Accelerate the Development of Certified Software for Train Control & Monitoring Systems

Franck CORBIER

Dassault Systèmes, 35 rue Haroun Tazieff, 54320 Maxéville, France

Keywords

Transportation, Railway, TCMS - Train Control & Monitoring System, Model based System Engineering, EN50128, SSIL – Software Safety Integrated Level, EN50128, PLCopen, ControlBuild, Automatic Code Generation, Software Certification, Certified Code Generator.

Abstract

The rail transportation industry is highly dynamic and driven by the need to meet with mandatory safety criteria. Modern railway transportation systems are ever more sophisticated and rely heavily on embedded systems and communication networks. Train builders and their suppliers have to develop and deliver these increasingly sophisticated embedded systems while meeting increasingly stringent quality, safety and certification constraints.

Driven by the EN-50128 standard 'Software For Railway Control And Protection Systems' [3], which proposes needed methods to be used in order to provide software for control systems, train builders have applied processes and adopted tools that help to develop and validate software in compliance with safety constraints. This paper will illustrate how model based system engineering approaches and frameworks provide full business process support to:

- Validate the functional requirements of the systems (requirement management);
- Design control and safety functions at the train, vehicle and equipment levels (functional design and software architecture);
- Design and develop the software applications of electronic controllers;
- Integrate, validate and qualify electronic systems in a progressive integration process;
- Manage all changes during the lifecycle of the product (change management but also option and variant management);
- Demonstrate the compliance with the required safety integrated level,
- Train and educate drivers, rail network operators and maintenance technicians.

This kind of development and validation process and the associated tools are now successfully used by many of today's train manufacturers and equipment providers.

Control engineers in railway commonly used development tools providing the IEC61131-3 languages 'Programmable Controllers – Programming Languages' [1] that are deployed in industrial automation for a long time. In this paper, we will give an overview of the languages properties and their integration in the development of software for railway.

Since the latest version of EN-50128 introduced in 2011, train control system providers are facing increased constraints on the development processes but also on the verification and validation activities. The certification of a railway control system demands ever more documentations and demonstrations that the software complies with the safety rules and criteria. All these additional tasks increase the development and validation load but are nevertheless legitimate since they are designed to ensure the safety of the product. As the railway operators ask for SIL2 product, this paper will present a new technologies helping railway OEMs and suppliers to increase the Safety Integrated Level of control software from SSIL0 to SSIL2 and to master the cost of the product.

State of The Art

In railway industry, the functions (i.e. doors, traction, brake, lighting, fire detection, air conditioning) done with conventional control system (hardware based contactors, relays, circuit breakers, etc.) are increasingly being managed by intelligent control units. Those controllers are coordinated or synchronized by a centralized and networked control system named TCMS (Train Control and Monitoring System). Depending of the safety level of the functions, the software is developed using various tools:

- softPLC providing IEC61131-3 languages
- Model-Based System Engineering frameworks (providing previous languages or data flow, state machine, etc.) and dedicated code generators;
- Hand coded in C or ADA for legacy reasons.

Some of these tools are coming from automation and control engineering (mainly for SSIL0-2 software applications) and from the avionic industry (EN50128-SSIL4 is roughly the same asDO178C-DALA) even if a product qualified for a standard is not automatically qualified for another standard).

With the growth of embedded controllers and networks in the rolling stock, train manufacturers and their sub-system suppliers need development methods and tools that are able to support the design and help stakeholder to validate requirements and functional specifications as early as possible in the project. In fact, major issues in developing these systems is the development process itself (focusing on the delivery of safety documentation) and the decreasing time and budget available for the test activities including physical tests. Like in other industries, the railway control engineers are looking for virtual test, progressive integration and qualification of the electronic control units and electro mechanical equipment.

In parallel, rolling stock and sub-system providers need to assess the overall safety of the systems, for both passenger and equipment, and to demonstrate compliance with the relevant railway rules & standards (EN-50126, EN-50128 and EN-50129 for hardware, software and product). Traditionally, the design process for embedded electronic boards is built upon a number of elementary practices and follows a V Cycle as show in *figure 1*. The EN50128 recommend the implementation of this kind of development process including verifications and validation tests.

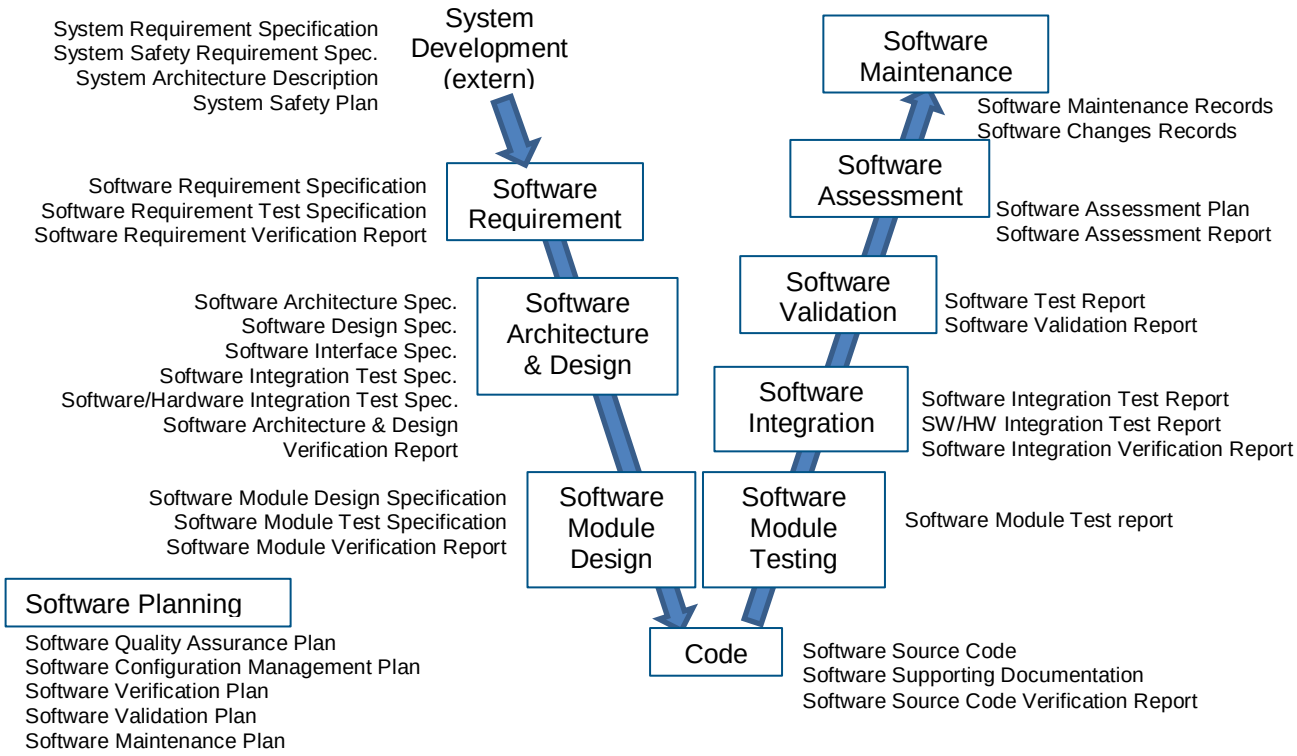


***Figure 1**: V-Cycle development process with documentation delivery*

Moving from a paper based development cycle…

Functional and design specifications are exclusively supported by paper based communications. The project goes slowly because the verification of textual documents requires many review iterations and exchanges between business units, central office, sub-contractors and customers. Too much time is spent in "meeting" clarification and explanation making the sentences of the specifications clear, complete and understandable by all stakeholders involved in the project. Today, too much effort is spent on "paperwork" rather than on systems design, integration and physical tests. The target is not to reduce the number and the content of paper (mandatory) but to ease this production.

The deployment of a V-Cycle methodology also introduces problem. As we can see in *figure 1*, the left side of the V-Cycle supports creation, definition, and refinement activities while the right side covers the testing activities. It means that if coding errors can be found immediately during the programming and software module testing phases, the specification errors can be detected only at the late phases of the V-Cycle during integration and validation phases. Consequently, projects frequently miss their delivery deadlines and are difficult to keep within budget. This legacy process raises important managerial issues that need to be addressed.

… to a Model Based System Engineering Approach

The first proposal was to move from a paper based approach to a model based system engineering approach that provides executable specifications in order to speed up the development of the control software and meet project deadlines. This process was first launched with Electronic Control Units (ECUs) for Train Control & Monitoring System projects and is today deployed on other sub systems like brake, doors or passenger information systems. For this purpose, Dassault Systèmes provides a development framework that enables the achievement of new productivity related objectives.

Model a concept, a control function, a software module

This development framework, named ControlBuild, provides a MIL (Model In the Loop) approach allowing railway engineers to specify and design a model of each control function prior to any implementation. This approach is made possible through the use of known, open and standardized languages.

In the railway industry, control engineers develop the embedded software using the languages defined within the IEC61131-3 standard for many years (see *Figure 2*). Both control engineers (including software and electrical engineers) and maintenance engineers (end users) daily use Sequential Function Chart, Structured Text, Ladder and Function Bloc Diagram for software development and modifications i.e. applying system corrections and improvements. Currently, dozens of PLC and CPU manufacturers provide coding tools based on the IEC61131-3 standard, taking the main place in the coding activities.

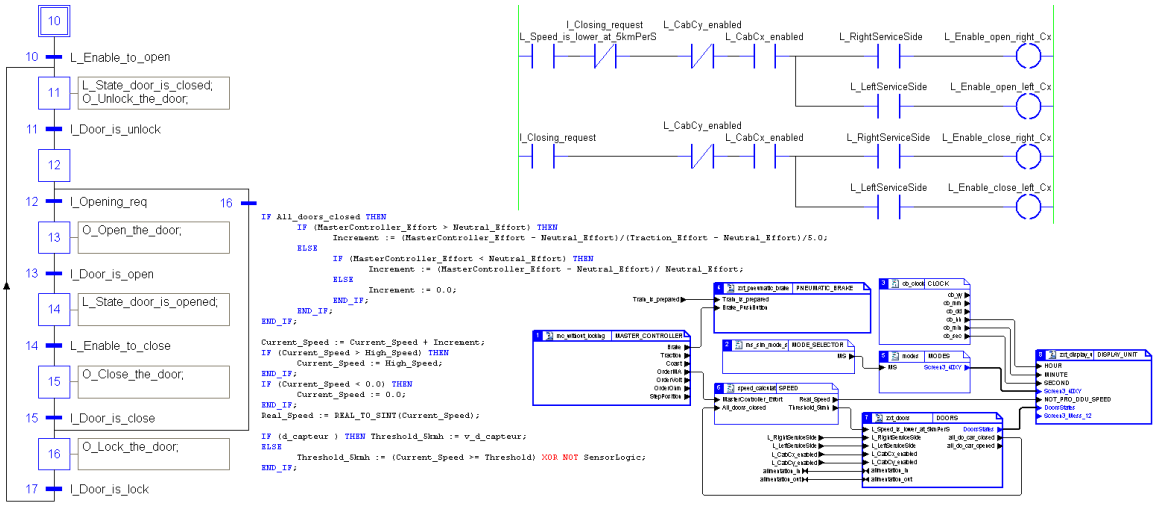


***Figure 2**: Simple models using IEC61131-3 languages*

Therefore, Control Engineers naturally improve their efficiency focusing more on design activity than on the coding phase thanks to these languages. The only requisites were that the tool has to provide:

- Code generators to ease the production of the source code of the software (in compliance with the required coding rules and properties);
- Tools and advanced features to verify, to test and to assess the models (static metrics, quality level of the model, assertions, automatic tests, code coverage, dead code identification, traceability with the requirements, etc.) as presented in the *figure 3.*

The functional design can be manually validated using scope, button, virtual panels and synoptic. Tests of the specification and design models can also be automatically executed using test procedures and functional tests features. This incremental validation approach enables time to be saved on software design and integration testing activities. The detection and the correction of an error during the system validation phases is close to 100 times more expensive than when the tests are made on the model during the design activity.

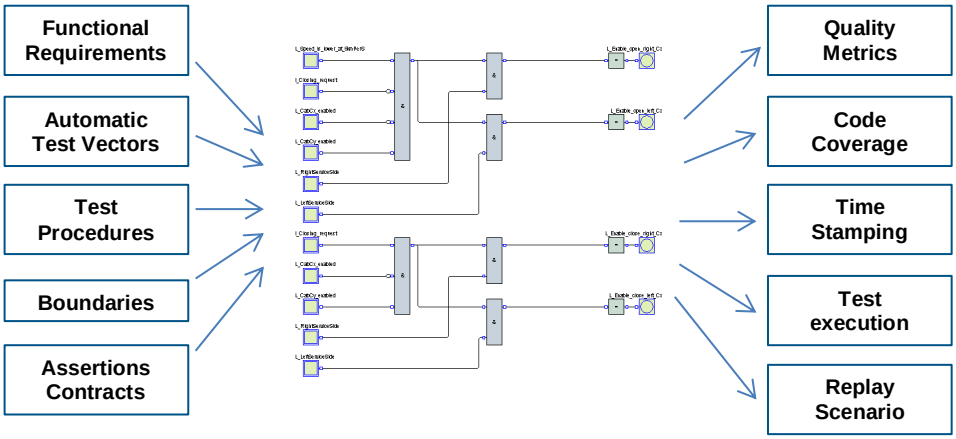


***Figure 3**: Test functionalities and features*

Map the functional model on the hardware architecture

The model of the system is independent of the hardware (manufacturer, brand, operating system, etc.) and the architecture (i.e. networks and protocols). It means that we have designed the model of the software which can be deployed on one controller or distributed on many controllers. At this step of the design phase, we have to map parts (called POU - Program Organisation Unit) of the model on a defined hardware topology of the train system or sub-system. The use of the FBD (Function Block Diagram) language helps the software architect to allocate the POU on the virtual controllers (refer to *figure 4*).

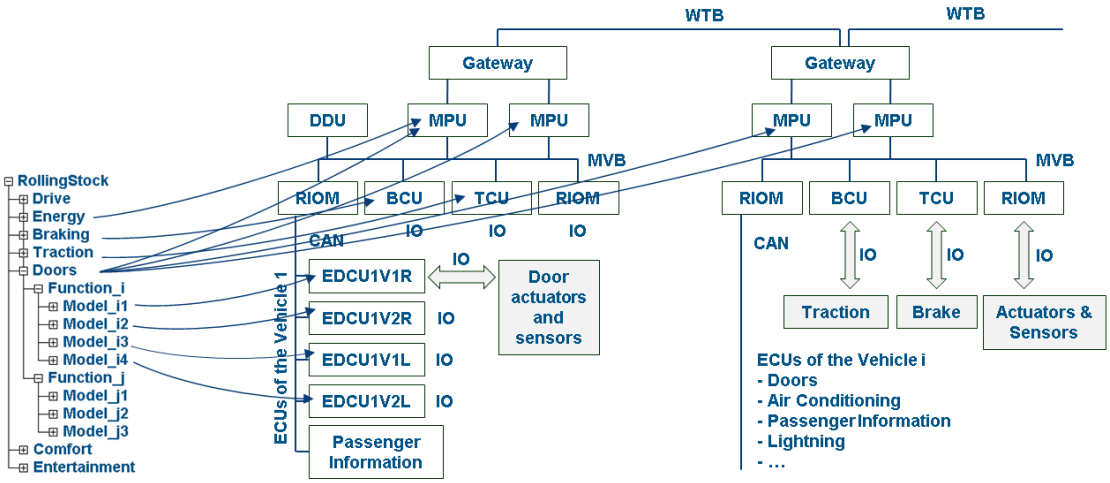


Figure 4: System Mapping – Models to Controllers

Certification of a Software Application

For railway control and protection systems, a certificate ensures that a product is compliant with the safety standards. To get a certificate, the product must be certifiable. It means that the product has been developed in such a way that it can be certified. The EN 50126 standard addresses system issues on the widest scale [2] and the EN 50129 standards addresses the approval process for individual systems [4]. Railway products embed software applications that have to be certified regarding the EN50128 standard. As in other industries, this railway standard define the notion of safety level to be achieved (SIL – generic standard, DAL for avionic, ASIL for automotive and SSIL for Railway) [5]. In avionic, the safety level determines the goal to achieve while in railway it is rather a means to implement.

SSIL – Software Safety Integrated Level

Even if the standard define 5 SSIL from 0 to 4 (4 = high level of criticality), the railway industry commonly proposes to use only 3 safety levels (technics to achieve the level 1 are closer to level 2, and level 3 is also not so far from level 4):

- SSIL0: comfort, passenger information, energy …
- SSIL2: traction/brake, doors
- SSIL4: automatic train control

To get the certification that a software application is compliant with a given SSIL, the software development team has to demonstrate to the evaluation engineers / certification authorities that:

- A well-defined software development process is used (quality assurance like ISO9001) including verification and validation tests,
- The software application covers every and only the customer requirements (using traceability requirement tools like Reqtify),
- The software application complies with the safety coding rules (like MISRA rules - Motor Industry Software Reliability Association),
- The software application is covered by test procedures,
- All associated document specification and test reports are available.

Be careful, SSIL0 doesn't mean 'no safety'; it is just the first level of safety and requires the same quality assurance process than for other SSIL. The safety case is just lighter.

Challenges and issues

Train operators are demanding even more SIL2 products for their trams, metros, inter-cities and high speed trains. It means that the Software Safety Integrated Level of the software applications of these new products have also to be compliant with SSIL2. In parallel, the latest version of the EN-50128 standard introduces more stringent activities and constraints on the development of software systems. Event for SSIL0 software application, number of companies improved their development process replacing manual programming (mainly C code) by model based design and automatic code generators.

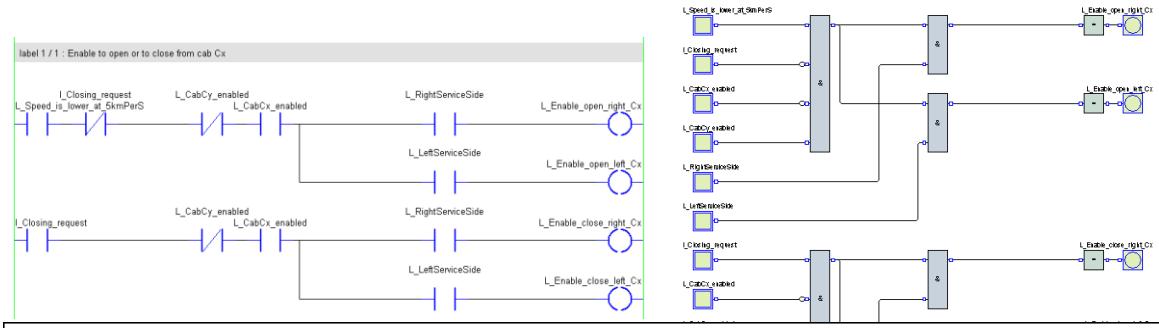
Development tools supporting the IEC61131-3 standard are the most used in the railway community of control engineers for SSIL0-2 software applications. A well-known tool used in the avionic industry, is rather used for the development of SSIL4 software for signalling - automatic train control systems – that concern less than 5% of the embedded software in a train).

An IEC61131-3 tool usually provides two kinds of code generators:

- Generation of a sequence of instructions to be executed by an interpreter. The certification is difficult to achieve because the execution machine / runtime must be certified by the tool provider. Some softPLC providers got a SIL certificate (linked to the IEC61508 generic standard) covering a limited set of instructions (AND, OR, NOT, etc.) and reserved for the execution of Boolean expressions;
- Generation of a source code to be compiled and executed on a real time platform.

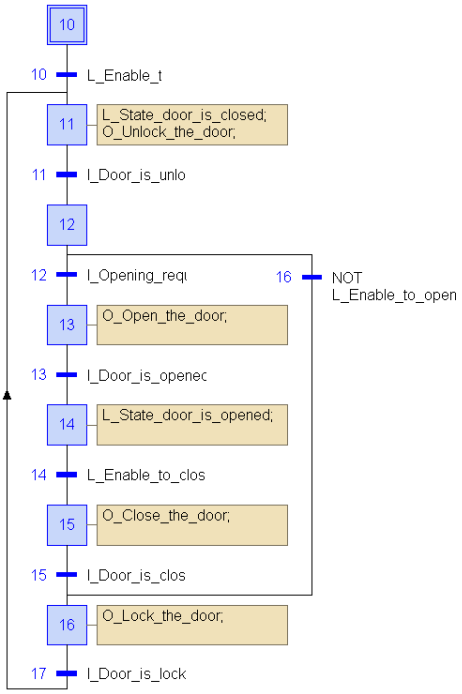
To reduce the dependency of the source code to the compiler, the EN50128 standard propose to define a subset of the language allowing a controllable execution (no jump, limited use of pointer, no dynamic memory access, no if without else, a default for switch case, only one return inside a function…).

There are many issues automatically solved because the IEC61131-3 languages are already limited (no pointer in structured text, no direct access to the operating system, etc.). Limited doesn't mean simple but sufficient to provide the code for the controller of the traction/brake systems or for the other converters, doors, air conditioning, pantograph, etc.




```
enableOpen = ((L_Speed_is_lower_at_5kmPerS && !I_Closing_request) && !L_CabCy_enabled) && L_CabCx_enabled;
L_Enable_open_right_Cx = enableOpen && L_RightServiceSide;
L_Enable_open_left_Cx = enableOpen && L_LeftServiceSide;
enableClose = (I_Closing_request && !L_CabCy_enabled) && L_CabCx_enabled;
L_Enable_close_right_Cx = enableClose && L_RightServiceSide;
L_Enable_close_left_Cx = enableClose && L_LeftServiceSide;
```

***Figure 5**: Simple example of C code generation from LD or FBD model*




```
/*pre-SFC*/
/*transitions G1*/
_Y_8 = _X_10 && L_Enable_to_open;
_Y_14 = _X_11 && I_Door_is_unlocked;
_Y_13 = _X_12 && I_Opening_request;
_Y_12 = _X_13 && I_Door_is_opened;
_Y_11 = _X_14 && L_Enable_to_close;
_Y_10 = _X_15 && I_Door_is_closed;
_Y_9 = _X_16 && (I_Door_is_locked && L_Enable_to_open);
_Y_15 = _X_12 && !L_Enable_to_open;
/*evolutions G1*/
if (_Y_8) {
    _X_10 = 0;
}
...
if (_Y_8) {
    _X_11 = 1;
}
/*timers G1*/
/*actions G1*/
if (CB_edge_bool(_X_11, _X_11_PREV) < 0) {
    L_State_door_is_closed = 0;
    O_Unlock_the_door = 0;
}
if (CB_edge_bool(_X_13, _X_13_PREV) < 0) {
    O_Open_the_door = 0;
}
if (CB_edge_bool(_X_14, _X_14_PREV) < 0) {
    L_State_door_is_opened = 0;
}
if (CB_edge_bool(_X_15, _X_15_PREV) < 0) {
    O_Close_the_door = 0;
}
if (CB_edge_bool(_X_16, _X_16_PREV) < 0) {
    O_Lock_the_door = 0;
}
```

***Figure 6**: Simple example of C code generation from SFC*

More than that, the modelling tools provide features and metrics allowing the developer to measure the complexity of the structure of the models (depth of the hierarchy, number of interfaces, number of lines, number of blocks, and number of sub-loops for example). As the IEC61131-3 standard includes textual language (ST: Structured Text), the model described in ST must also be compliant with rules such as those defined by the MISRA group.

However, at the end of the development process, the generated software application has to be certified like a manually programmed certifiable software application.

Certified SSIL2 Software generated by a Certified Code Generator

The aim of a software code generator is to help developers to remove the coding activity and to ease some verification and safety demonstration. It is why most of the development tools provide an automatic translator that takes into account a systems model and delivers source code for implementation on electronic control units.

At this step, the software engineers have to provide deliverables and demonstrations for the evaluation of the static and dynamic criteria defined in the safety standard:

- First, from a quality point of view, the generated software has to be readable, testable, verifiable and, maintainable. This means that coding rules have to be assessed as if the software were written by hand (depth of the function calls, number of overlapping 'IF' statements, number of inputs, outputs & parameters, the justification of use of pointer or global variables, dead code, out of boundary protection, defensive code, etc.).
- Another activity is to demonstrate that the proposed test cases totally cover the test requirement specification document, and then the customer requirements by using requirement traceability tools.
- Then, the software engineers have to demonstrate that the generated software has the same behaviour as the IEC61131-3 models if validation activities were already done at the design level. If not, all unit tests have to be executed and reported as if the software code were manually written.

The aim of a certified code generator is to insure that the generated code satisfy a certain level of confidence regarding the accuracy of the outputs relative to inputs. A certified code generator helps developers to remove unit tests and number of manual demonstrations.

Prerequisite: use of a well-defined input format

Since the first release of the IEC61131-3 programming standard, users want to be able to exchange their programs, libraries and projects between development environments. in fact, there are more than one hundred tools providing the IEC61131-3 textual and graphical languages but saving the description using their own specific format. Although this was not the intent of the standard itself, it was a task that the independent organization PLCopen [6] committed itself to. IEC61131-3 is focused on the software development environment.

This resulted in a workgroup named TC6 for XML. This committee defined an open interface between all different kinds of IEC61131-3 software tools. It provides the ability to transfer the information that is graphically presented on the windows of one coding tool to other coding tool. This format supports textual description (i.e. for structured text, SFC activity and receptivity) but also graphical information, like position and size of the blocks, connections (including route) between the objects of the language.

The SSIL2 Certified Code Generator

With aims of sustainability, we decided that our SSIL2 Certified Code Generator would take XML files defined by the PLCopen organization as the input and provide SSIL2 C code as the output. As in avionic, it is mandatory that the SIL level of the code generator has the same SIL level as the code it produce.

The *figure 6* shows that the SSIL2 Certified Code Generator is composed of 4 modules (light grey):

- The PLCopen XML comparator: this module verifies that the XML input file is compliant with the version of the tool.
- The IEC61131-3 library: the C functions associated to each IEC61131-3 function block have been certified by static analysis and unit tests.
- The XML to C code generator.
- The report generator of the code generation execution

These 4 certifiable software modules have been developed following a SSIL2 development, verification and validation process. The evaluation process has been executed by the certification authority named CERTIFER [7] which deliver the 8270/0157 certificate: *SILCoder version 3.00 meets the 551L2 requirements of the standard EN50128:2011*.

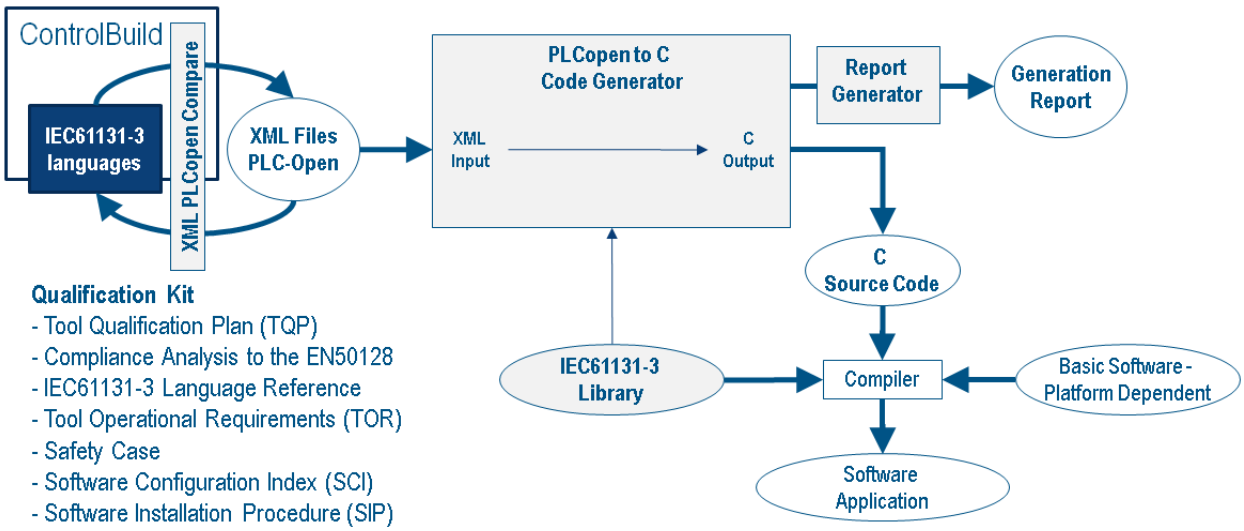


***Figure 6**: Certified Code Generator*

The certificate is associated to an annex which gives the list of the definition documents (requirements, plans, specifications, design, tests and reports) with the related version used for the evaluation, the results of the evaluation and also the perimeter of the certification.

Some tangibles and intangibles results

Contribution of Model based design and model based testing methodologies

A model-based development process must be used closely to the requirement management process. Modelling frameworks allows stakeholders to simulate the requirements, the functional specification, the design specification and then the software and the hardware (engineers from the automotive fully play with Model In the Loop, Software In the Loop, Hardware In the Loop and today Driver or Human In the Loop). The customer requirements and needs are immediately understood by the system provider. Functional specification can be dynamically explained to the customer, same for design, software and hardware. For example, OEM provides us return of investment showing that only 3 reviews compare to 10 before are necessary to get the design acceptance from the customer.

Executable specifications allow easy verification and validation with both internal and external non experts groups. End users like drivers or maintenance engineers are now involved in the specification reviews and design reviews rather than at the end of the project. Verification and validation at every stage of the V-Cycle significantly decreases system failure risks by finding errors at the earliest.

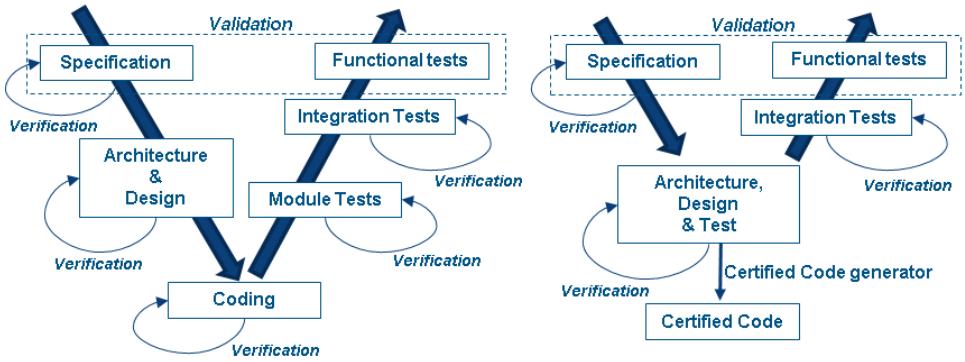


***Figure 7**: Improvement of the V-Cycle using Model based Design and Certified Code Generator*

Value of a Certified Code Generator

The SSIL2 Certified Code Generator delivers SSIL2 Certified C Code. As the software application is not just certifiable but automatically certified, software engineers can remove some demonstrations activities from their development and verification processes. The certificate gives the commitment and ensures that the software application is compliant with the SSIL2 readability and testability criteria and the recommended programming quality rules.

The software to embed in a real time controller is composed by two parts (see *figure 6*): the software application that is specific to one project and the "basic software" which makes the link between the software application and the operating system or directly the hardware if the target has not an OS.

As the specific application part is automatically certified using the Certified SSIL Code Generator, the product provider has only to certify the software dependent part of the generic target using his corporate SSIL2 Development and Verification & Validation process. In fact the certification of the basic software has to be done one time independently of the projects by the team managing the execution platform as a product: Each train maker defines and certifies a platform (hardware + basic software) that will be used during 5-10 years for multiple railway projects).

The final product is based on certified **generic** basic software (from OEM) and a certified **specific** software application (from our SILCoder). Certificates of the platform and of the code generator replace lot of tests and demonstration and save time and money. The Control System provider can focus directly on the software integration activities and software/hardware integration activities.

What we test is what we embed

How to demonstrate that the behaviour of the model has the same behaviour as the software? This big issue is solved using our IEC61131-3 development framework (ControlBuild) and our certified code generator (SILCoder). Each IEC61131-3 model is saved using the PLCopen format (native format for ControlBuild) and a unique C code is automatically generated from the XML PLCopen description. The C code of each model is generated one time and is linked with the right target platforms for simulation, code generation, debug, and deployment or test bench objectives as we can show in the *figure 8*.

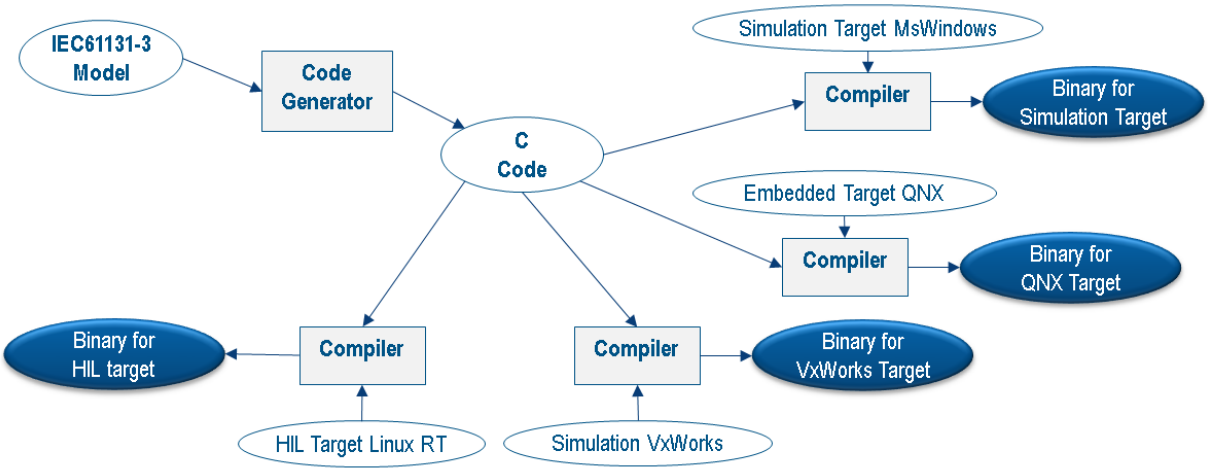


Figure 8: *What is simulated is what is embedded*

It means that we reuse the same generated C code (from the models) to provide software applications but also real time simulators (actuator, sensors and virtual controllers) for the HIL platforms (Hardware-in-the-Loop, Iron Bird in avionic). The test engineers focus on the virtual world making number of verifications. Like with Virtual Iron Bird, it is easier to execute test procedures and insert failures on the virtual environment (actuators/sensor) of the virtual product under test. Objective of virtual bench is to find and remove the last errors while the validation on HIL is to make the final demonstrations and to provide the deliverables for certification authorities.

Value for the railway industry

The IEC61131-3 languages, mainly used by the software engineers, will continue to take an ever more important place in development and delivery of software applications for railway. As the Certified Code Generator delivers the right level of safety for the control software, the gap to change the level of the certification of the software applications from SSIL0 to SSIL2 is significantly reduced. But for sure, it also means that the system provider must still upgrade his development and verification & validation process and integrate the use of modelling and code generation tools.

As in avionic with tools like SCADE®, the value of such a model based design and testing approach using ControlBuild coupled with a Certified Code Generator is promising in terms of productivity benefits and organizational effectiveness for the train makers. For the whole process, the time required to design and implement SIL2 Certified Train Control & Monitoring Systems is now halved with the ability to capitalize and reuse systems knowledge and certified assets on others projects through the standardization of train functions (and their certified products and software).

With the other IEC61131-3 tools providers, we are engaged in the official standardization of the XML PLCopen format. The name of this future standard will be IEC61131-10. This will open the use of the Certified SSIL2 Code Generator for all the developer's community. More and more products will be compliant with the SIL2 level for the safety of passengers, operators and systems.

Terminology

ASIL: Automotive Safety Integrated Level
BCU: Brake Control unit
CAN: Controller Area Network
DAL: Design Assurance Level
DCS: Digital Control Systems
DDU: Display Driver Unit
ECU: Electronic Control Unit
EDCU: Electronic Door Control Unit
EE: Embedded Electronic
FBD: Function Block Diagram
HIL: Hardware In the Loop
ICD: Interface Control Documentation
IP: Intellectual Property
LD: Ladder Diagram
MIL: Model In the Loop
MVB: Multifunction Vehicle Bus
OEM: Original Equipment Manufacturer
OS: Operating System
PLC: Programmable Logical Controllers
PLM: Product Lifecycle Management
POU: Program organization Unit
SFC: Sequential Function Chart
SIL: Safety Integrated Level
SSIL: Software Safety Integrated Level
ST: Structured Text
SUT: System Under Test
TCMS: Train Control & Monitoring System
TCU: Traction Control Unit
WTB: Wire Train Bus
XML: eXtended Mark-up Language

References

[1] IEC61131-3, Programmable Controllers – Programming Languages, 2001

[2] EN50126, Railway applications - The specification and demonstration of Reliability, Availability, Maintainability and Safety (RAMS), 2000

[3] EN50128-2011, Railway applications - Communication, signalling and processing systems - Software for railway control and protection systems, 2011

[4] EN50129, Railway applications - Communication, signalling and processing systems – Safety related electronic systems for signalling, 2003

[5] Jean-Louis Boulanger, Formal Methods – Industrial Use from Model to Code, ISTE-WILEY, 2012

[6] PLCopen for efficiency in automation, http://www.plcopen.org

[7] CERTIFER, Certificat de Type N°8270/0157 Edition 1, 2014