

# Integrating System and Software Engineering Activities for Integrated Modular Avionics Applications

Thierry LeSergent  
Esterel Technologies

Frederic Romeas and Olivier TOURILLION  
EADS Eurocopter

## ABSTRACT

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 and incompatible tools. This often leads to a difficult collaboration, with at some point, redundant information and inconsistencies.

This presentation introduces a solution, based on the SysML standard for system modeling, and on the 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 while improving cooperation due to a unified tool framework for System and Software models, the same requirements traceability and documentation generation tools, and 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 like 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.

The system and software engineering flow is demonstrated on an avionics application use case from Eurocopter.

It starts with a description of the intended application functionality. A description of the IMA equipment, modules, and partitions is then provided. This is followed by a description of the application architecture together with a tabular description of the allocation of the system functions to the architecture physical components and a description of the allocation of functional data to physical messages.

Finally, the software blocks are mapped to a SCADE Suite software model. This interface is the basis for synchronization between system and software engineering activities. The synchronization action shall be explicit and it can be triggered either by the system team or software team.

When the software design and verification is completed, the code can be compiled and loaded on the target. ARINC 653 configuration files (IMA) and glue code can be automatically generated from scripts based on the system and software model APIs.

**CITATION:** LeSergent, T., Romeas, F. and TOURILLION, O., "Integrating System and Software Engineering Activities for Integrated Modular Avionics Applications," *SAE Int. J. Aerosp.* 5(1):2012, doi:10.4271/2012-01-2139.

## SYSTEM ENGINEERING TOOLSET

The first requirements for a toolset supporting the System Engineering activities for software predominant subsystems are:

- Functional decomposition of the high level system requirements
- Description of the architecture of the system
- Allocation of the functions to the architecture components

Additional requirements include:

- Traceability with High Level Requirements
- 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 SCAD 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 [2] 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. SCAD 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 SCAD System solution relies on tabular views,

managed as additional diagrams. Note that this does not introduce additional SysML model objects that would make SCAD System incompatible with the standard.

## SCAD System

SCAD 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 SCAD Suite, the toolset detailed hereafter for the support of the software development processes, so the same ergonomics;
- Completely integrated into SCAD Suite, for example using the same project and file management system;
- Features a synchronization mechanism [13, 15] between the system design model authored in SysML with SCAD System and the software design model authored with SCAD 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 1 shows the resulting IDE with SysML Block Definition Diagrams and Internal Block Diagrams in the same framework as SCAD Suite diagrams. The selection of a tab in the tree window (left side) switches between the system and the software objects trees.

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

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

Another important feature provided both by SCAD System and SCAD Suite is a programmatic interface (model API) that allows implementing automated verification activities, or automatic generation of project specific file, e.g. configuration files or wrapping code. 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.

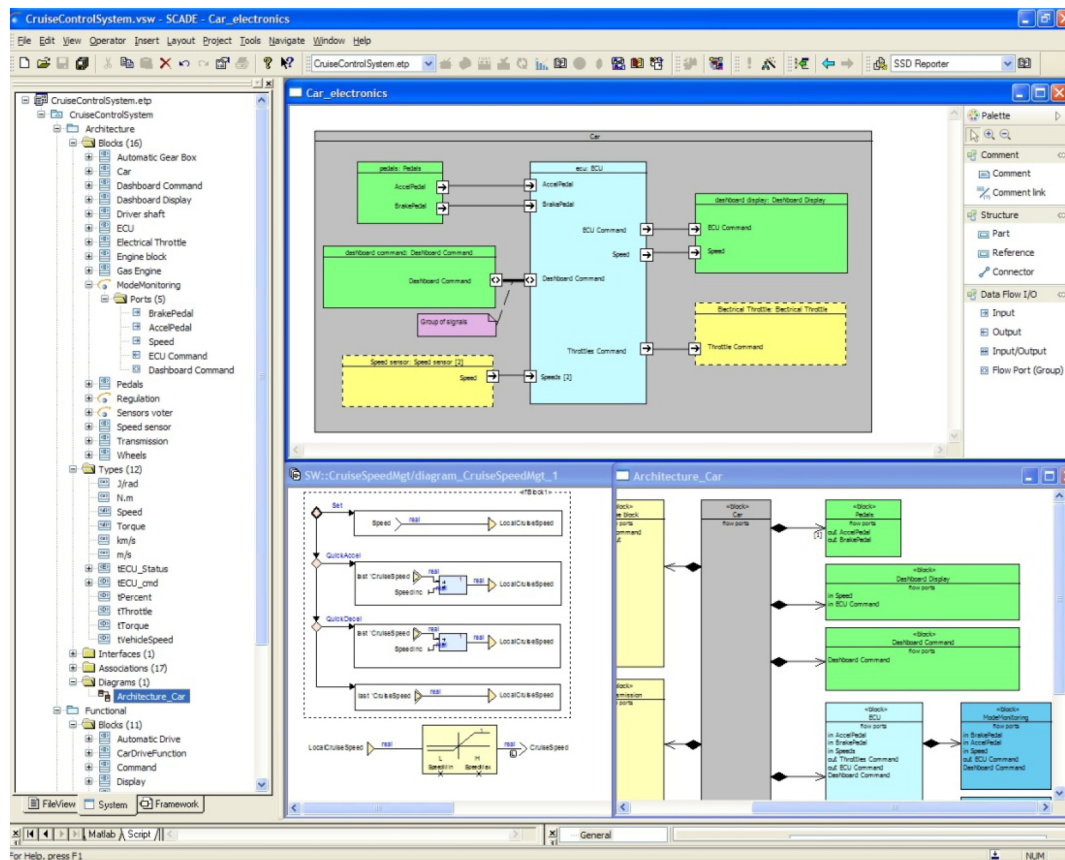


Figure 1. Integrated IDE: SCADE System and SCADE Suite windows

## Data Dictionary

A common technique used for the development of complex software predominant systems relies on a set of “data”, information exchanged between the software components of the system.

These databases are used as detailed specifications exchanged between OEM and equipment manufacturers. Also, a set of in-house tools have been developed over the years for verification activities and code generation, so it is important for a new system engineering tool to be able to import and export data dictionaries saved in these databases.

The inconvenience of the management of these data with databases is that they are technically separated from the model defined in the system engineering tool. Keeping the information coherent, such as data names and topological information (source and target of each data), is a challenge.

The solution proposed by SCADE System is the following:

- Data can be defined as SysML objects, for example values of a block, with a name, a datatype and on which a set of properties can be defined (see below the way these are defined),
- Data are managed in tables allowing easy find/sort/filtering and direct edition of their properties,

- Tables of data can be exported/imported as csv files, allowing inter-exchange with external databases,
- Data can be allocated on SysML connectors by simple Drag&Drop,
- The topological information associated to the data is not stored as data properties but “computed” from the allocations and end-points of the connectors. This information is exported in csv files as properties of the data (on import, only name, datatype and properties values are imported),
- In addition to the topological allocations, data can be allocated on each other.

A typical usage of this mechanism is the following:

- As part of the functional analysis of the considered subsystem, the functional data exchanged between the functions are defined and allocated on the connectors representing the flow dependences of the functions,
- When the architecture of the system is defined and the functions are allocated to the architecture components, one has to define how the function data shall be implemented. This is done by defining, using the same tool feature, logical data. The functional data are allocated on the logical data, which in turn are allocated on the architecture topological elements, connectors or ports.

- In complex architecture where data are packed/unpacked at several architecture levels, the mechanism can be used again, modeling that several logical data can be transported by e.g. a message, itself defined as a data allocated to topological elements,
- The data and allocations done as part of the model are of course traced with High Level Requirements where appropriate, and reported in the documentation generated by SCADE LifeCycle Reporter.

Figure 2 summarizes this methodology, on a simple example. The use case section below shows such modeling methodology on a concrete IMA example.

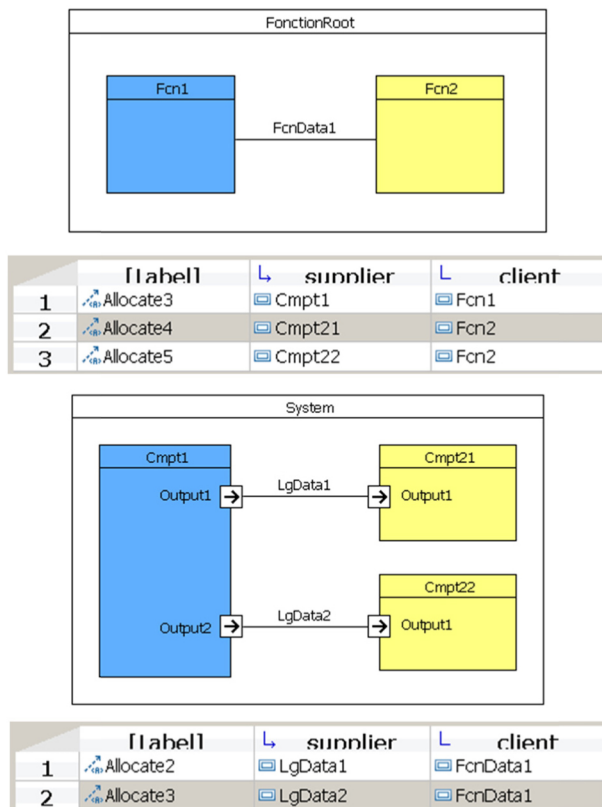


Figure 2. Data dictionaries usage.

Data, functional or logical, shall carry information such as “value range”, “position in a message frame”, etc. Defining this set of properties and for each data the value of such properties must be supported in a friendly way by the engineering tool. The technical foundation of SCADE System, the Eclipse MDT Papyrus project, allows defining such information through the standard profile mechanism. This customization has the advantage of being an effective standard; the storage file for the profiled model follows OMG XMI supported by most SysML/UML COTS tool. But the profiling mechanism is a rather low-level mechanism, not easily managed by system engineers who are experts in their domain but not necessarily in UML technology.

To provide the necessary means to system engineers, SCADE System comes with an annotation mechanism reused

from SCADE Suite [10] that has been proven to be very effective in many complex industrial projects. A simple textual language allows defining structured information types (aka “annotation schema”) associated to any model object, e.g. the SysML block values that implement the data. The fields of these information types can be directly extracted from the IDE and set as columns in the table representing the data dictionaries. Users directly set the data information in these tables, and they can be imported and exported in a straightforward way with external data bases.

The on-going improvement of this mechanism includes saving the annotations in separated files, one per annotation type. This way, each specialist using the same model can set his own information, managing these in versions without modifying the model files themselves. Due to the associations between the model elements and the annotation being made with OIDs, this framework strongly supports model upgrades (when loading his annotations, the user is automatically notified if a model element he has annotated has been deleted).

## 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 [9] 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 [10, 14] 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 [12, 14]. Characteristics are: