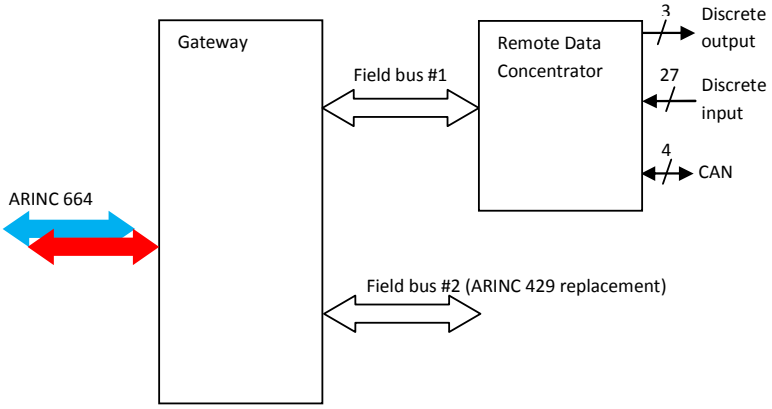

Figure 3 : example of IMA architecture sub-element

b. Requirements & constraints

i. Certification and safety considerations

Safety aspects and their corollary, certification issues, are key points in the aerospace world in general and in the Integrated Modular Avionics in particular.
These considerations are sustained by a few concepts:

- Robust partitioning in space, time and considering inputs/outputs (a faulty application shall not disturb other applications),
- Protection against high-energy (mainly neutrons) particles,
- An incremental development and certification process involving different actors and requiring modularity and configuration parameters' segregation and the definition of a usage domain.

The partitioning topic is addressed by the [ARINC653] standard, which provides as well an application programming interface. The certification subject is mainly covered, at software level, by the [DO178B] standard.
Although security of information is not yet a real concern, it may become an emerging need in a near future.

ii. Functionalities and performances

Most of aerospace applications do not require high levels of timing performance (with an exception for braking and landing gear systems): expressed latencies and responsiveness are not lower than a few tens of milliseconds. At the opposite, they may require high computing performances.
Concerning the presented use case:

- Hundred of messages are exchanges on the [CAN] busses, between 6 to 40 "label like" messages are exchanges on field bus #2. Most of the inputs and outputs require a refresh rate between 500 msec and 1 sec, this can fall down to 50 msec concerning some few information on field bus #2.
- Concerning [ARINC 664] messages, 96 messages distributed on 45 virtual links are received (with a worst case data rate of 320 bytes every 30 msec), 25 messages distributed on 16 virtual links are transmitted (with a worst case data rate of 244 bytes every 50 msec).
- Expected responsiveness (worst case) for a given function following a particular event is around 2 sec.

Concerning the functionalities, a few ones are commonly used and as a consequence are expected:

- fault management to detect and identify problems during system initialization or at runtime, to report these problems with possible correlation and take action at different levels(platform, partition or service, task),
- an [ARINC 615A]/[ARINC 665] compliant data loading service including correctness verification and compatibility check features,

- Instrumentation services for debug and integration purpose.

Finally, Freescale processors, being commonly used in the aerospace domain, are preferred.

iii. Costs

Recurring and non-recurring costs represent, as in many domains, a key point. It is not possible to give accurate figures; it is easier to identify common expensive items.
In relationship with the above considerations, these items are:

- A certification kit to support the corresponding activities ; such a kit can include partitioning demonstration, worst case execution time analysis, test procedures and scripts, etc…
- Supplemental developments (porting, customization, additional drivers developments, etc…).

Long-term considerations to assess a cost are re-usability, adaptability, scalability of the considered components, toolset to support applications development and certification activities, etc…
Ease of installation and host platform requirements, documentation quality and technical support, should not be let apart.

c. Synthesis

i. Possible implementation using the LVCUGEN framework

Static architecture depicted below represent a possible technical implementation onto two different kinds of avionics equipments. It shows that the LVCUGEN framework, including system partition enhancements (to support avionics standard), and provided a buffered inputs and outputs management, is well suited to avionics application.
The gateway (gateway between the [ARINC 664] Aircraft Data Communication Network and the Remote Data Controller) shall host user partitions (identified “PAx”) in charge of implementing the functional requirements, and system partitions (identified “SYSx”) that have to perform generic/common tasks: [ARINC 615A] compliant dataloading, Built-In Test and monitoring, instrumentation.
For this purpose, the available generic engines have to be completed to comply with avionics standards or directive: the HSEM may be a basis for a control & monitoring partition, the MMDL for an [ARINC615A] partition. A network management partition should be developed from scratch.
Several tasks are commonly managed within a user partition, thus requiring the use of a guest OS. Moreover, an [ARINC 653] API is required.
Regarding the timing requirements, the definition of four 15 msec MInor Frames (MIF) within a MAjor Frame (MAF) is enough. The wake-up of the I/O server partition before and after every partition scheduling should ensure the required information freshness and latencies requirements, and is supported by the processor considering its computing capabilities.

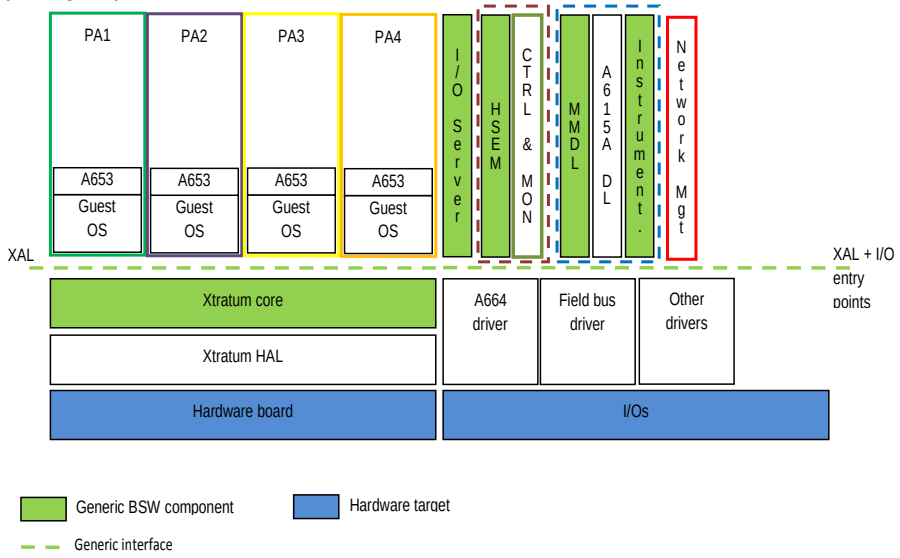

Figure 4 :static architecture of the Gateway software

The Remote Data Concentrator (or RDC, which purpose is to concentrate I/O data towards the gateway) will not host user partitions, but configuration tables. Features equivalent to the gateway’s system partition are required, except the network management one, thus requiring the here above described

enhancements concerning MMDL and HSEM. There is no need for an [ARINC653] layer, and mono-threaded partitions could be convenient. Access to the field bus or to the I/Os is exclusive to the concerned partition, an I/O server is therefore not mandatory, but it may constitute a basis for the development of the I/O and field bus management partitions (see figure below).
The main functionality of a RDC being the computation and routing of data flows, a scheduling plan with a MAF constituted of two 70 msec MIFs, fulfils the timing requirements.

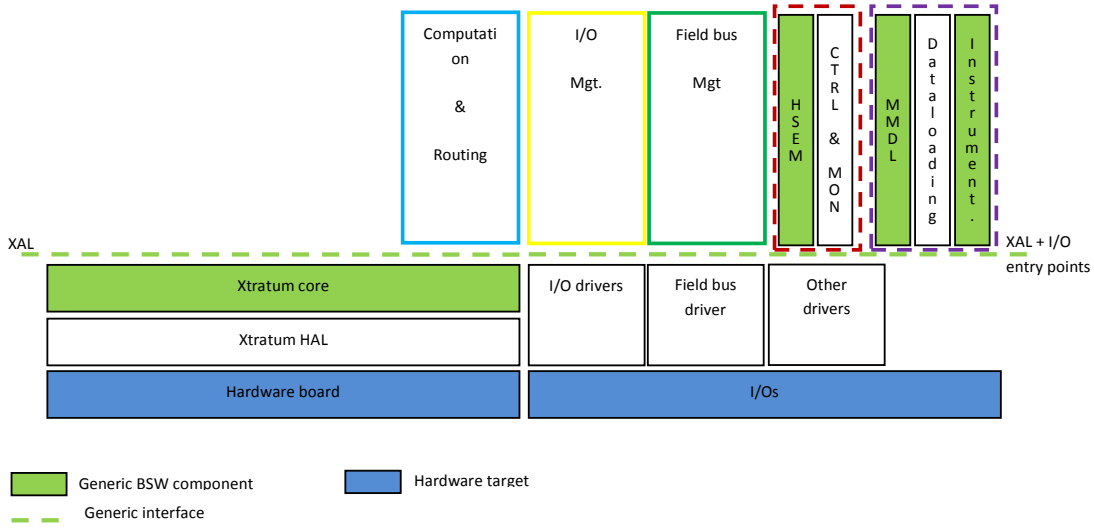

Figure 5 :static architecture of the Remote Data Concentrator software

ii. Strong points

The evaluation performed by Intertechnique / Zodiac Aerospace shows that the LVCUGEN frameworks provides :

- Strong spatial, time and input/output scheduling.
- Time partitioning demonstrated at configuration level via Xoncrete.

And that:

- Development effort reduction using the generic components to answer to avionics standards requirements.

iii. Weak points and improvement opportunities

At the opposite, the following improvements should be considered in order to ease the LVCUGEN framework diffusion in the aerospace world :

- Economical and/or licensing model to be clarified,
- A tasks/threads management capability at partition level,
- Availability of an [ARINC653] layer.

Finally, concerns remains on unbuffered input/output low latencies requirements leading to an over-scheduling with loss of efficiency and even unschedulability.

5- CONCLUSION

The preliminary assessments performed by Continental Engineering Services and Intertechnique / Zodiac Aerospace concluded that the LVCUGEN framework, in its current definition :
- is well suited to Gateway and Remote Data Concentrator applications of Intertechnique / Zodiac Aerospace (aeronautical domain) and to Gateway application of Continental Engineering Services (automotive domain)
- is not adapted to applications requiring extreme reactivity to events (typical max latency of several µs).
More globally speaking, these results provide an encouraging trend that confirms the cross-domains potentiality of the LVCUGEN framework for aeronautical and automotive domains, in the class of applications from low to high reactivity, which generally constitute a very big proportion of embedded systems (probably around 70%). This trend is fully in line with the conclusions of the experimentation already performed by the CNES in the space domain on the perimeter of satellites payloads.

On the basis of these very encouraging results, it is now necessary to consolidate the cross-domains

potentiality of the LVCUGEN framework by a real implementation on representative hardware targets, in contexts representative of operational R&T projects.
The new AVIONIQUE-X French project for the space domain (space vehicles) and the future ASHLEY European project for the aeronautical domain (SCARLETT continuation) could constitute very good opportunities to perform these experimentations, through the development of computer prototypes, based upon the LVCUGEN framework.
No particular automotive R&T project could be identified up to now, probably due to the fact that the whole automotive domain is currently investing a lot in the [AUTOSAR] implementation and that the new paradigm induced by the LVCUGEN framework would constitute an additional step, that probably comes too early. However, the current trend of the automotive domain, consisting in higher integration and bigger computed assistance into critical functions will probably lead, on longer terms, to the emergence of future R&T projects, where the LVCUGEN framework could take its place.
Another option could be to try to set up “from scratch” a new cross-domains R&T project, aiming at defining a common standard to these three domains, and defining a new generic (cross-domains) framework, inherited from the LVCUGEN one, but the efforts done in 2011 to raise such a project showed such an objective necessitates a temporal convergence between the technological roadmaps of all the involved domains, which does not appear to be realistic. So this solution may not be the preferred option.

Whatever the context that will be found for these future experimentations, the distribution, licensing and governance schemes of the LVCUGEN framework already appeared to be crucial questions, in the sense that these three topics are completely linked with the business model that can be effectively reached by this framework. The additional question behind these considerations is the capability to create the economical conditions that would enable the emergence of a cross-domains supplier (techno-provider of the generic framework), that will be ready to invest on this technology. These crucial questions apply also to the XtratuM framework (ie hypervisor and associated tools), whose future seems to be tightly linked with the future of the LVCUGEN framework.

In this complex context, the current priority for the CNES, which is a national agency dedicated to space, is to determine the way the LVCUGEN framework could be made open to experimentations by end-users, in particular those of other domains than space, while being able to provide the necessary technical support for these experimentations.
Probably the new IRT AESE (Institute of Research and Technology for Aeronautical, Space and Embedded Systems), currently created in Toulouse in the frame of a French public loan (“Grand Emprunt”), could temporally constitute the best relay for this activity, until a cross-domains techno-provider of the LVCUGEN framework can be formally identified.

6- REFERENCES

a. Standards

[ARINC615A] Software Data Loader using Ethernet Interface – Ref ARINC 615-A
[ARINC653] Avionics Application Software Standard Interface - Ref ARINC 653
[ARINC664] Aircraft Data Network – Ref ARINC 664
[ARINC665] Loadable Software Standards – Ref ARINC 665
[AUTOSAR] AUTomotive Open System Architecture (AUTOSAR) - http://www.autosar.org/
[CAN] Controller Area Network – Ref ISO 11898
[DO178B] Software Considerations in Airborne Systems and Equipment Certification – Ref RTCA DO178B/EUROCAE ED-12B
[DO297] Integrated Modular Avionics (IMA) Design Guidance and Certification Considerations – Ref RTCA DO-297/EUROCAE ED-124
[LIN] Local Interconnect Network - http://www.lin-subbus.de/
[PUS] Ground systems and operations – Telemetry and telecommand packet utilization – Ref ECSS-E-70-41A

b. Publications

[Xoncrete]
V. Brocal, M. Masmano, I. Ripoll, A. Crespo, P. Balbastre and J-J Metge.
Xoncrete : a scheduling tool for partitioned real time systems
In *ERTS² 2010. Embedded Real Time Software and Systems. May. Toulouse 2010*

[XtratuM]
M. Masmano, I. Ripoll, A. Crespo, J-J Metge and P. Arberet.
XtratuM : an open source hypervisor for TSP embedded systems in aerospace.
In *DASIA 2009. Data Systems In Aerospace. May. Istanbul 2009*