INTEGRATING SYSTEM AND SOFTWARE ENGINEERING FOR CERTIFIABLE AVIONICS APPLICATIONS

Thierry Le Sergent
Mathieu Viala
Alain Le Guennec
Frédéric Roméas
thierry.lesergent@esterel-technologies.com
mathieu.viala@esterel-technologies.com
alain.leguennec@esterel-technologies.com
frederic.Romeas@eurocopter.com

Esterel Technologies France
Eurocopter France

INTRODUCTION

Avionics systems are complex systems that integrate hardware, communication media, have many interactions with other subsystems within or outside of the aircraft, and, for the system discussed in this presentation, integrate software that must be developed according to DO-178B guidelines. System engineering and software engineering are two engineering disciplines that are historically handled by teams with different cultures, and when their engineering processes are supported by tools, use different, incompatible, tools. This very often leads to difficult collaboration, with at some point redundant information, and inconsistencies.

This presentation introduces a solution, based on standards, SysML for system modeling and on the renowned SCADE Suite® product from Esterel Technologies for the development of DO-178B certified software components. This solution, named SCADE System™, allows system and software engineers to work with the right formalism for their respective domains, but improve cooperation because of:

- A unified tool framework, allowing technology sharing for API based scripts for e.g. cross verifications between System and Software models,
- The same requirements traceability and documentation generation tools,
- A synchronization mechanism for the data that are at the frontier between the two engineering domains.

This solution can be the basis to develop systems that have to adhere to both functional standards, such as ARINC 653 (IMA) or ARINC 661 (CDS) and certification standards ARP 4754 (Systems), DO-178B (Software), or DO-297 (IMA).

The presentation details, in particular, how this solution can be applied in the system and software engineering processes evolving in parallel in industrial avionics projects, and how the work in the system and software engineering can be synchronized.

SYSTEM ENGINEERING

Aerospace guidelines

Good practices in System engineering and Software engineering are detailed in aerospace guidelines ARP 4745 / ED-79 and DO-178B / ED-12B.

Figures Figure 1 and Figure 2 below highlight the global picture; this paper focuses on relationships between the System Development Processes and the Software Development Lifecycle. The System Development Processes provide in particular:

- The System Requirements allocated to Software
- The description of the System Functions (what the system shall do)
- The description of the System Architecture

From these, the Safety Assessment Processes provides the FDAL (Function Development Assurance Level) assignment to the Software components, as pictured figures Figure 3 and Figure 4.

The Software Development Life-Cycle (DO-178B) starts from the System Requirements allocated to Software and the FDAL assigned.

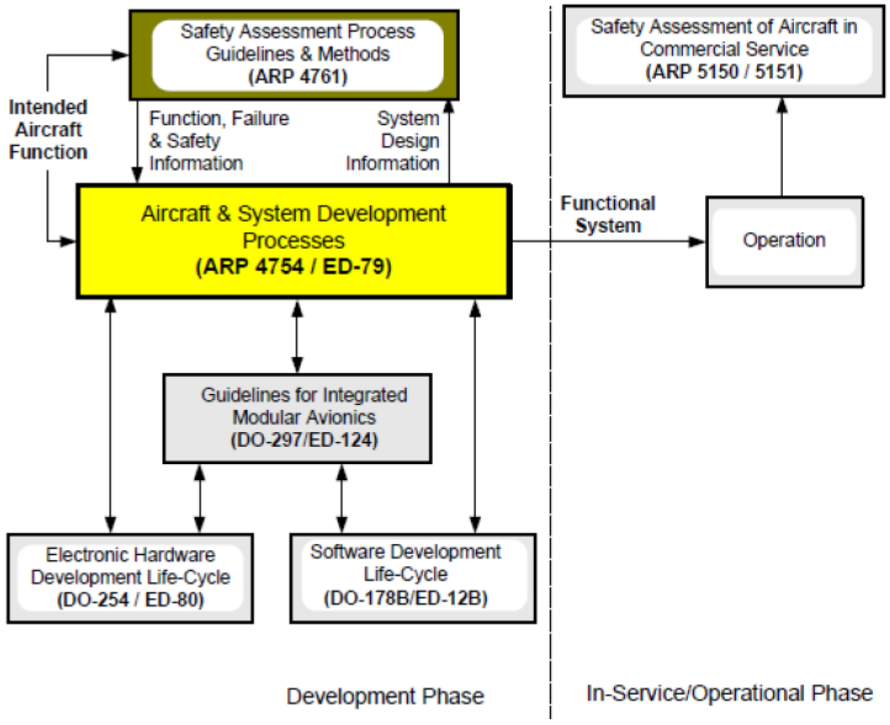

Figure 1 – ARP 4754 – FIGURE 1

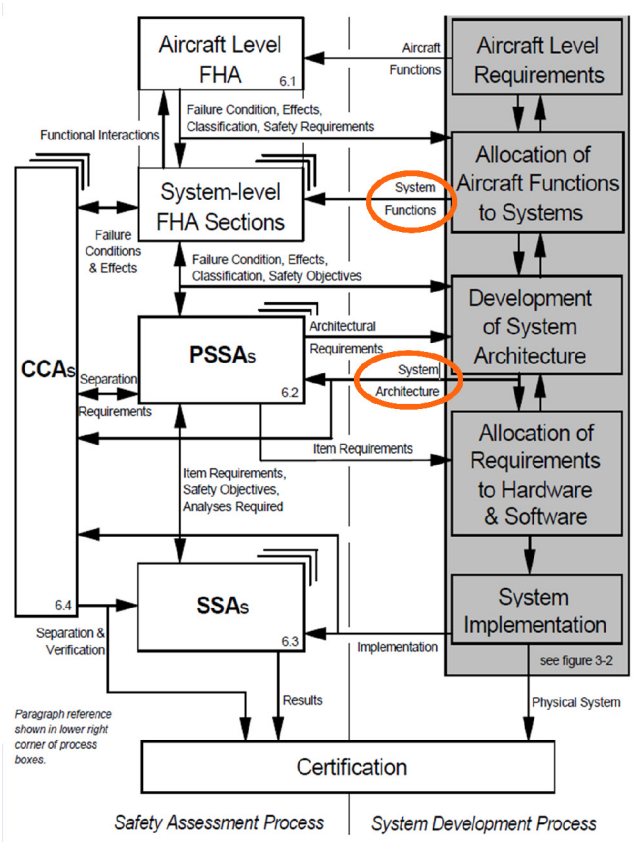

Figure 2 – ARP 4754 – FIGURE 7

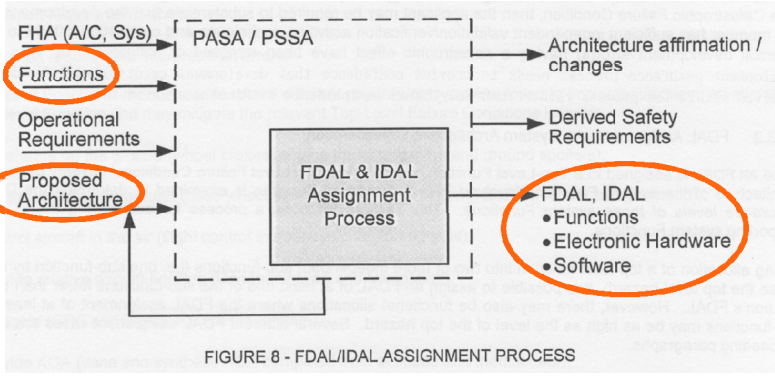

Figure 3 – ARP 4754 – FIGURE 8

TABLE 2 - TOP-LEVEL FUNCTION FDAL ASSIGNMENT

Top-Level Failure Condition Severity Classification	Associated Top-Level Function FDAL Assignment
Catastrophic	A
Hazardous/Severe Major	B
Major	C
Minor	D
No Safety Effect	E

Figure 4 – ARP 4754 – TABLE 2

System Engineering Toolset

From the guidelines quickly summarized above, the first requirements for a toolset supporting the System Engineering activities are:

- Functional decomposition of the high level system requirements
- Description of the organic architecture of the system
- Allocation of the functions to the architecture components
- HW or Software implementation choice for the system components

Additional requirements include:

- Generation of complete documentation of the outcome of the above activities
- Graphical description, as usual means of communication for engineers
- Seamless path to SW development tools
- Seamless path to verification tools

The SCADE System tool detailed in this paper fulfill all above requirements.

The way the tool is used to follow the required activities is then detailed on a use case.

OMG SysML and Modeling Tools

OMG's SysML is becoming the standard in system engineering. Adhering to the standard is important; it allows, to a certain extent, exchange of models between tools from different tool vendors, and students trained in engineering school are more quickly operational within companies, etc. One inconvenience, as often reported by system engineers, is the complexity of the tools produced by the UML world. Indeed, although SysML aims to provide a simplified notation targeted to system engineering, the fact that SysML is defined as a UML profile introduces some complexity that is not always masked to end users. SCADE System

strives to hide this UML flavor by defining a system engineer-oriented view of the tool GUI. The underlying technology relies indeed on UML and its SysML profile, thus allowing the standardized XMI persistence mechanism, but the IDE is customized to manage "pure" SysML objects allowing the user to concentrate only on system designing, without blurring his/her mind with UML considerations.

One of the nice features of SysML is the way in which the diagrams provide views of the model. The graphical aspect is indeed very important for all engineers; SysML's capability to define diagrams that are partial views of the model allows scalability to models of any size. For example, if a block has hundreds of ports, one needs not to represent all of them in one single graphical view involving the block. Still, it is also important to have the ability to handle model aspects exhaustively, for example, in documentation, for reviews, etc. The SCADE System solution relies on tabular views, managed as additional diagrams. Note that this does not introduce additional SysML model objects that would make SCADE System incompatible with the standard.

SCADE SYSTEM

SCADE System provides a model-based development tool dedicated to system engineering teams, taking into account the required capabilities and constraints listed above. Its technical foundations are the following:

- Based on OMG SysML [2];
- Relies on open technologies, supported by an active community (Eclipse, Papyrus [5]);
- Shares the same IDE framework as SCADE Suite, the toolset detailed hereafter for the support of the software development processes, so the same ergonomics;
- Completely integrated into SCADE Suite, for example using the same project and file management system;
- Features a synchronization mechanism [11, 13] between the system design model authored in SysML with SCADE System and the software design model authored with SCADE Suite. This feature is detailed in the next section.

This provides a unique and complete framework for both system and software teams, still allowing each team to use the most adapted formalism with respect to its concerns.

Figure 5 shows the resulting IDE with SysML Block Definition Diagrams and Internal Block Diagrams in the same framework as SCADE Suite diagrams. The selection of a tab in the tree window (left side) switches between the system and the software objects trees.

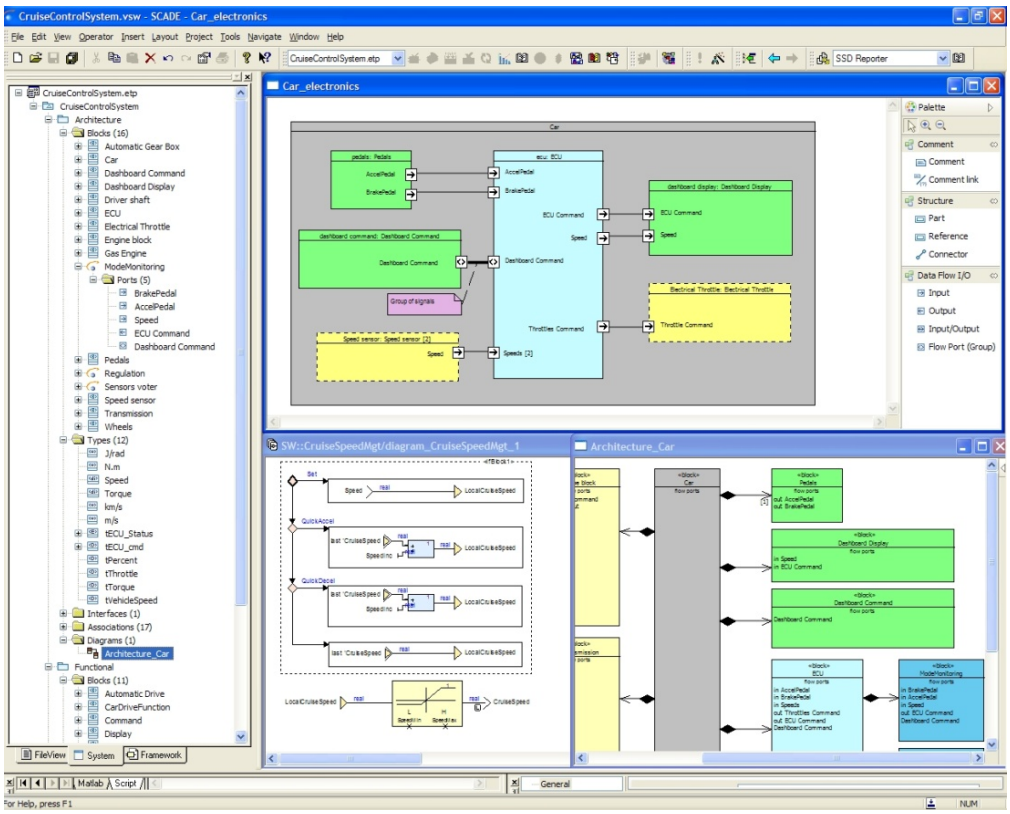

Figure 5 **– Integrated IDE: SCADE System and SCADE Suite windows**

SYSTEM TO SOFTWARE ENGINEERING

Both System Engineering and Software Engineering activities benefit from model-based technologies. But the nature of the models involved in both activities is different. For Software development process, SCADE Suite [7] from Esterel Technologies has been used successfully for the past 20 years for highly critical embedded software that must comply with the highest level of safety, DO-178B FDAL A for aerospace systems.

The solution proposed to implement the "System requirements allocated to Software" transition from the ARP 4754 System Engineering Process is to have a synchronization mechanism between SCADE System and SCADE Suite. Another dimension to take into account in that proposition is that in industrial projects both activities are running in parallel.

SCADE Suite toolset is introduced before the System-Software synchronization is detailed.

SCADE Suite

SCADE® (Safety Critical Application Development Environment) is both a notation and a toolset [8, 12] that was specifically designed for the development of critical systems in aeronautics, railway, and industrial domains that must comply with certification standards. In the rest of the paper, “Scade” is used to refer to the modeling notation and “SCADE Suite” refers to the toolset that supports the Scade notation.

The Scade notation includes both block diagrams and state machines that can be nested at any level. This powerful notation has been formally defined to provide a rigorous description of the software behavior, leaving no room for uncertainty [9, 10]. Characteristics are:

- Strong typing;
- Explicit initialization of data flows;
- Explicit management of time with clocks;
- Concurrency based on data dependencies;
- Deterministic execution.

The SCADE Suite toolset supports a model-driven paradigm in which the Scade model is the software specification.

The SCADE Suite KCG C code generator has been qualified for avionic systems with respect to DO-178B [3] at the highest level of safety (Level A) and for industrial and railways domains with respect to IEC 61508 up to SIL 3 and EN 50128 up to SIL 3/4. This, in addition to several verification tools such as SCADE Suite Model Test Coverage, provides evidence that the code that will be embedded fulfills the high-level software requirements.

SCADE Suite provides fine-grained traceability tools, consistency checks, and automatic documentation generation. They fulfill all needs for the software engineering activities but do not answer the system engineering specificities discussed above.

System-Software synchronization

As part of the System development processes shown in Figure 2, the architecture components that shall be developed with software are identified. This is done in SCADE System with a simple property associated to blocks.

SCADE System avoids duplication of efforts and inconsistencies between system structural descriptions and the full software behavioral description designed through SCADE Suite models. To that effect, a mapping between SysML block interface and SCADE Suite operator interface has been formally defined and implemented in SCADE System Synchronizer.

The synchronization action between the system and the software models shall be explicit – it is under the control of the user. Depending on the project organization, the action could be triggered either by the system team or by the software team.

Figure 6 details the complete process:

- The SysML block to synchronize is abstracted automatically with a canonical form, the "Component Specification Proxy", implemented as a SysML opaque behavior
- The SCADE Suite operator is abstracted automatically with a canonical form, the "Component Implementation Proxy" implemented as a SysML opaque behavior
- As the two abstractions are of same nature, simple diff, merge or exchange between the two proxies realize the synchronization expected.

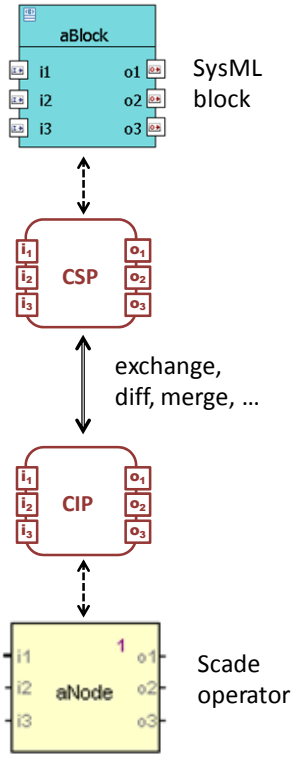

Figure 6 **– SysML-Scade synchronization.**

SYNCHRONIZATION WITH DISPLAY APPLICATIONS

This synchronization mechanism can be easily extended to other software development environments with the same philosophy. For example, the SCADE Display® [7] tool is a versatile graphics design and development environment for critical Human Machine Interfaces (HMI). It is typically used in the aerospace, rail transportation, nuclear and industrial domains for the design of critical embedded display systems such as multi-function displays, head-up displays, digital instrumentation and control panels, etc. From a WYSIWYG design entry, automatic code is generated from the SCADE Display KCG™ code generator, and takes credits from its qualification.

The display models managed by SCADE Display receive inputs to control the graphical elements, position, color, etc. Interactive HMI designs are made from primitives for active areas, multiple pointing device or keyboard events management; they provide outputs values to the SW environment interacting with the display. In short, a SCADE Display model has a typed interface exactly as a SCADE Suite operator. When a block in the system model represents a display, the input/output variables dictionary of a SCADE Display model can be synchronized with the system model as presented above for SCADE Suite.

Shared system and software engineering supported activities

In addition to the modeling and the synchronization features presented above, the following advanced features in SCADE System allow a complete industrial process from system engineering to certified embedded code generated with SCADE Suite and SCADE Display.

- Fine grain traceability feature, shared with SCADE Suite and SCADE Display;
- Semantic model diff;
- Automatic generation of documentation, shared with SCADE Suite and SCADE Display.

Another important feature provided both by SCADE System and SCADE Suite is a programmatic interface (model API) that allows implementing automated verification activities. The fact that both tools share the same infrastructure opens an easy path to verifications that could involve both the system and the software designs models.

ARINC 653 USE CASE

This section shows how SCADE System is used for a typical avionics system based on ARINC 653 IMA (Integrated Modular Architecture) architecture, by illustrating some of the main activities.

Figure 7 below illustrates the basic concept of IMA and the differences between “non IMA” and “IMA” systems, in order to introduce the development process using SCADE System.

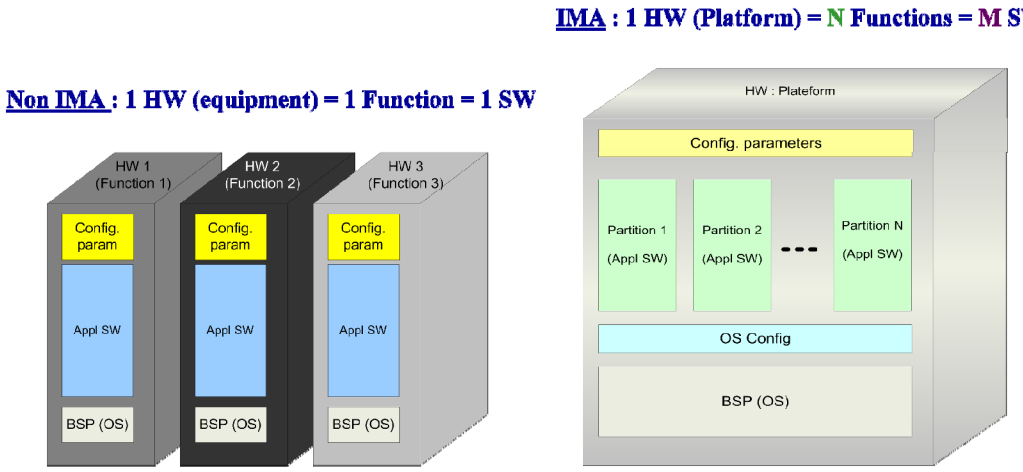

Figure 7 – ARINC 653 IMA principles

Starting from system definition, a functional analysis identifies the main functions, sub-functions and all of the necessary data between functional blocks. At this stage, only “functional data” like speed, torque, frequency, etc. are considered (no message definition nor types).

Note: This activity is independent of ARINC 653 / IMA context as it deals only with functional aspects.

These functional blocks are defined as SysML blocks, using BDD (Block Diagram Definition) and IBD (Internal Block Diagram Definition).

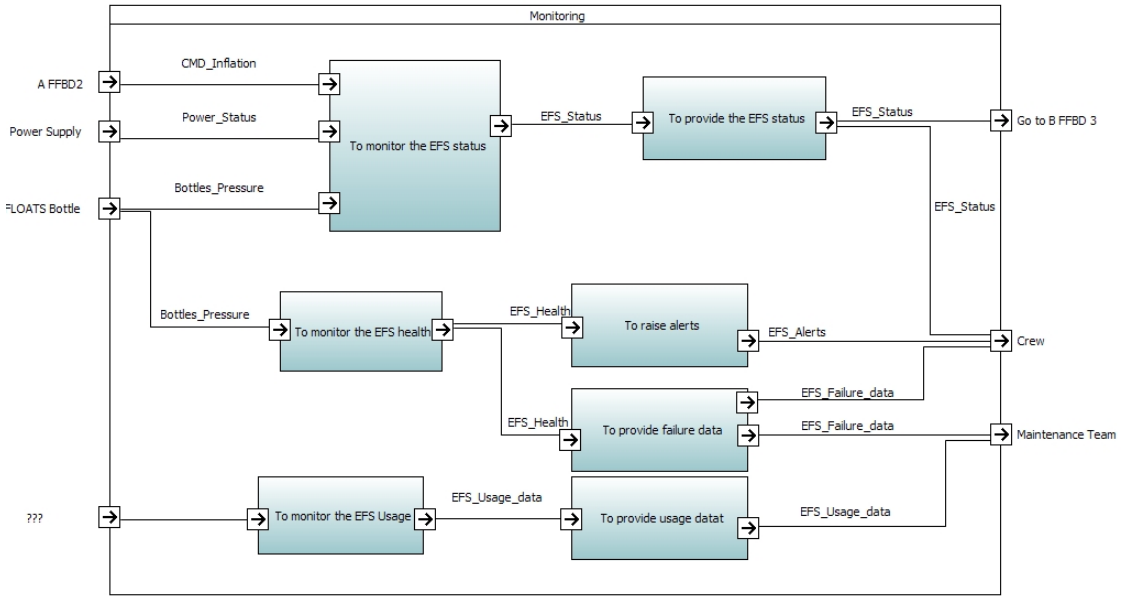

Figure 8 – Functional diagram within SCADE System

Once functional analysis has been performed, a preliminary physical architecture can be designed to identify main system parts and communication media. Then it will be refined into detailed architecture diagrams.

In a typical ARINC 653 architecture, these physical blocks can be:

- *Equipment*: Physical "HW box" that can be IMA platform or classical ECU
- *Modules*: HW board within the IMA platform running several SW partitions
- *Partition*: Application SW that implements one or several functions or only a part of a function. A partition has only "virtual ports" to communicate the other virtual ports or physical ports

The diagram in Figure 9 below is an example of the physical architecture corresponding to the functional diagram above. Each physical block implements one, several or a part of (sub)-function. The data are named and allocated to message names.

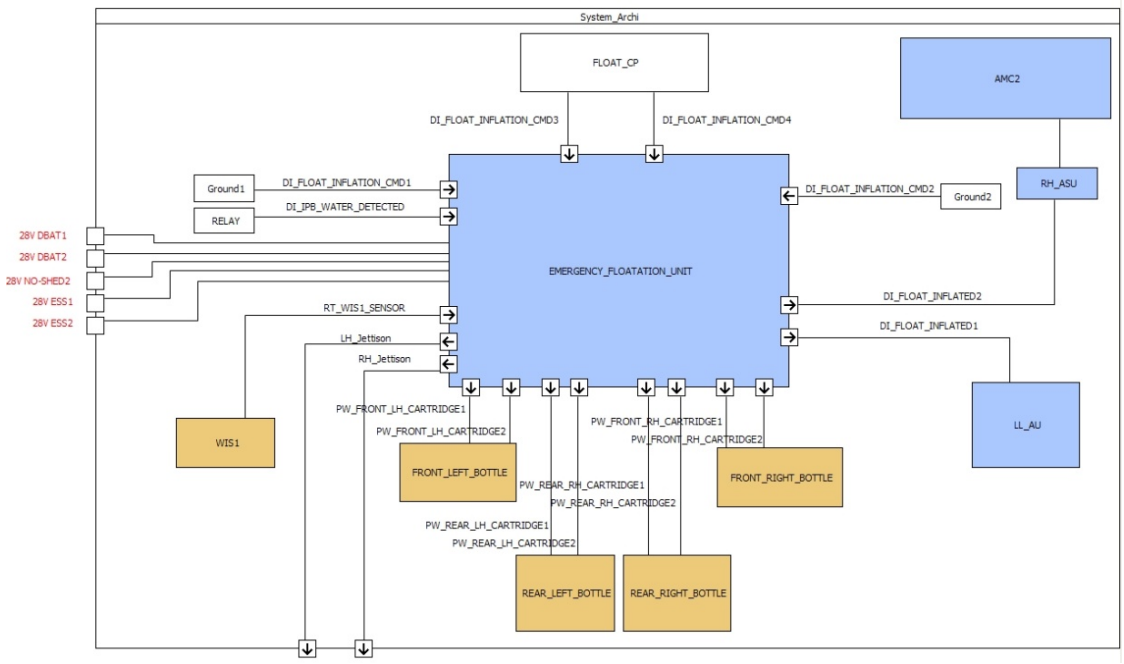

Figure 9 – Preliminary physical architecture within SCADE System

All functional blocks shall be allocated to one or several physical components and all functional data shall be also allocated to physical messages. This allocation can be performed in SCADE System using allocation table as displayed figures Figure 10 and Figure 11.

supplier	client
To Acquire inflation command	FLOAT_CP
To detect helicopter immersion	WIS1
To acquire the information to authorize or not the inflation	EMERGENCY_FLOATATION_UNIT
To acquire the information to authorize or not the inflation	WIS1
To compute conditions to enable disable inflation	FLOAT_CP
To inflate the floats	EMERGENCY_FLOATATION_UNIT

Figure 10 – Allocation of functions to physical components

supplier	client
DI_IPB_WATER_DETECTED	Water_Immersion, Water_Immersion, Immersion_Status
RT_WIS1_SENSOR	Water_Immersion, Water_Immersion, Immersion_Status, Height above water, O
LH_Jettison	
RH_Jettison	
PW_FRONT_LH_CARTRIDGE1	Trigger_Bottle
PW_FRONT_LH_CARTRIDGE2	Trigger_Bottle
PW_REAR_LH_CARTRIDGE1	Trigger_Bottle
PW_REAR_LH_CARTRIDGE2	Trigger_Bottle
PW_FRONT_RH_CARTRIDGE1	Trigger_Bottle
PW_FRONT_RH_CARTRIDGE2	Trigger_Bottle
PW_REAR_RH_CARTRIDGE1	Trigger_Bottle
PW_REAR_RH_CARTRIDGE2	Trigger_Bottle
DI_FLOAT_INFLATED1	
DI_FLOAT_INFLATED2	
DI_FLOAT_INFLATION_CMD1	CMD_Trigger_EFS, CMD_Inflation, CMD_Inflation, CMD_Inflation
DI_FLOAT_INFLATION_CMD2	CMD_Trigger_EFS, CMD_Inflation, CMD_Inflation, CMD_Inflation
DI_FLOAT_INFLATION_CMD3	CMD_Trigger_EFS, CMD_Inflation, CMD_Inflation, CMD_Inflation
DI_FLOAT_INFLATION_CMD4	CMD_Trigger_EFS, CMD_Inflation, CMD_Inflation, CMD_Inflation

Figure 11 – Allocation matrix between functional data and physical messages

Detailed architecture design process

First the Equipments are defined as SysML blocks; a dedicated profile allows architects to provide specific information on the blocks, ports, and connectors representing the inter-equipments connections. This information will define the IRS (Interface Requirement Specification). This can be typically message name, unit, Transmitter and Receivers names, length, type, etc.

SysML Internal Block Diagrams provides the graphical view expected by the architects (Figure 12 – Equipments architecture diagram).

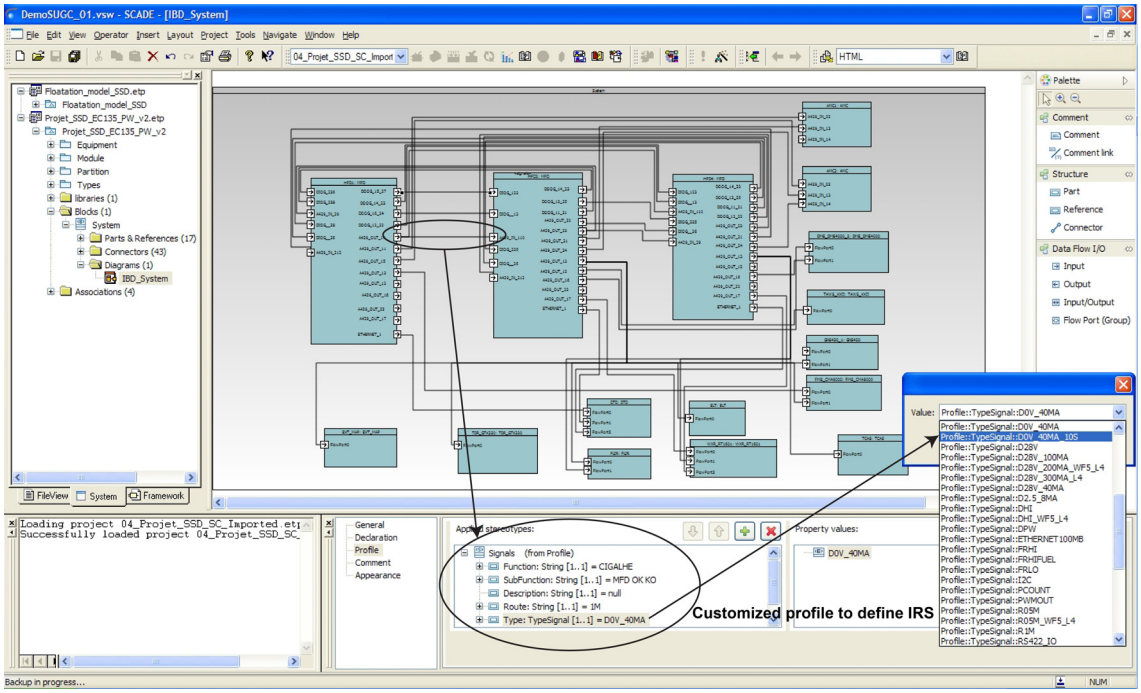

Figure 12 – Equipments architecture diagram

Second, the Equipments are refined with other SysML blocks representing the modules; the same SysML constructions are used (Figure 13). Connections between modules ports (HW board connections) and equipment ports can be modeled using standard SysML connectors, but with specific customized profile.

Third, the Modules themselves are refined into Partitions, again defined with SysML blocks, ports and connectors (Figure 14). At that level, the ports of the Partitions are typed with SW types, some precisely defined, e.g. with enumerated types, some only abstracted, e.g. structured data for which the fields are not defined by the system engineers.

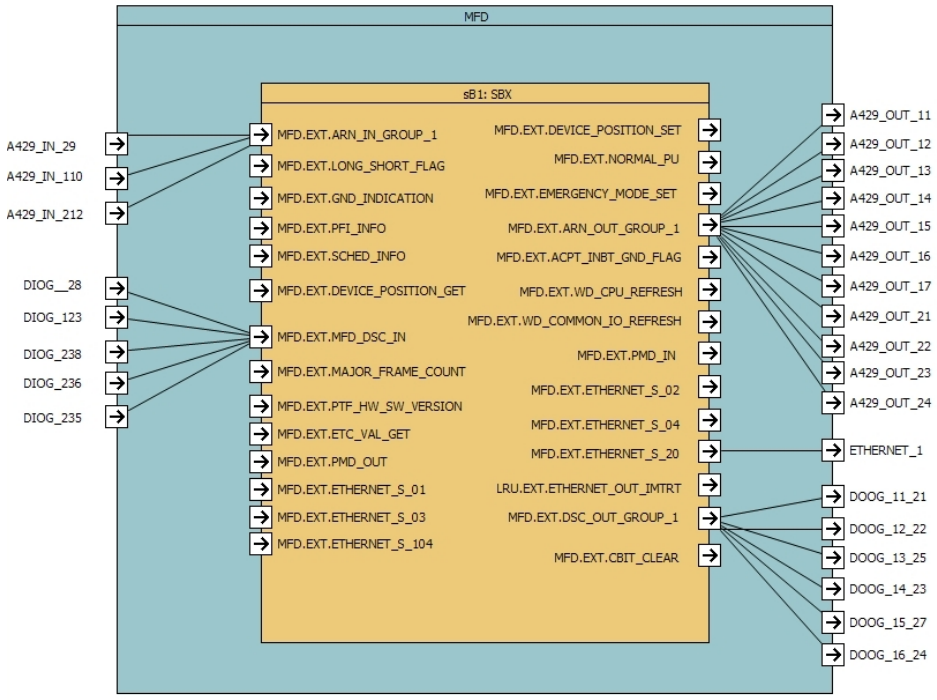

Figure 13 – Modules diagram (within an equipment/platform)

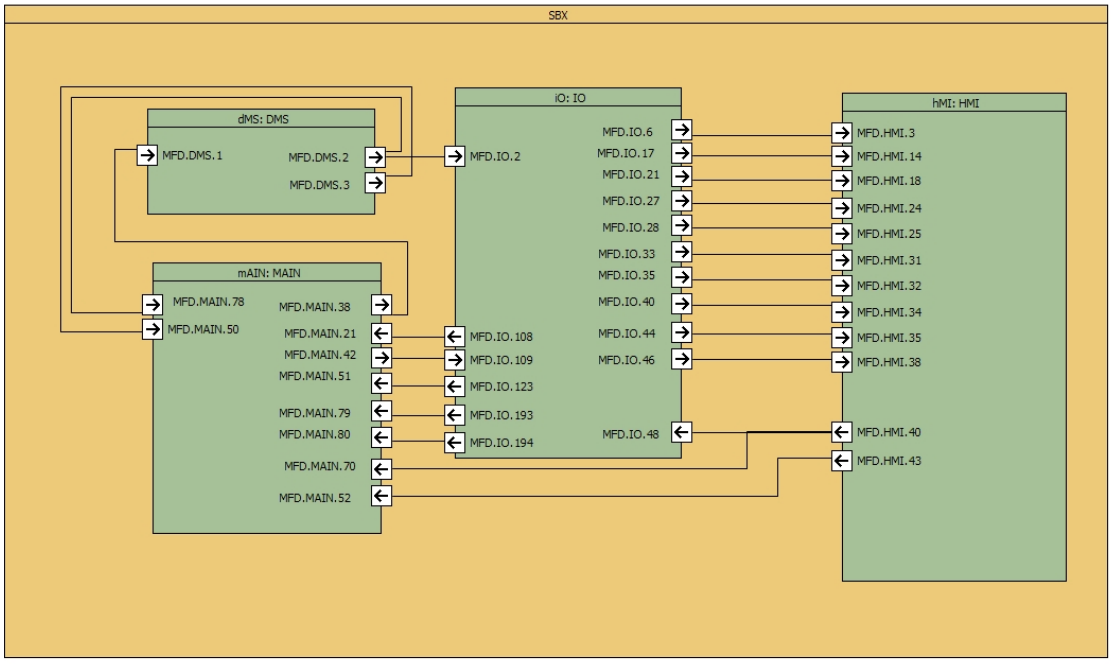

Figure 14 –Partitions diagram (within a module)

Now the Partitions, as a sub-part of the system model, can be exported as standalone SCADE System models to be implemented by SW specialists, thus avoiding sharing the IP of the whole system model. Another means, also supported by SCADE System, is the use of model libraries; if a block is defined as a library model together with all datatype it uses, the library itself can be shared with the SW engineers.

To initialize the SW design of a partition, the synchronization with SCADE Suite is used; a SCADE operator is automatically generated with all inputs and outputs, named and typed according to the System information. Figure 15 shows these operators and datatypes generated automatically in SCADE Suite.

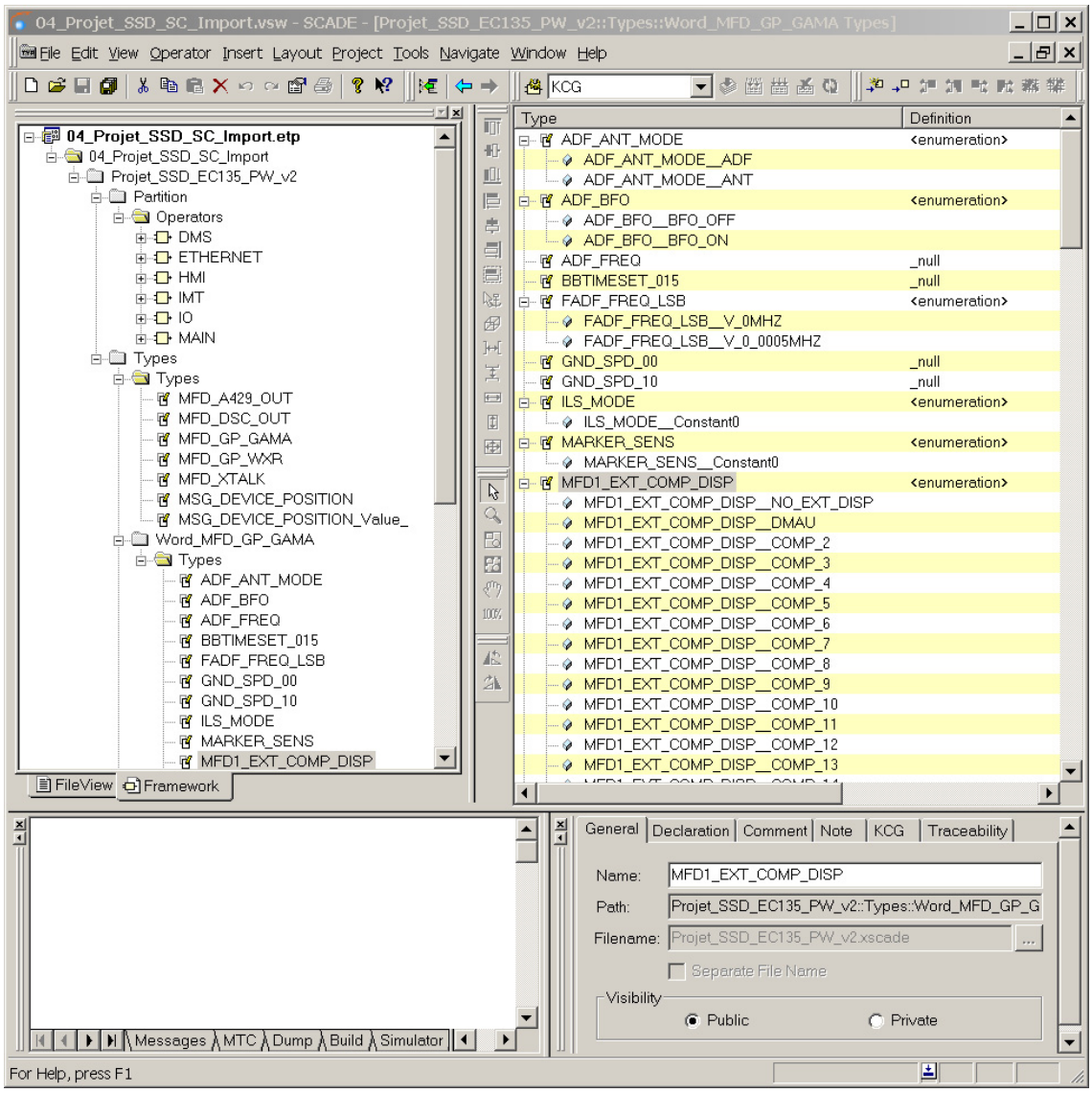

Figure 15 – Synchronized software operators and datatypes

The SW design proceeds with SCADE Suite modeling constructs. In a real project, neither the system model nor the SW design is done at the first shot; both engineering teams are working in parallel in an incremental way. Re-synchronization between the SW interfaces and the system model Partitions can be made either on SW design side or on system model side. Thanks to the traceability information set automatically between the corresponding elements, the re-synchronization algorithm does not affect the pieces that were synchronized previously, thus does not "break" the usage of these elements in their respective models. The semantic diff feature can be used to establish with the other team the contract of the update requested.

When the SW design and verification is completed, the code can be compiled and loaded on the target. ARINC 653 configuration files and "Glue code" must be written to integrate the code generated by SCADE Suite KCG into the actual Modules of the Equipments. This can be generated from the Modules description in the system model with custom scripts using the model API.

CONCLUSION AND FUTURE WORK

This paper has introduced a comprehensive toolset for model-based System and Software engineering. The system architecture design and the Software design can evolve in parallel, interfaces being re-synchronized automatically on request from either side. The technique has been demonstrated on an ARINC 653 system example.

The scope of Model-Based System Engineering is not restricted to the modelization of the system architecture presented in the use case section of this paper. Functional decomposition, analysis and allocation are important aspects that are also supported by SysML. One analysis means of high interest for system engineers is the animation/simulation of the functional model. For that, the functions must be described with precise behaviors that can be computer interpreted or compiled. The Scade language brings both a graphical description and precise, simple, non-ambiguous semantics, making it a good candidate to describe system functions. The AGeSys project, led by Esterel Technologies, has among others, the objective of a deeper integration of SCADE Suite in SCADE System to provide such functional simulation to system engineers.

REFERENCES

1. "Systems Engineering Handbook, a Guide for System Life Cycle Processes and Activities", SE Handbook Working Group, INCOSE, January 2010.
2. "OMG Systems Modeling Language (OMG SysML)", OMG, Version 1.2, June 2010
3. "DO-178B Software Considerations in Airborne Systems and Equipment Certification", RTCA/EUROCAE, 1992
4. "ARP4754 Rev A. Guidelines for Development of Civil Aircraft and Systems", SAE Aerospace, Revised 2010-12
5. Papyrus, http://www.eclipse.org/papyrus
6. Listerel Critical Software Lab, http://www.listerel.org
7. Esterel technologies SCADE products, http://www.esterel-technologies.com
8. SCADE Language Reference Manual, http://www.esterel-technologies.com
9. "The Synchronous Dataflow Programming Language LUSTRE", N. Halbwachs, P. Caspi, P. Raymond, D. Pilaud. Proceeding of the IEEE, September 1991.
10. "A Conservative Extension of Synchronous Dataflow with State Machines", J.L. Colaço, B. Pagano, M. Pouzet. EMSOFT'05
11. "Bridging UML and Safety-Critical Software Development Environments", Alain Le Guennec, Bernard Dion. ERTS 2006
12. "SCADE 6: A Model Based Solution For Safety Critical Software Development", François-Xavier Dormoy. ERTS 2008
13. "Using SCADE System for the Design and Integration of Critical Systems", Thierry Le Sergent, Alain Le Guennec, François Terrier, Sébastien Gérard, Yann Tanguy. SAE Aerotech Congress 2011