Avionic-X: A demonstrator for the Next Generation Launcher Avionics

David Monchaux, Philippe Gast, Jérémie Sangare

Avionic-X: A demonstrator for the Next Generation Launcher Avionics

David MONCHAUX
Centre National d'Etudes Spatiales, Direction des Lanceurs
Rond-Point de l'Espace
91023 Evry Cedex - France
e-mail: *david.monchaux@cnes.fr*

Philippe GAST, Jérémie SANGARE
ASTRIUM Space Transportation
Route de Verneuil
78133 LES MUREAUX Cedex – France
e-mail: *philippe.gast@astrium.eads.net*
jeremie.sangare@astrium.eads.net

***Abstract*— The purpose of this paper is to describe Avionic-X, which is a high level integration bench for the next generation launcher avionics. First we will introduce the project and its context, second we will present the work logic, starting by studies on avionics architecture and technological enablers, and finally we will give an overview of the Avionic-X platform itself.**

Keywords: Avionic-X, Distributed architectures, Multi-core platforms, Network embedded systems, Middleware, Virtualization, Innovative architectures, Model-based system engineering, IMA.

I. INTRODUCTION

Avionic-X is a project of demonstrator for the Next Generation Launcher avionics, funded by the French PIA [1] (with CNES as operator, and ASTRIUM-ST as architect) and European partners. It aims at maturing technologies and integrating them in order to demonstrate all avionics functions of a launcher, thus reaching TRL6 and IRL3 in 2015-2016. Afterwards, Avionic-X will enter into its second life, and will become a functional integration bench (Iron Bird) used for testing new equipments, benchmarking different concepts, while benefiting from the platform features:

- Test cases for benchmarking. It will be possible to play and replay a specific part of a launcher mission, or to run entire missions;
- Modularity and connectivity (the demonstrator will be compatible with several communication buses, protocols and field buses);
- Electrical Ground Support Equipment (EGSE), power distribution and control, exploitation means;
- Real-time flight simulator for closed loop tests;
- A "virtual platform" (Virtual Iron Bird), which is an important by-product of the model-based development, and could be used to perform virtual tests before actually plugging the new hardware onto the platform.

[1] Plan d'Investissement pour l'Avenir

This concept for Avionic-X is derived from the "Iron Bird" or the "Aircraft Zero" in aeronautics. During the first years, Avionic-X would be more analogous to the *Aircraft System Validation Rig* combined with the *Virtual Iron Bird*, when using terms from the "Power Optimized Aircraft" (POA) European framework program.

Avionic-X will build upon previous Research & Technology (R&T) activities, and on spin-in from other innovative sectors such as aeronautics, automotive, satellites, to prepare for the future development of a new European launcher (maiden flight in 2025). Indeed, the practical target for this integration bench is not Ariane 5 avionics evolutions but the Next Generation Launcher (NGL), also known as Ariane 6 in France.

II. CONTEXT

We won't change our methods unless forced to. But today, the established western rocketry industry is being challenged (by new business models, by emerging space nations…). Access to space shall be rendered affordable for European states, in a "cut cost or die" logic. Therefore, in order to prepare a future development program we have to go back to the drawing board, and question our previous historical choices, whilst keeping both eyes open for new technologies and new architectures.

The main drivers for the Next Generation Launcher avionics are:

- Reduction of the Total Cost of Ownership (with an emphasis on recurring costs) of the Launch System, including obsolescence treatment costs;
- Avionics mass reduction and miniaturization;
- Avionics performance improvement, keeping at least the same reliability.

Today, several initiatives coexist in a complementary way in Europe to prepare the Next Generation Launcher Avionics, from ESA, national agencies and industry. The figure 1 below outlines how Avionic-X project is integrated with the Future Launchers Preparatory Program (FLPP).

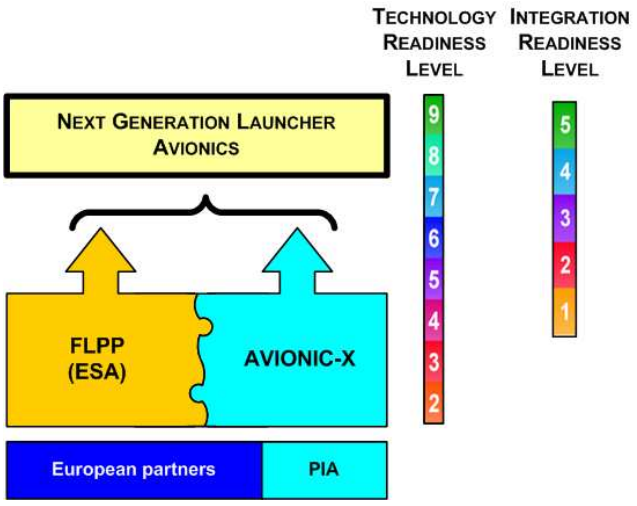

Figure 1. Context of the Avionic-X project

Actually there are several other ESA technology programs besides FLPP which include avionics aspects, for instance:

- Basic Technology Research Programs (TRP) for TRL from 1 to 3. In particular ESA's Deep Sub-Micron initiative is working with industry to design a new generation of space-worthy microchips;
- General Support Technology Programs (GSTP) for TRL from 2 to 8.

Besides, the Launch Vehicle (LV) avionics sector is too small a market to allow for self-standardization and to fund specific technologies development. The "Not Invented Here" syndrome is not an option. Therefore we have to seek ideas and technologies in other innovative sectors, and establish a spin-in, rather than recreate everything from scratch.

Several trends from outside the Launch Vehicle (LV) sector seem promising:

- The *"More Electric Aircraft"* trend in aeronautics. On a launcher, this could translate into electromechanical actuators (EMA) for Thrust Vector Control, and electrical valves in cryotechnic engines, thus giving birth of a *"More Electric Launcher"*.
- Standardization is a trend in the satellite industry with the Space Plug-and-Play Architecture (SPA), the SAVOIR [2] initiative, and more generally for embedded systems with the DDASCA [3] consortium. This could also encompass the IMA concepts (Integrated Modular Avionics), and the ARINC 653 Time and Space partitioning (TSP) standard [3]. Several projects are currently trying to spin-in from the aeronautical domain the TSP/IMA technology for space applications.

Three of the ongoing initiatives leading towards more standardization in avionics, with reference architectures and building blocks, are presented hereafter:

- SPA (Space Plug-and-Play Avionics),
- SAVOIR,
- DDASCA.

SPA was developed in United States by the Air Force Research Laboratory (AFRL) [4]. The draft SPA standard has already been released through the American Institute of Aeronautics and Astronautics (AIAA). The SPA effort is a response to the need for reduced design, fabrication, integration, and test schedules (and therefore related engineering costs) for small spacecraft, thanks to self-configuration and self-organization.

SAVOIR stands for Space Avionics Open Interface Architecture. The space industry and Agencies have recognized this already for quite some time: The level of standardization in the spacecraft avionics systems should be raised in order to increase efficiency and reduce development cost and schedule.

It has been proposed to federate several ongoing initiatives under the common "Space Avionics Open Interface Architecture" initiative. Within this initiative, the approach based on reference architectures and building blocks plays a key role. The SAVOIR Advisory Group (SAG) members regroups space agencies, large and small system integrators and equipment/software suppliers.

Lastly, DDASCA is a newly born consortium dedicated to Distributed Dependable Architectures for Safety-Critical Applications. It regroups universities and research centers, system integrators and equipment/software suppliers, with standardization organisms.

It aims at engineering and standardizing reference architectures with generic building blocks, that will be modular, provable, reusable and safe, to build hardware and software platforms for safety-critical applications (DAL A – SIL 4).

Within the Avionic-X project frame, we will strive to take in a wide range of promising concepts, emerging standards and innovations (with a TRL above 2), and perform trade-off analysis w.r.t. their interest in a launcher system context, and finally prototype and test the most interesting candidates on the platform.

III. AVIONIC-X WORK LOGIC

A. *Work Logic presentation*

Avionic-X project is more than a platform or a list of technologies. Its work logic is built on three axes:

- #1 - Demonstrator activities (target IRL-3),
- #2 - Technological activities (target TRL-6),
- #3 - Launcher avionics engineering.

As shown on the figure hereafter, these three branches of Avionic-X interact all along the project in order to reach our targets in terms of TRL and IRL.

[2] Space AVionics Open Interface Architecture

[3] Distributed Dependable Architectures for Safety-Critical Applications

[4] With the help of the Swedish company ÅAC Microtec.

The needs for demonstrations arise equally from theoretical studies on various launcher avionics architectures ("SEL-X" activities) in a top-down approach, and from technological enablers studies in a bottom-up approach. The "technological branch" also focuses on innovative methods with the same goal in mind: reducing the Total Cost of Ownership (TCO) of the launcher system.

The project Avionic-X is currently in phase A (system concepts feasibility). It is due to pursue its phase B (specification and preliminary design) from the beginning of 2012, with an incremental development cycle, similar to the *"spiral model"* common in software engineering. This will offer several windows of opportunity to introduce new technologies and new partners in the project.

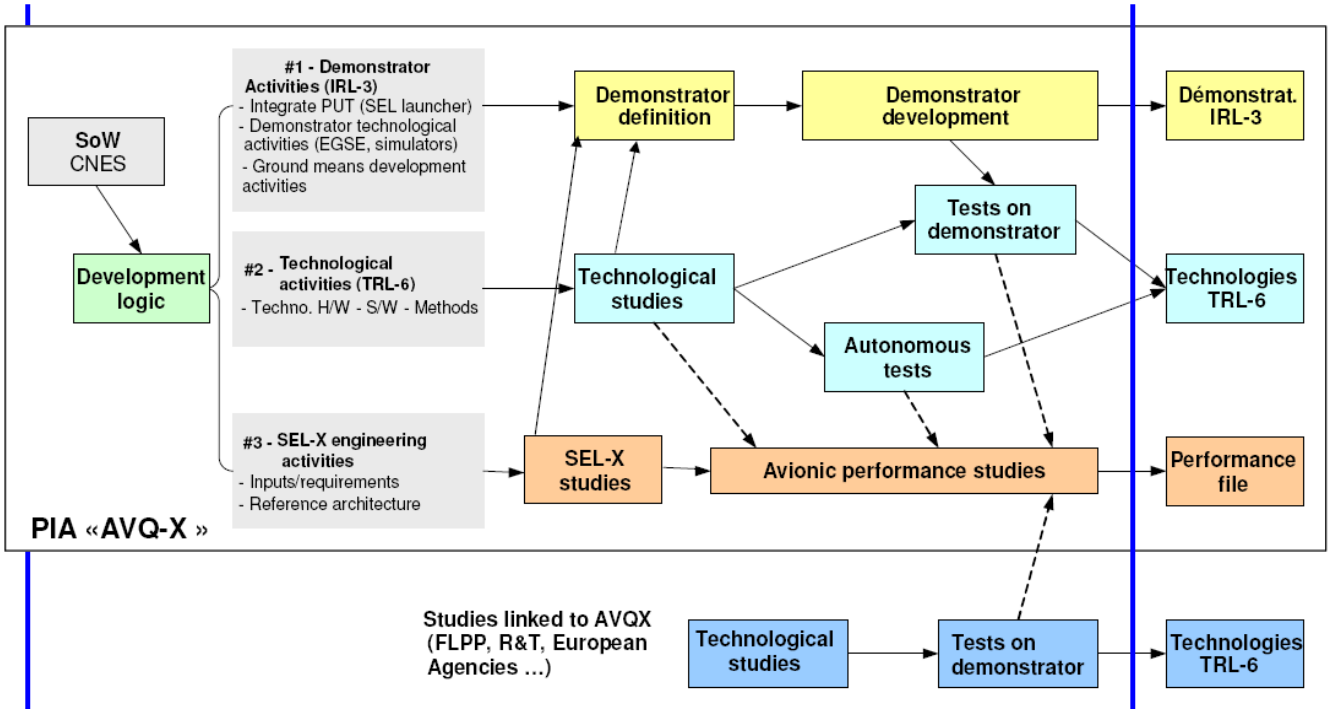

Figure 2. Work Logic of the Avionic-X project

B. *Launcher avionics engineering ("SEL-X")*

This activity branch of the project aims at identifying the key drivers and requirements for a launcher avionics system, and following a top-down approach, taking into account the technological candidates, in order to propose several possible avionics architectures (named "SEL-X"), and assess their performances and cost.

But first, to prepare the future, we have to look back at the past. So let's take a look at the Ariane 5 launcher family. Ariane 5 avionics (see figure 3) follows three main drivers: It is a duplex system, it is organized in several sub-systems (Flight Control, Telemetry, Electrical Power and Flight Safety Sub-Systems), and it is also strongly constrained by the rule of "geographical return" (which may have led to a practical but non-optimal architecture).

Ariane 5 avionics has evolved since the beginning of Ariane 5 program, in order to solve obsolescence problems or to accommodate the various versions of the launcher, but its duplex architecture stays essentially the same. The duplex choice was not derived from system requirements but was seen as a way to increase robustness and ensure reliability.

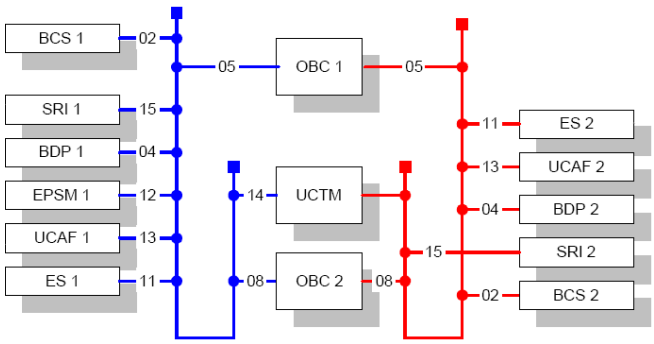

Figure 3. Overview of Ariane 5 upper composite avionics architecture

Ariane 5 avionics system breakdown in sub-systems will not necessarily be retained, and for the moment, geographical return constraints should be put aside, in order to look for "more" optimal solutions.

The trade-offs performed on the SEL-X architectures led us to consider several potential improvements: First, a stronger integration (in order to reduce the number of equipments, and the recurring costs), with standard modules

in a rack-based architecture (following the IMA trend) and use of Time and Space Partitioning; second, a hot redundancy concept, and a communication based on a time-triggered protocol (see § III.C.1)).

1) Toward a more integrated architecture (IMA)

A “centralized” architecture is an architecture where the main tasks are executed on the same processing unit. In a “distributed” architecture, the tasks are distributed through several equipments.

The current Ariane 5 avionic architecture is mainly centralized, as there is an On-Board Computer (OBC) centralizing all the sensor data to compute the GNC and the sequential algorithms for commanding the launcher. However it is also partially distributed as other equipments also have “intelligent” functions to perform, before or after the OBC computations. The architecture is organized on a “master-slave” mode, meaning that the centralized OBC takes all the decisions, based on the sensor inputs and measurements. This was a way to ensure the determinism required for a high level of reliability and availability.

The choice of Ariane 5 was the result of a top-down allocation of the main avionics functions to different equipments, with clear industrial perimeters and responsibilities, accepting the fact that each industrial would probably have to develop or to implement computing means in an heterogeneous way and without any harmonization.

In general this was not a problem as most of the equipments only needed very simple functions, easy to implement within simple components such as DSP, ASIC or FPGA. But for complex equipments like the SRI (Inertial Measurement Unit), the computing unit could have been harmonized or merged with the OBC itself.

Today, the evolution of CPU capacity allows us to envision a stronger centralization (or integration) of the architecture, for example by merging the SRI and OBC numerical calculations on the same computing node, in order to reduce the number of equipments and the associated weight.

We reach here the concept of Integrated Modular Avionics, which comes from Aircraft development. The main idea is to reduce the amount of embedded avionic hardware by sharing the same hardware and software resources between various sub-systems. In this concept, a standardized CPU modules would allow to reuse the same module for all the launchers functions requiring computing capabilities.

This standard Processing Module would be physically regrouped with other standardized or specific modules in racks, as it has been done in the satellite industry (for example the Spacebus 4000 avionics concept described in [5]). Within Avionic-X we call this standard Processing Module “MDHB-X”, which stands for “Modular Data Handling Block” and is presented in § C.2).

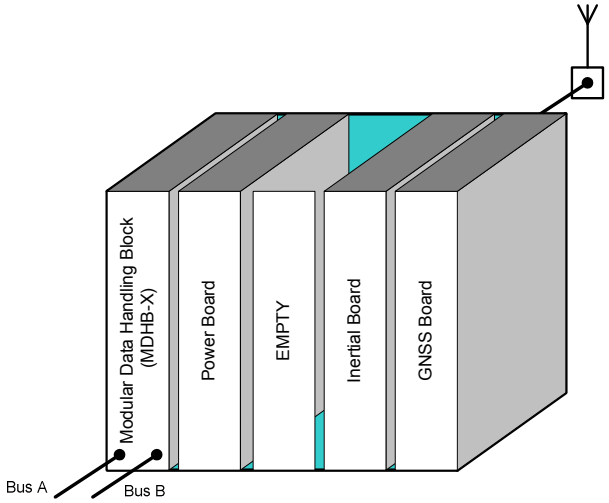

Figure 4. Example of a rack-based architecture

2) Redundancy concept

The current Ariane 5 system is based on a duplex architecture composed of two onboard computers in hot redundancy. The nominal one is in charge of all operations on both avionics chains and the redundant one only spies a subset of the 1553 messages to maintain its own flight context. In case of auto-detected errors on the nominal OBC, an error signal is sent and a hand-over is done to the redundant OBC which takes the control of the 1553 buses and continues the mission. The redundant OBC becomes “the last survivor” and the nominal OBC cannot recover the flight control.

The first redundancy concept which we aim at testing on Avionic-X is not dissimilar to Ariane 5’s one, but improves the transition phase and simplifies the design of the flight software. As a matter of fact, in this concept both OBCs execute the same software in parallel, and the selection of the command which shall be executed is performed by the actuator. The failure of one OBC is then almost transparent, hence a true “hot redundancy”. The reinsertion of the faulty OBC could also be envisioned, in case of a transient failure followed by a reset and successful auto-tests.

This concept of “selection by the actuator” could eventually be scalable to a triplex concept, for specific missions where the exposition to natural radiative environment could otherwise lead to unacceptable unreliability figures.

C. Technological enablers

The second activity branch of the project aims at identifying innovative technological candidates (hardware, software, methods), performing trade-offs and feasibility studies w.r.t. a launcher system, and elaborating roadmaps to mature the relevant technological enablers up to TRL 6.

The technologies we are looking at in the frame of the Avionic-X shall cover the biggest part of a launcher avionics

functional needs. At this stage of the project, all trade-offs are not completed yet, so we will detail only a few of them hereafter, focusing on ERTS² targets:

1) *Communication System*

The Communication system, which is the "backbone" of the digital system, comes naturally first, as it will support the integration of every other building blocks on the demonstrator in the next increments.

This work package has included several trade-offs, w.r.t. bus technology (optical, cable, wireless…), bus topology (star, ring…), communication protocols, field buses.

Moreover, it appears interesting to follow the evolution in communication systems from a message-centric model to a data-centric design. In a message-centric architecture, the communication system does not recognize the data embedded in the messages it carries. In opposition, a data-centric design could be represented as a global virtual database, with applications accessing it without worrying about the distributed aspect of the system. The "smart Telemetry" concepts follow this data-centric design too. For example, a sensor may acquire data at 100Hz, but only updating the virtualized data when the change is bigger than a predefined limit, thus avoiding bottleneck points in the communication system.

The communication system has to answer to criteria defined by the architecture of the new launcher. The first SEL-X studies identified the following targets:

- At least 50 or 100 times the Ariane 5 communication system throughput in order to fit the necessary increase of command/control data flow due to the "hot" redundancy flight control concept suggested in SEL-X (see § B.2)), and moreover, to host the launcher telemetry data flow which is managed until now on Ariane 5 by a dedicated bus.
- A better exchanges determinism, ensured as must as possible at standard level, firstly to avoid a costly development of a "tailor-made" deterministic layer and also to simplify the hot redundancy management and synchronization between the communication system nodes.
- Reduce the communication system harness weight,
- Reduce the electric consumption,
- Increase Communication System flexibility.

The current state of the preliminary design includes at least three main communication systems on the demonstrator:

- One copper bus with a time-triggered Ethernet communication, for example TTEthernet. With a throughput of 100 Mbits/s (possible 1Gbits/s) based on Ethernet (LAN compatible), TTEthernet manages determinism at standard level by offering Time Triggered services based on PTP IEEE 1588 protocol, the TT messages. This standard also takes into account Rate-constrained (RC) messages like ARINC664 (AFDX) and is also compliant with basic Ethernet messages, called Best Effort (BE) in the standard. It benefits from the long Ethernet experience, tools and has already been chosen in space projects. Here is a purely theoretical view illustrating the three kinds of messages managed by the TTEthernet standard:

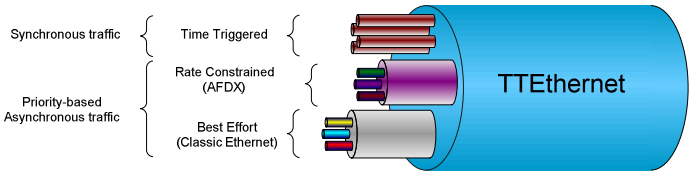

Figure 5. Different kinds of TTEthernet traffic

- One optical bus, for example FibreChannel. The main benefits of using optical fiber at physical level are an increased bandwidth and an immunity to Electromagnetic Interference (EMI). Thus, the harness weight could decrease due to wires shielding reduction and the uselessness of EMC filters at board level. Moreover, FibreChannel has been crafted to easily map other protocols, so that existing software written for legacy protocols can be easily ported. Determinism and fault tolerance may be mapped to Fibre Channel. It is compatible with 1553 (but without the limitations of the MIL-STD-1553 standard in terms of throughput, message size,…) and it offers a low latency network.
- One MIL-STD-1553 bus, which would ease the connection of existing equipments on the demonstrator. It offers industrial partners a well-known space bus to plug possible off-the-shelf demonstrations using this standard.

2) *Modular Data Handling Block (MDHB-X)*

In this branch of the Avionic-X project, we will define and demonstrate the MDHB-X solution for data processing and communication bus interface (such as identified in § B.1)), while following several trends:

- IMA and TSP for space applications;
- Multi-core architectures.

IMA changes drastically the usual industrial way of working: Sub-system suppliers are no longer the suppliers of the avionics part of their sub-system (that would be generic bricks), and could perhaps become only "application software providers".

As for the multi-core aspects, we have to prepare the use of multi-core processors in space systems. Multi-core architecture is a solution to introduce additional processing power, to improve the fault detection isolation and recovery (FDIR), and finally to implement more easily time and space partitioning.

Taking this into account, we will define and demonstrate a generic processing and Input/Output module answering different needs for data handling in a launcher. This "Modular Data Handling Block for Avionic-X", or MDHB-X, is presented on the figure below.

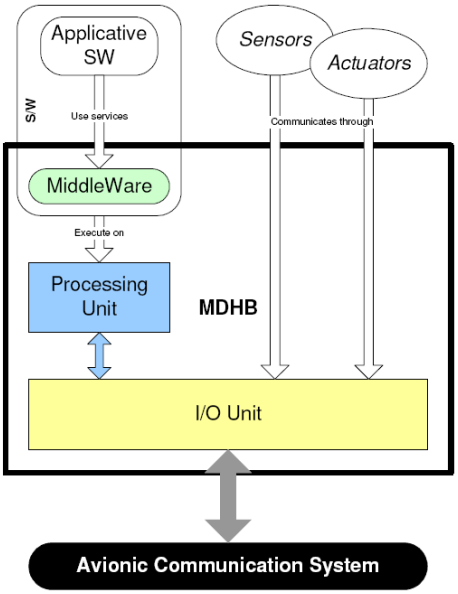

Figure 6. Modular Data Handling Block for Avionic-X (MDHB-X)

The Modular Data Handling Block (MDHB-X) aims at providing a generic, versatile and configurable computing system solution. Each MDHB-X unit can be made of various predefined combinations of Processing Modules (PM) and I/O Modules (IOM). Through this mechanism, it is possible to create a wide range of MDHB-X units thanks to PM and IOM generic building blocks.

Here are shown some configurations of the MDHB-X:

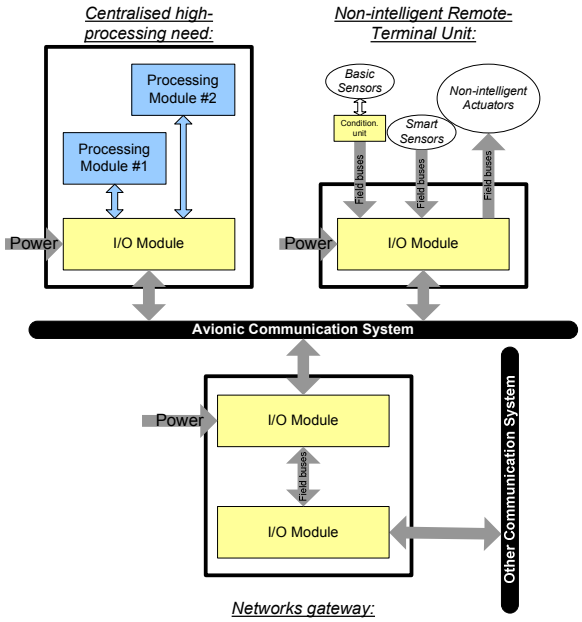

Figure 7. Various configurations of the MDHB-X

Several candidates are foreseen to be assessed as processor, like the LEON4-based NGMP platform, or some ARM implementation, both potentially in single or dual core configuration. The IOM will lie on a custom FPGA solution.

The different I/Os to handle include for instance the communication system and sensors ("intelligent" or not, and accessed through conditioner unit or directly).

Each Module of MDHB-X units will lie on a separate board, and the integration could be done thanks to a rack, connecting them together through a backplane bus for example.

From a software point-of-view, the spaceflight field is on the verge of an evolution from federated avionics and data handling architectures to Integrated Modular Avionics (IMA) paradigm, as it happened a few years ago with airborne systems. By allowing running several previously segregated pieces of software on the same computing platform, IMA in space applications is expected to have an impact on the overall avionics' mass, volume, power consumption and recurrent cost.

As the physical boundaries of federated subsystems disappear, IMA has to implement a new kind of separation between the different pieces of software involved, through the technique of *partitioning*.

In the frame of Avionic-X demonstrator, the Middleware layer represents every piece of software between the bare hardware and the applicative software, including the partitioning layer (inside or outside the RTOS) and the OS.

Here is a scheme of the Middleware in Avionic-X:

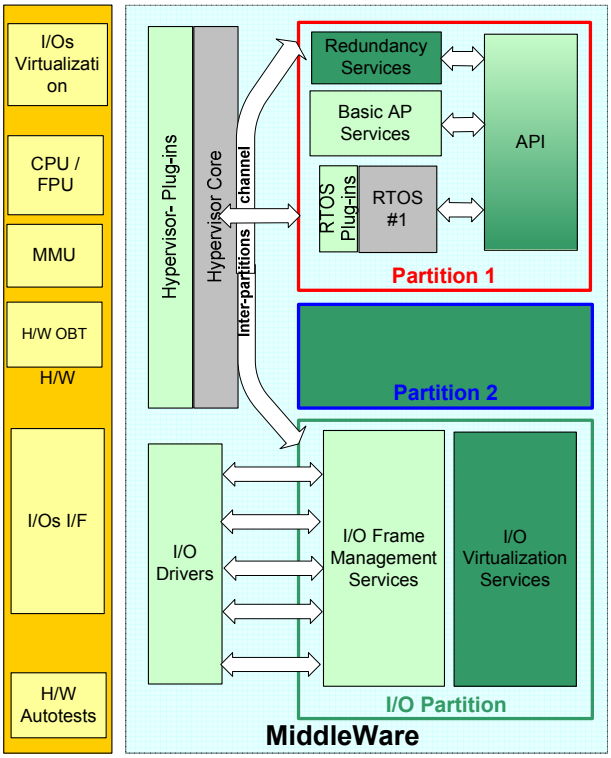

Figure 8. Middleware in Avionic-X

In partitioning-based software, like in ARINC-653 systems for instance, isolation, protection and determinism lie on the concept of Time and Space Partitioning (TSP). It allows a strong segregation of pieces of software of mixed criticalities and/or mixed suppliers by implementing a static allocation of fixed amounts of memory for each partition, and a fixed amount of CPU time allocation of each partition according to a cyclic preemptive execution scheme. In this light, the MDHB-X represents the physical implementation of IMA.

Actually, the market offers a wide panel of partitioning solutions, from bare metal hypervisors (or Virtual Machine Monitors) to *userspace* RTOSes over partitioning microkernels, and care must be taken when choosing a suitable option depending on the project. XtratuM hypervisor is currently foreseen, as well as in CNES and ESA studies.

Another form of abstraction mechanism used in IMA system is the concept of *Input/Output virtualization*.

This consists in presenting to every software instance the same abstracted and generic interface to manage I/Os by isolating high-level software and the communication system, having available each piece of data at each node of the system, solving equipment synchronization problematic, supplying data consistent with the equipment need (not only the last update), reading an entire engineering data like a quaternion (without update during acquisition) and ensuring data consistency.

The resulting generic layer will be usable on every piece of equipment (i.e. MDHB-X).

3) *Other technological enablers*

The other work packages and technologies we are looking at in the frame of the Avionic-X project are not fully detailed in this paper, but they encompass:

- Flight Control:

Within Avionic-X we will explore different inertial navigation sensors (e.g. hemispherical resonator gyro or fiber-optics gyro), which would be less expensive than the gyrolaser technology used on Ariane 5. GNSS hybridization will also be considered, in order to improve the navigation precision or to compensate for less precise gyros, and to provide real-time localization for safeguard purposes.

MEMS gyrometers distributed along the launcher will be considered in order to improve the robustness of Thrust Vector Control. As a matter of fact, vibrating structure gyroscopes manufactured with MEMS technology have become quite inexpensive and widely available.

- Pyrotechnics:

Both opto-pyrotechnics and advanced electro-pyrotechnics will be assessed during the Avionic-X project.

- RF communications:

In order to improve the RF links between the launcher and the ground means (higher bit rate with less energy consumption), we will test and mature directive antennas, either phased array antennas or active antennas on innovative materials.

- Power Generation and Distribution (Digital Power Control);
- Data acquisition and sensors;
- Ground System – Onboard Interfaces (electrical and radiofrequency parts of the Ground to Launch Vehicle Interface);
- EGSE and Simulators;
- Harness and connectors, taking into account that the communication and power supply harness of a typical Launch Vehicle electrical system represent more than 10 kilometers of cable of various types !

D. *Methods: A focus on Model-Driven Engineering*

1) *The MDE approach*

The MDE approach is meant to increase productivity by maximizing compatibility between systems (via reuse of standardized models), simplifying the process of design (via models of recurring design patterns in the application domain), and promoting communication between individuals and teams working on the system (via a standardization of the terminology and the best practices used in the application domain).

MDE reduces costs, in particular hidden costs or cost overruns, not foreseen at the beginning of a software project. The famous “Chaos Report” (an industry study by the Standish Group) [1] is nowadays contested, however it found that for IT (information technology) projects, the average cost overrun was 43 percent, and 71 percent of projects were over budget.

Moreover, according to [4], available statistics on bugs in embedded systems show that approximately 75% of them are caused by ambiguities or misunderstandings between system requirements and software requirements. Moreover, such errors are generally found late in the project life, thus are particularly expensive to correct [5].

In the frame of the European Project ASSERT (see [2]), it has been estimated that a gain of 10% is achievable in terms of productivity during software engineering, due to:

- the use of formal modeling, proof and verification at system level,
- data modeling and code generation techniques.

[5] Original citation in French: *« Les statistiques disponibles […] sur la cause des bugs dans les systèmes embarqués montrent qu’environ 75% de ceux-ci sont lies à des ambigüités ou des divergences de compréhension entre spécifications systèmes et spécifications logicielles. Ce type de bugs a pour circonstance aggravante d’être généralement trouvé très tard dans les projets et donc d’être particulièrement coûteux à corriger ».*

The MDE covers the whole range from software-intensive systems to on-board code, and relies on the use of various modeling languages, which are presented in § D.2).

The following figure shows a typical development cycle for a space transportation system, using a MDE approach:

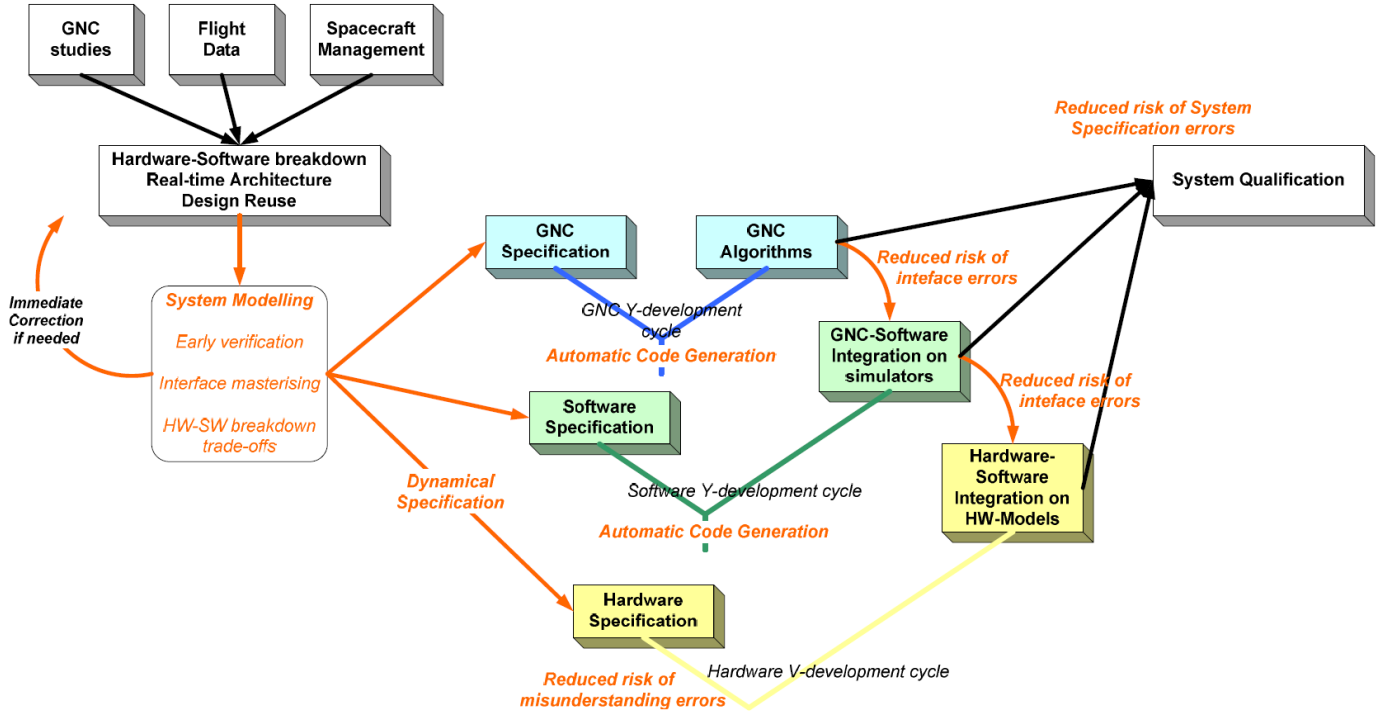

Figure 9. Model-Driven Engineering process for a space transportation system

2) Software intensive system Engineering

In launchers systems, the role of the software is more and more important in functional chains. Furthermore, on one hand the embedded software becomes more complex and harder to master. On the other hand, the introduction of IMA (Integrated Modular Avionic) principles in avionics architecture implies new software rules: use of TSP (Time and Space Partitioning), distributed software architecture...

It is essential to ensure the consistency of the software engineering all over the system development. Another very important aspect is to limit the risk and anticipate the potential problems which could be faced during the development: early validation shall also be an objective. To fulfill these objectives, it has been decided, in the context of the Avionic-X project, to define a Model Driven Engineering (MDE) process to support the embedded software development.

The advantages of using models can be presented as below:

- Consistency improvement:
  - Models are more formal than hand written document (misunderstanding limitation),
  - Models can be analysed to check development rules, to verify properties (software verification process improvement)...
  - Model transformation using tools instead of manual transformation from documents.
- Early validation:
  - Models can be executed to verify behaviour of the system,
  - Models can be used to ease the build of validation tests.

As shown in the ECSS-E-40 [6] following overview (see figure 10), the planned MDE process will cover all the software development activities from the “software related system requirements process” to the “validation w.r.t Technical Specification activity”. In addition to the following figure, another activity is essential: the “System data requirement” which consists in the management (all over of the system development) of all data exchanged between the different parts of the software intensive system.

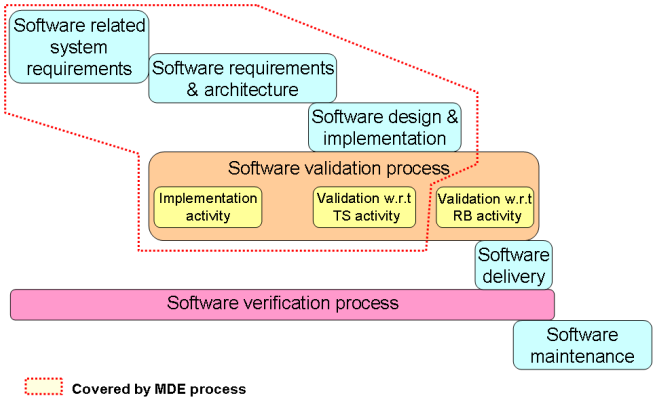

Figure 10. Software engineering process overview

For each activity, the following models are foreseen:

- System data requirements:

  System related data model that will represent all the data and interfaces types in a common language (currently an extension of ASN1.0 is envisaged). This model is refined by all teams involved from their unit/dimension until their software interface implementation (taking into account the different level of implementation: equipment software, communication protocol, low level software, applicative software).

- Software related system requirements

  Software-intensive system SysML model which describes the static architecture: definition of functional blocks, interfaces (data flow and control flow), mission data and the non algorithmic behaviour: mode automata, time line, activation condition, activation frequencies and acyclic actions… this model is independent of the chosen system architecture (redundancy policy, processing units …). It corresponds to the System Software Specification (SSS) in the ECSS wording.

  Interface requirement Document is automatically generated from a refinement of the System related data model.

  Software-intensive system design model which describes the processing units, the avionic communication means, the physical equipments, the software partitions, redundant equipments, data flows and their associated latencies, the allocation of functional blocks to partition, the partition allocation to processing units (AADL [6] is currently envisaged to describe this model). This model corresponds to a part of SRS in the ECSS-E-40 wording.

- Software requirements & architecture

  Software static architecture model which describes the static decomposition (in terms of software components/objects) of the software. This model will use UML for manual code and Scade for code that will be automatically generated. Interfaces between the two formalisms will be ensured using the ASN1.0 language (and automatically generated wrappers).

  Software-intensive system detailed design model which is a refinement of the previous system design model. It contains the thread definition, subprograms (interfaces), data flow timing constraints, worst case timing (bus and subprogram estimations).

  Interface Control Document is automatically generated from a refinement of the System related data model.

- Software design & implementation

  The implementation will be made using Automatic Code Generation (ACG) for interfaces (code skeleton) from UML model, and Scade models are refined until containing all the functional aspects (to be able to generate a final flight code).

- Validation & Verification related to implementation and models

  One of the goals of all these modelling activities is also to provide early validation and/or automatic replay/generation of (early) validations activities. The following list is far from exhaustive:

  - *Software-intensive system SysML model*: use of simulation with a system granularity. Automatic generation of validation tests from sequence diagrams.
  - *Software intensive system detailed design model*: automatic generation of on-board software scheduler (or configuration) and for numerical validation software, TSP configuration, verification of all the worst case timing information (data flow latencies, worst case execution times …)
  - *Software architecture models*: automatic generation of integration tests.
  - *Software design and implementation*: unit test & coverage performed at model level for Scade models. Use of formal proof (ADA 2012) to replace some unit test for (suited) manual code.

- Models consistency

  The Avionic-X models shall ensure some consistency between their different views. Part of the work is made by the refinement of the data interfaces defined in the system related data model. But it is far from sufficient.

[6] AADL stands for “Architecture Analysis & Design Language”, but it is interesting to remind that at the beginning, AADL stood for *Avionics* Architecture Description Language.

Intra-model consistency will be verified using OCL rules for AADL, UML, SysML models. Scade models will be verified using in-house TCL scripts.

Inter-model consistencies have many different means to be verified. Most of them will be assured by using scripts which will import a version of the interface and data types. For instance, an Acceleo script can be written in order to translate an ASN1.0 data type in a UML/SysML/AADL data type.

- Others Models

In parallel to these models dedicated to on-board software development, the MDE activity in Avionic-X will also cover following modelling: Data handling system with SystemC (data handling system specification: used for specification verification and simulator generation (for applicative software development support)); Equipment Simulator Software Architecture using UML and ASN1.0 interfaces (for functional closed loop validation of the on-board software); Algorithm Prototyping Models using in-house common formalism (for a faster and safer translation of prototype to on-board software).

IV. AVIONIC-X DEMONSTRATION PLATFORM

The last activity branch of the project shall cover all platform needs in order to be ready for the first test phase at the end of the first increment, and shall manage the various demonstrator configurations, deriving from the other two activity branches (SEL-X and technological enablers, see part III).

The key properties of this demonstrator are modularity, openness and connectivity. It will keep evolving, and will be able to host different demonstrations (unitary or integrated). The "virtual layer" (shown at the bottom of figure 11) and the simulators will allow several testing configuration, from SWIL to HWIL.

In the Statement of Work, the overview of the platform was drafted as follows:

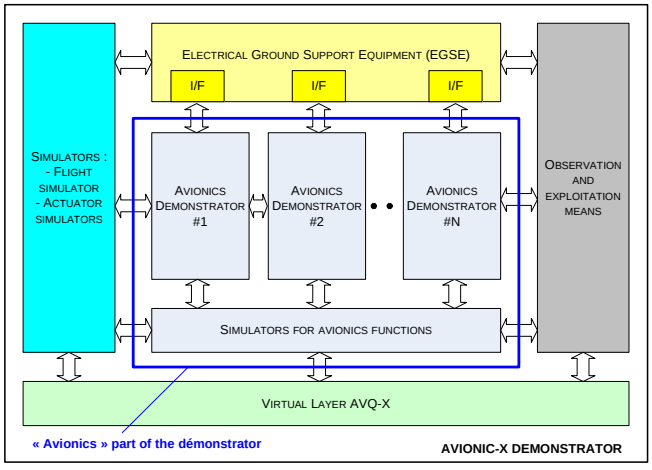

Figure 11. First draft of an overview of the Avionic-X demonstrator

Then we have been fine-tuning the requirements and objectives during the feasibility phase, and a first set of "platform means" has been written down, as shown on figure 12, which gives an overview of the global concept of Avionic-X demonstrator.

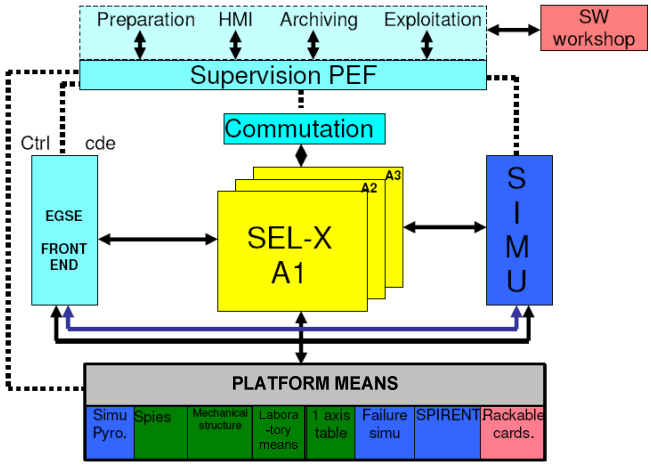

Figure 12. Overview of the Avionic-X demonstrator

The EGSE is mainly composed of the following sub-systems:

- The test supervisor based on the AITS (*Advanced Integration and Tests Services*) product which offers the user interface to control the tests and commanding activities;
- The front-ends equipments which interface with the product under test (Time server, Telemetry, Power supply, Avionic buses I/F…).
- The matrix connections whose objective is to dispatch the signals between the front ends and the product under test taking into account the SEL-X configuration;
- The avionic simulators interface with the test supervisor;
- and specific check-out equipments if necessary.

The following figure shows how the three activity branches of the Avionic-X project interact continuously in order to reach our objectives of TRL and IRL, and build performance files for various launcher avionics architectures, based on test results and benchmarks.

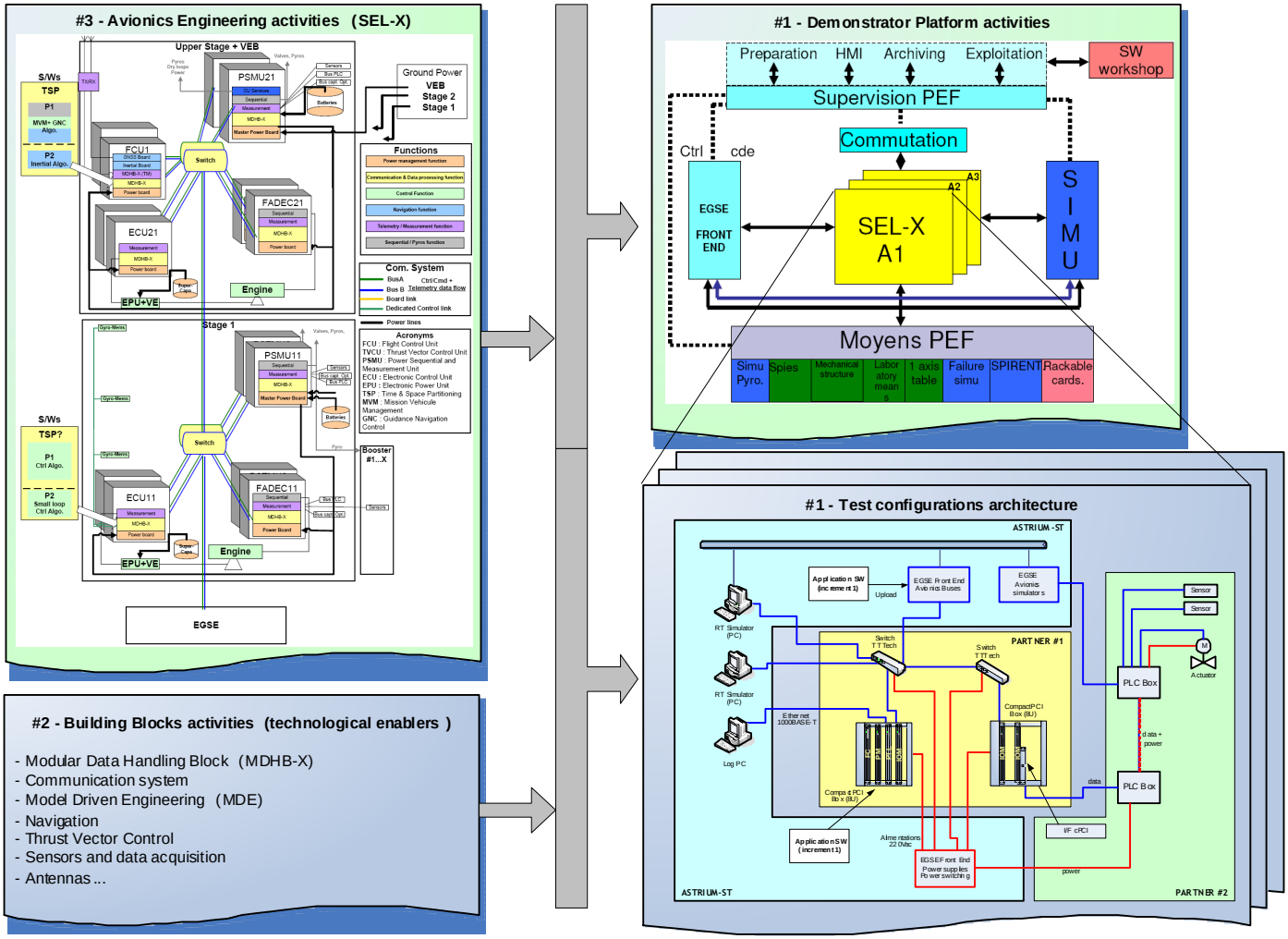

Figure 13. Links between the three activity branches of Avionic-X

V. CONCLUSION

The Avionic-X project covers a wide range of technologies and avionics functions. It will provide a test bench to integrate innovative technologies. Its complementarities with other existing programs like FLPP will allow to contribute effectively to future launcher developments. It was initiated in the frame of the French PIA, but it remains open to the European partners involved in Avionics systems.

The development plan is foreseen to adopt iterative cycles in order to integrate new technology demonstrations which are not necessarily identified yet. This flexibility will allow, up to 2013, to associate new partners who would propose innovative solutions for launcher avionics.

VI. REFERENCES

[1] Standish Group (2004) CHAOS Report. West Yarmouth, Massachusetts: Standish Group. (Report).

[2] The assert-project, 2008, European Space Agency. http://www.assert-project.net.

[3] Avionics Application Software Standard Interface (ARINC-653), AEEC (Airlines Electronic Engineering Committee).

[4] Rapport de D. Potier sur les Briques Génériques du Logiciel Embarqué, 7 octobre 2010

[5] SPACEBUS 4000 Avionics : key features and first flight return. J-M. Pasquet, 24th AIAA International Communications Satellite Systems Conference (ICSSC) - June 2006

[6] Space Engineering – Software, ECSS Standard n°ECSS-E-40 (European Cooperation for Space Standardization)

VII. GLOSSARY

AADL	Architecture Analysis & Design Language
ACG	Automatic Code Generation
AFDX	Avionics Full DupleX switched ethernet
AFRL	US Air Force Research Laboratory
AVQ-X	Avionic-X
CNES	Centre National d'Etudes Spatiales
DAL	Design Assurance Level (according to EUROCAE ED-12B / DO-178-B standards)
DDASCA	Distributed Dependable Architectures for Safety-Critical Applications
EGSE	Electrical Ground Support Equipment
EMC	ElectroMagnetic Compatibility
EMI	ElectroMagnetic Interference
ESA	European Space Agency
FDIR	Failure Detection Isolation and Recovery
FLPP	Future Launchers Preparatory Program
GNC	Guidance, Navigation & Control
GNSS	Global Navigation Satellite Systems
GSTP	General Support Technology Program
HWIL	Hardware-In-the-Loop
IMA	Integrated Modular Avionics
IMU	Inertial Measurement Unit
IOM	Input-Output Module
IRL	Integration Readiness Level
LV	Launch Vehicle
MDE	Model-Driven Engineering
MDHB-X	Modular Data Handling Block
MEMS	MicroElectroMechanical Systems
NGL	Next Generation Launcher
NGMP	Next Generation Multi-Purpose Microprocessor
OBC	On-Board Computer
PIA	Plan d'Investissement pour l'Avenir
PM	Processing Module
POA	Power Optimized Aircraft
R&T	Research and Technology
RTOS	Real-Time Operating System
UML	Unified Modeling Language
SAVOIR	Space AVionics Open Interface Architecture
SAG	SAVOIR Advisory Group
SIL	Safety Integrity Level (IEC/EN 61508)
SPA	Space Plug-and-Play Architecture
SWIL	Software-In-The-Loop
SysML	Systems Modeling Language
TCO	Total Cost of Ownership
TRL	Technology Readiness Level
TRP	Technology Research Program
TSP	Time and Space Partitioning