A methodology for automatic generation, formal verification and implementation of safe PLC programs for power supply equipment of the electric lines of railway control systems

M. Niang, B. Riera, A. Philippot, J. Zaytoon, F. Gellot, R. Coupat

A methodology for automatic generation, formal verification and implementation of safe PLC programs for Power Supply Equipment of the Electric Lines of Railway Control Systems

M. Niang(**), B. Riera(*), A. Philippot(*), J. Zaytoon(*), F. Gellot(*), R. Coupat (**)

(*)Université de Reims Champagne Ardenne, CReSTIC EA 3804, Reims, France

(**)IP.TE (CES), Direction de l'ingénierie, SNCF, La plaine Saint Denis - France

A methodology for automatic generation, formal verification and implementation of safe PLC programs for Power Supply Equipment of the Electric Lines of Railway Control Systems

Abstract:

~~Industry 4.0 requires proposing advanced methodologies and tools adapted to control engineers in order to improve safety, flexibility and to save time during automation projects.~~

To improve the design, Verification and Validation phases of Power Supply Equipment of the Electric Lines control systems at French Railway Company (SNCF), this paper proposes an integrated methodology, for automatic generation, formal verification and implementation of safe Programmable Logic Control (PLC) programs. The main objective is to save time and to improve the "overloaded" workflow of systems engineers.

This methodology is compliant with the traditional engineering workflow. The first phase of the methodology focuses on the automatic generation of PLC programs, wiring diagrams, and test-based recipe books, based on reusing and adapting similar models of existing projects to the new specifications (corresponding to functional and safety requirements). The second phase is related to the application of formal verification and control synthesis techniques to guarantee the safety of the control installation. The first phase of the methodology has been successfully deployed at SNCF. The second phase is currently being evaluated.

Keywords: Railway engineering, Standardization, Control system, safety, formal verification, validation, Virtual Commissioning

1. INTRODUCTION

The SNCF (French Railway Company) is in charge of the management, exploitation and operation of railway infrastructure in France. This infrastructure totals 30 000 km railway lines with 560 traction substations containing Power Supply Equipment of the Electric Lines (PSEEL). The company is composed of two entities: "SNCF Réseau" (formerly RFF, French Rail Network) which is in charge of management, operations and development of railway infrastructure, and "SNCF Mobilité" which manages transportation of travelers and merchandises.

The PSEELs are the electrical supply points of the electrified lines, called catenary. Their role is to transform, to supply, and to rectify in the case of DC supply, the electrical energy at high voltage into lower voltages (1500V DC or 25kV AC) compatible with traction units (trains). These electrical systems are subject to strict railway safety standards (EN 50126, 2012; IEC 60870-4, 1993). A PSEEL is broadly composed of three

electric subsystems:

- A High Voltage part (HV) connected directly to the electricity transport and distribution networks.
- A Transformer Group (GT) which transforms the received High Voltage energy into a lower voltage adapted to traction units. It is mainly composed of transformer(s), switches, circuit breakers, and a rectifier (in case of DC supply).
- A Track Feeder (FT) which distributes the converted energy to traction units through catenaries. A Track Feeder is composed of switches and circuit breakers.

For example, the PSEEL presented in Figure 1 contains five electric subsystems: one High Voltage part, two Transformer Groups, and two connected Track Feeders supplying three catenaries. A PSEEL is a distributed system that is remotely controlled. The role of the control system is to manage the PSEEL in such a way to supply traction units with electricity while guaranteeing the safety of the installation. It is composed of electric cabinets, each of which controlling a specific electric subsystem and including mainly:

- one or more PLCs (Programmable Logic Controller),
- a terminal block through which the inputs/outputs of the PSEEL are connected to the PLC(s),
- several digital protective relays for monitoring and protecting the PSEEL against electrical accidents.

A control system communicates also with a centralized supervision system which supervises all the PSEEL located in a given area.

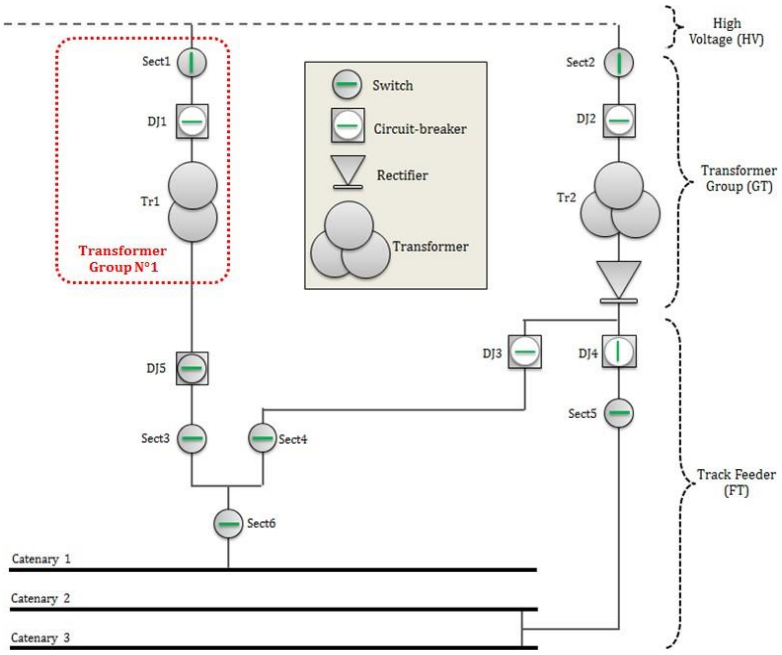
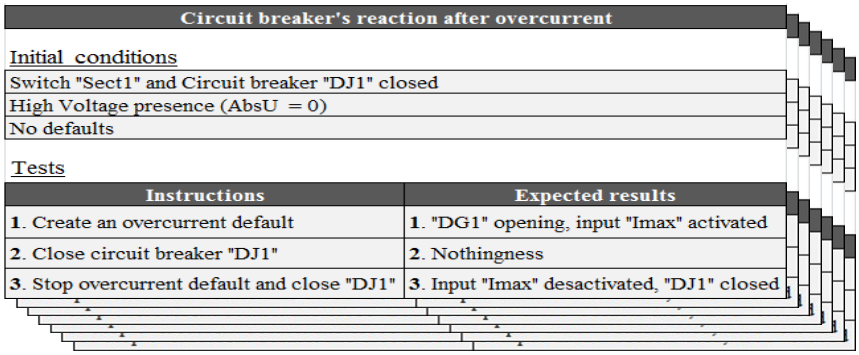


Figure 1. Architecture of PSEEL

At SNCF, design and V&V (verification and validation) phases of a PSEEL have been traditionally performed by systems engineers who are subject to a very tight schedule during an automation project. The design phase of the control system consists traditionally in manual design of models/programs which are commonly called deliverables in the domain of railway automation. These deliverables are the electric

diagrams of control system's cabinets, the PLC programs, and recipe books for V & V. The recipe book is a document comprising tens or hundreds of test cases (Figure 2) which are defined, each, by the initial conditions required to perform the test and a set of sequential test instructions to be executed by an operator with a set of expected results for each instruction.



Circuit breaker's reaction after overcurrent	
Initial conditions	
Switch "Sect1" and Circuit breaker "DJ1" closed	
High Voltage presence (AbsU = 0)	
No defaults	
Tests	
Instructions	**Expected results**
1. Create an overcurrent default	**1**. "DG1" opening, input "Imax" activated
2. Close circuit breaker "DJ1"	**2**. Nothingness
3. Stop overcurrent default and close "DJ1"	**3**. Input "Imax" desactivated, "DJ1" closed

Figure 2. An example of a test case of a recipe book

To avoid starting from scratch, the engineers reuse and adapt similar deliverables of existing projects to the new specifications. Then after proofreading of the new deliverables, the V&V phases are conducted in factory before real commissioning. During this V & V phase, the engineers verify (through manual testing) the conformity of the control system (PLC programs and electric cabinets) with respect to functional and safety requirements. The know-how and accumulated experience during the last decades have lead the control engineers of SNCF to consider that a PSEEL's control system is valid if it satisfies all testing procedures of the recipe books.

The manual design phase of PLC programs and recipe books is time consuming and error-prone due to the repetitive tasks involved, the complexity of control system, and the sheer size of required deliverables. The V&V process of railway control systems is also conventionally informal and mostly manual (Vu, 2015), hence time-consuming, costly, and error-prone. The lengthy manual tests of the V&V phases, though involving around 100 test procedures, are not exhaustive enough to formally guarantee the safety of the installation.

These working conditions can cause significant workload and stress leading to human errors during both design and V&V phases of control system.

To contribute to the improvement of the design and V&V phases of PSEEL's control systems, a long-term partnership has been established since 2012 between SNCF's engineering management division and CReSTIC laboratory (of the University of Reims Champagne Ardenne). The overall aim is to improve and automate the traditional workflow as shown in Figure 3. The first phase of this partnership (Coupat et al., 2018, Coupat, 2014) focused on the improvement of design phase of PSEEL's control system. This phase resulted in the development of a method/tool for the automatic generation of deliverables to avoid repetitive tasks during design phase, in such a way that SNCF's engineers can focus their attention on cognitive tasks. It resulted not only in a reduced error, workload and stress for the engineers, but also in an estimated 115 hours reduction of the time required for a project (Coupat, 2014). As SNCF engineers have to achieve at least 10 PSEEL automation projects per year, the automatic generation of deliverables represents a total gain of at least 1150 hours per year to the SNCF.

The second phase of this partnership has started in 2015 and focuses on V&V. Its resulting improvements are twofold:

- Application of formal verification and control synthesis techniques on PLC programs in order to guarantee installation safety (Niang et al., 2017; Niang, 2018) ;
- Use of virtual commissioning for automatic verification and validation of control systems, including PLC programs and electric cabinets' wiring (Niang 2018).

The full implementation of the developed V & V techniques is currently under evaluation by the SNCF. The initial feedback from the engineers is positive in terms of maintenance, quality and safety improvements as well as overall project time and comfort.

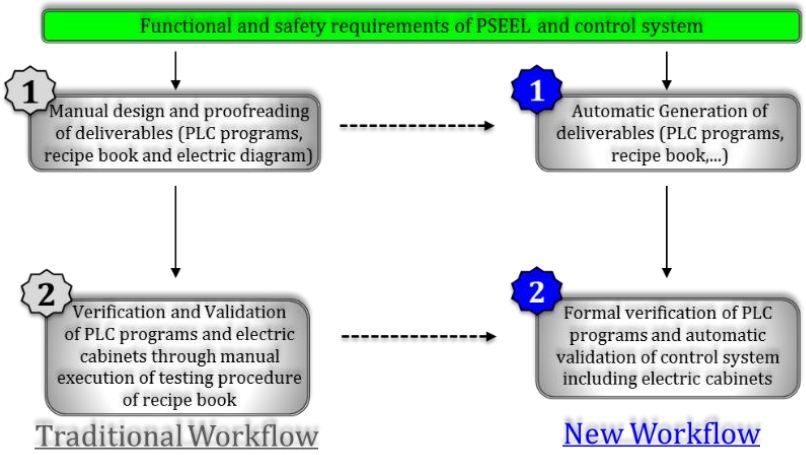


Figure 3. Improvement of the traditional engineering workflow

The main contribution of the paper is related to 1) the formulation of the overall methodology underlying the new workflow; and 2) the presentation of the new approach for V&V of PSEEL's control system.

Section 2 deals with the state of the art on design and V&V techniques of control system. Section 3 presents the design phase of the new workflow (Coupat et al., 2018) and the associated methodology for automatic generation of deliverables. The approach for the V&V phase of PSEEL's control systems is presented in Section 4. This approach involves: 1) Formal verification of safety and functional requirements of PLC programs (Section 4.1); and 2) Automatic validation of control systems (Section 4.2). Section 5 applies the formal verification approach to a transformer group.

2. STATE OF THE ART

This section presents an overview of design methods (Section 2.1) and V&V methods (Section 2.2) for control system.

2.1 Design of control system

Control engineers today mainly handle the development of industrial automation applications by direct implementation of the control task based on the interpretation of informal specification and text documents. This procedure is assisted by standardized engineering tools for the programming of PLCs. However, the informal specifications of the control software have to be manually and intuitively transferred into the

control program as they are not formally defined in practice due to a lack of time and expertise (Zaytoon & Riera, 2017). This practice (Timothy, 2007) most often leads to a deficient documentation of sequential interdependencies within the control program and additional costs caused by the erroneous interpretation of the textual requirements. To solve these problems, many formal approaches have been proposed for the design of logic controllers. An overview of formal approaches for the synthesis and implementation of logic controllers is provided in (Zaytoon & Riera, 2017).

One class of formal approaches, qualified as control synthesis, (Ramadge & Wonham, 1987; Roussel & Lesage, 2014, Vieira et al, 2017; Zaytoon & Carré-Ménétrier, 2001) aims at generating the control laws that satisfy the required properties by construction, without involvement of the designer (or at least by limiting his/her involvement as much as possible). To this end, the Supervisory Control Theory (Ramadge & Wonham, 1987), provides algorithms for the synthesis of supervisory controllers from their specifications: a supervisor is computed from two distinct automata, the first representing the discrete-event model of a given Plant and the second representing the Specification that describe the required controlled behavior of plant. The implementation of the synthesized supervisors to control industrial plants is an active research field and the major difficulties are related to: i) the state-space explosion problem when a supervisor is to be synthesized for complex systems involving many components; ii) how to model the plant and the desired specification at the granularity level required for control implementation; and iii) how to deal with the semantic and behavioral discrepancy between the abstract SCT supervisors and the resulting control realization.

Modelling languages and appropriate methods from the software engineering domain have also been adapted to the automation engineering domain to generate control code from a specification model of the controller. Though having great opportunities and a wide acceptance in academic field (Delaval, Rutten, & Marchand, 2013; Hajjar, Dumitrescu, Pietrac, & Niel, 2015; Lukman, Godena, Gray, Hericko, & Strmcnik, 2013; Thramboulidis & Frey, 2011; Witsch & Vogel-Heuser, 2009), most of those methods and tools still lack acceptance in industrial practice. The reason is twofold: on the one hand formal methods proposed in the literature are based on modeling languages which are usually not familiar to control practitioners. On the other hand, most model-driven approaches are not easy to apply in practice since they only allow performing modifications and revisions within the (formal) models. However, practical experience of the PLC engineering shows that the major modifications are directly implemented within the PLC code and thus need to be re-documented into the corresponding specification.

In the industrial automation field, companies like Siemens (with Comos[1]) and Schneider Electric (with Unity Application Generator, UAG[2]) propose environments dedicated to the description of a specific system, which can lead to the generation of the code corresponding to their PLC programming platform (TIA, Unity Pro, …). Even if a standard like PLCopen[3] proposes a standardized design format (in XML files) of PLC programs, the compatibility between PLC programs from different PLC manufacturers is not a reality today. Specific tools have therefore been developed in industry (particularly for manufacturing systems) to support the control engineer's work at the different stages of controller design, implementation and test, independently of the hardware. The main idea is to offer a fully customizable software environment (like ODIL from Prosyst[4], and ControlBuild from Dassault Systemes) for performing automation studies

[1] www.siemens.com/comos
[2] www.schneider-electric.com/en/product-range-presentation/628-uag-unity-application-generator/
[3] www.plcopen.org
[4] www.prosyst.fr

and generating the associated documentation, PLC programs and Human Machine Interfaces.

These open Automation Software Platforms allow seamless progress through all phases of the application development cycle and offer the advantage of using DSM (Domain Specific Modeling) to ensure the consistency and quality of deliverables. These innovative environments for designing and validating control software applications are based on a model-driven approach and supported by a structured set of libraries. They enable the control engineer to model, simulate, test, and deploy the IEC 61131-3 control applications. Key Highlights and Benefits of these tools for industry are: project time reduction (models and data are defined only once), development costs, quality and safety improvements, management and minimization of plant commissioning risks, maintenance of control software, compliance with industry safety-related standards such as~~,~~ EN 50123 (EN 50123, 2004) and IEC 61508-1[5]. However, these tools, to be really efficient and easy to use, must be linked to formal approaches to test and to guarantee the quality of the deliverables and particularly the generated PLC code. Consequently, they require modifying the traditional workflow for automation projects.

2.2 V&V of control system

The process of verification and validation (V&V) is essential for any successful automation project. It ensures that control systems meet the functional specifications while it guarantees installation safety. Conventionally, the verification and validation process of PSEEL control systems is informal and mostly manual, hence time-consuming, costly, and error-prone (Vu, 2015). Thus, the improvement of V&V for railway control systems is an active research topic, investigated by several research groups (Vu, 2015; Haxthausen et al., 2010; Winter, 2012; Ferrari et al., 2011). The most used methods for V&V of control system can be classified in 3 categories: testing, virtual commissioning, and formal methods.

2.2.1 Testing

Testing is the most used approach in the industrial world (Bjørner and Havelund, 2014) and has contributed to V&V of control systems for several decades. According to the IEEE-729 standard (IEEE-729, 1983), testing is the process of exercising or evaluating a system by manual or automated means to verify that it satisfies specified requirements or to identify differences between expected and actual results (i.e. it meets the specifications, see Figure 4). Thus a control system is valid if it satisfies all testing procedures of the recipe book. Testing activities are structured into sequences of test cases, and a test case is a specification fragment covering a test requirement, i.e., the expected behavior of the system.

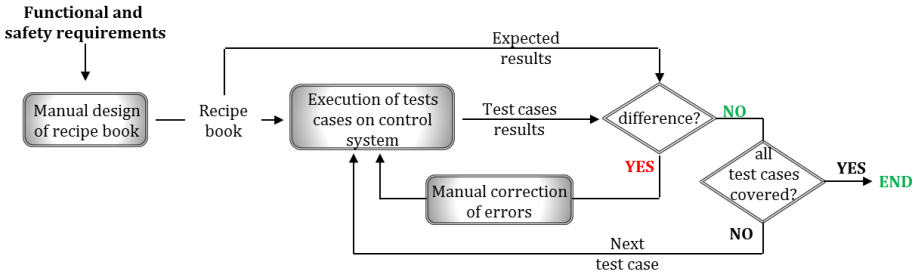


Figure 4. Testing approach

[5] https://webstore.iec.ch/preview/info_iec61508-1%7Bed2.0%7Db.pdf

Testing architecture can be open-loop or closed-loop (Babic, 2014). In open-loop testing, the input signals for a controller are test stimuli sent during test execution (Mani and Prasanna, 2016; Ma and Provost, 2019). In closed-loop testing, test stimuli are sent to the plant and then induce consequence on the controller evolution (Park et al., 2010). Test execution can either be manual or automated (Leitner et al., 2007) depending on whether they are executed without or with the assistance of tools and programs. The choice of testing mode (manual or automated) depends on various factors, including project requirements, budget, timeline, expertise, and suitability (Admin, 2014). Both manual and automated testing offer benefits and disadvantages as indicated in Table 1, which is inspired from Admin (2014), Leitner et al. (2007), and Fernandez et al. (2013).

Manual testing	**Automated testing**
not always accurate due to human error, hence less reliable	more reliable thanks to tools and/or programs assistance
time-consuming, taking up human resources	executed by software tools, hence significantly faster
more appropriate if the test cases cannot be automated or require human intervention	more appropriate if test cases can be automated
more adapted if test cases require tester's knowledge, experience, analytical/logical skills, creativity, and intuition	not well adapted in such situations
practical only if the test cases are run once or twice, and frequent repetition is not required	practical when the test cases are run repeatedly over a long time period
allows for human interaction, which stimulates user-friendliness or improved expertise	does not entail human interaction and cannot guarantee user-friendliness or enhanced customer expertise

Table 1. Comparison between manual and automated testing

Four testing levels are traditionally used to facilitate the diagnosis of possible errors in control system:

- Unit testing: consists in testing separately any block of the system (function, programs...) to verify that it meets the specifications.
- Integration testing: focuses on the interfaces between units, to make sure they work together well.
- Acceptance testing: consists in testing the whole control system in factory (Factory Acceptance Test - FAT (ISA-62381, 2012)) without connecting it to the real plant.
- Installation testing (real commissioning): The system is tested again in pre-production on site (Site Acceptance Test - SAT (ISA-62381, 2012)) with the presence of the real plant.

Installation testing of control system is only possible after integration. As a result, many detected errors must be fixed during the real commissioning, with the risks of personal injuries, damaging equipment and delays. To reduce these errors, Virtual Commissioning method is now gaining widespread attention.

2.2.2 Virtual Commissioning

Virtual commissioning (Drath et al., 2008) is a technique that is used for the development and testing of control systems (hardware and software parts) which are used for the operation of complex machines and systems. While the real commissioning of a manufacturing system involves a real plant system and a real controller, ~~the~~ virtual commissioning deals with a virtual plant model and a controller. The controller can either be i) virtual in the case of Software-In-the-Loop Simulation (SILS), or ii) real in the case of Hardware-

In-the-Loop Simulation (HILS).

The physical plant is replaced by a virtual model~~, a faithful replica of the existing one,~~ executed in a real time simulator equipped, in the case of HILS, with inputs / outputs and communicating with the control system via a physical interface (Figure 5). During the two last decades, HILS have been applied to transportation systems, traditionally in the aerospace and automotive fields and, more recently in railway systems (Baccari et al., 2012; Di Tommaso et al., 2005). SILS on the other hand is based on a fully virtual model that is used upstream to HILS (Lee and Park, 2014). The execution of PLC program is simulated in a PLC emulator (SoftPLC) which interacts directly with the real time simulator of the plant.

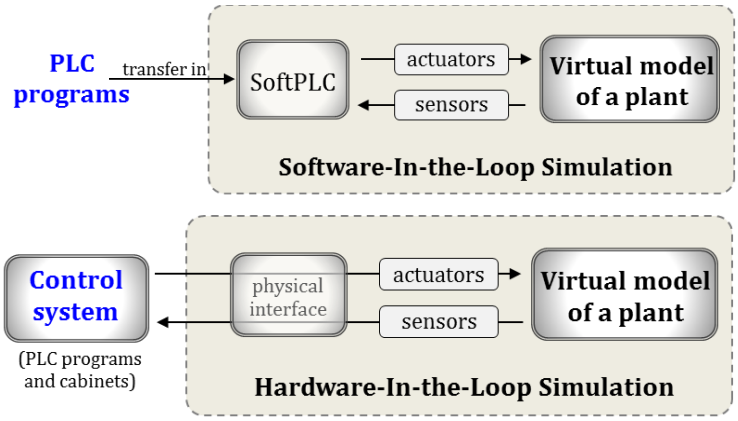


Figure 5. Software and Hardware-In-the-Loop Simulations

There are many benefits of using virtual commissioning throughout the entire engineering process (Kleijn, 2014). The total engineering time and prototype waste are reduced and the errors are less expensive to correct since they are detected earlier. Moreover, the control system quality and safety are largely improved because the engineer works on a virtual system on which he/she can test several risk-free scenarios.

2.2.3 Formal methods

Formal methods (Clarke and Wing, 1996) are mathematical techniques for the specification, development and verification of software and hardware systems. Formal verification proves the correctness of the system by checking whether a formal model of the system satisfies the properties or requirements (Fernandez et al., 2013). In contrary to other testing mechanisms that can never guarantee error-free applications, these formal techniques are efficient for V&V of control system. Therefore, formal methods are strongly recommended by many industrial standards for safety/mission-critical hardware/software systems in aerospace, aviation, defense, railway, or finance domains. For example, in the railway domain, the CENELEC standard (CENELEC - EN50128, 2012) strongly recommends the use of formal methods in development and verification of systems with the highest safety requirement.

Application of formal methods to the railway domain has been investigated by many research groups, to show how these techniques would help producing efficiently more robust railway control systems. For this purpose, Vu (2015) used formal methods to provide tools supporting efficient verification of safety-critical railway control systems, applied to "Danish railway interlocking systems". The ultimate goal is to produce methods for developing railway control systems efficiently while ensuring safety. A general overview of these trends can be found in Fantechi (2014) and Ferrari et al. (2013). Two most well-established approaches

in formal verification are model-checking (Clarke et al., 1999; Baier and Katoen, 2008) and theorem proving (Duffy, 1991; Fitting, 1996).

Model checking is a technique that relies on building a finite model of a system and checking automatically that a desired property holds in that model ~~(Ovsiannikova et al., 2017)~~. Model checkers provide counter-examples, when the model does not satisfy a given property. Generally speaking, the check is performed as an exhaustive state space search that is guaranteed to terminate if the model is finite. System behavior is formalized using automata (Moor et al., 2002) or Petri nets (Reisig, 2013) and the properties are formulated in temporal logic or automata. Model checking is limited by the state explosion problem. Because it implies an exhaustive state space traversal, the resources used in the process (memory, run-time) quickly become a limiting bottleneck. However model checking is one of the most used and promising techniques for verifying safety properties of systems thanks to its capability to be fully automated ~~(Kesraoui, 2017)~~. Some well-known tools for model-checking are NuSMV[6], SPIN (Holzmann, 2004), and Uppaal (Larsen et al., 1997). An integrated environment for formal, model-based verification of the execution control of function blocks following the IEC61499 international standard has been proposed by Vyatkin and Hanisch (2004).

Theorem proving is another well-established formal verification technique in which both the system and its expected properties are expressed as formulas in some mathematical logic. A formal system is built by defining a set of axioms and a set of inference rules. The proof method consists in finding the axioms of the properties from the axioms of the system, using the inference rules (Roussel and Faure, 2006). A major advantage of theorem proving is the avoidance of the state-explosion problem. But it involves a mathematical proof of a design's correctness and is limited by the high manual effort required.

3 AUTOMATIC GENERATION OF DELIVERABLES

The solution for automatic generation of PSEEL's deliverables (Coupat et al., 2018) dedicated to SNCF's engineers has been achieved by adapting a software package (named ODIL) to automatically generated deliverables from a graphic description which represents the PSEEL architecture. The design of such a single-line diagram (also called layout) of PSEEL is the first task of the systems engineers when handling a new a project. ODIL (Coupat et al., 2018) is an automatic generation solution developed by the company Prosyst[7] and based on DSM methods (Domain Specific Modeling) (Kelly and Tolvanen, 2008). It can be adapted to generate the required deliverables for a specific domain, in a specified format. ~~For example, the Renault car manufacturing company has adapted ODIL to generate the required deliverables through standardization.~~

3.1 Methodology for automatic generation of PSEEL's deliverables with ODIL

Figure 6 represents the steps of the methodology that has been used to adapt the ODIL solution to the domain of PSEEL. The aim of the methodology is to improve the quality of generated PSEEL deliverables by formalizing the accumulated experience of control engineers and providing standardized domain-specific templates for PLC programs and recipe books. These templates consist of rules to enrich or adapt prototypes (or examples) of predefined code. A summary of the methodology is given below and the details can be found in (Coupat et al., 2018; Coupat, 2014).

Step 1: Reverse engineering of existing PLC programs and recipe books coming from previous projects to

[6] http://nusmv.fbk.eu/
[7] www.prosyst.fr

understand their structure. This is a process of analyzing a system to obtain a representation at a high level of abstraction (Chikofsky and Cross, 1990) to allow the engineers to standardize the programming principles and to ensure that the existing PLC code is not erroneous.

Step 2: Standardization of deliverables to make them uniform. For example, a PLC program dedicated to control a circuit breaker of a Transformer Group (type 25kV AC) should always have the same structure, whereas several types of programs have been used so far by engineers to achieve the same function.

Step 3: Applying structural analysis to decompose PSEEL, control systems, and deliverables by using meta-models that are defined through DSM.

Step 4: Formalization and definition of a set of templates necessary for automatic generation of deliverables. The templates are pieces of code specific for each PSEEL's sub-system.

Step 5: Integration of the templates into the data structure of ODIL software to generate the deliverables (PSEEL and recipe book models, Uppaal models, Functional and Safety requirements, PLC programs) to be used during the second phase of verification and validation.

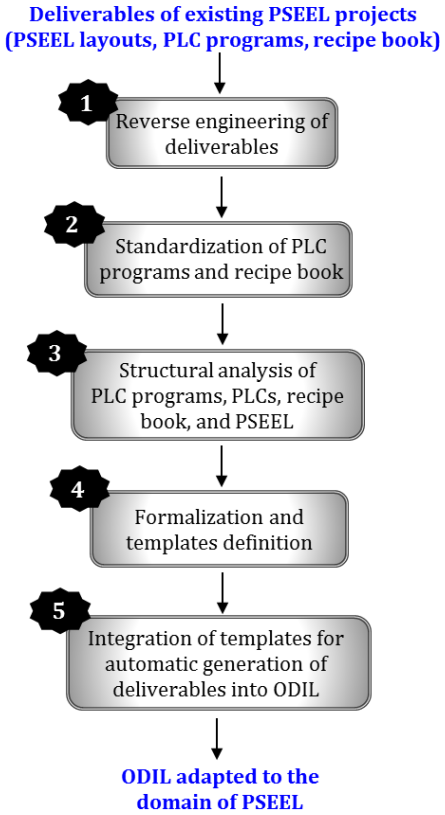


Figure 6. Adaptation of ODIL solution for automatic generation of PSEEL's deliverables

3.2 Automatic generation of PSEEL's deliverables with ODIL

The resulting adaptation of ODIL to PSEEL can now be used for the automatic generation of PLC programs and recipe book during an automation project (phase 1 of the new workflow in Fig. 3) as indicated in Figure 7.

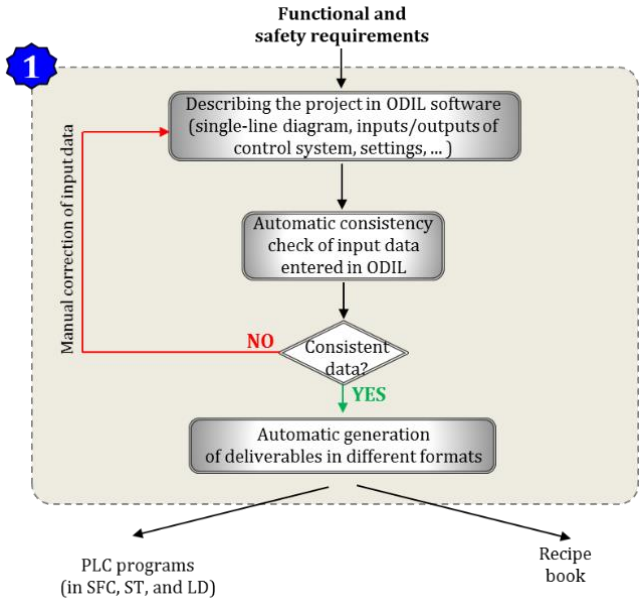


Figure 7. Automatic generation of deliverables

The first step of automatic generation of the deliverable consists in using the developed ODIL PSEEL interface to describe the input data of the project according to functional and safety requirements. These data correspond to the single-line diagram of the target PSEEL, the inputs/outputs and parameters of the control system. The engineer selects, configures, and assembles the different electric subsystems predefined in ODIL (Transformer Group, Track Feeder, High Voltage part...) in order to reconstruct the PSEEL (Figure 8). Then, according to specifications, he/she defines and parameterizes the control system's structure (number of PLCs, inputs/outputs...) before connecting its inputs/outputs to the PSEEL represented in ODIL's interface. Further to the description of the project, the software automatically generates the deliverables if the input data satisfy a set of consistency rules defined in a data structure integrated into ODIL. Some examples of consistency rules are the following:

- A PSEEL should contain at least one Transformer group and one Track feeder;
- A Track feeder should always be connected to at least one catenary;
- Each electric subsystem should be connected to one or more PLC.

In the case of inconsistency of input data, ODIL's interface provides information about the number and type of errors to help the engineer correct the errors. If the input data are consistent, ODIL automatically generates: 1) the target PLC programs dedicated to PSEEL's control system; and 2) the recipe books, traditionally used by engineers, for V&V of control systems in the factory.

In certain cases, concerning about 10% of PSEELs, the deliverables cannot be fully generated by ODIL, because of their particularities which cannot be standardized. In these cases, the engineers should manually complete the partially generated deliverables.

One of the strengths of this methodology is to translate the systems engineer's know-how into templates which correspond to their current working methods. The automatic generation avoids multiple entry of the same information in different deliverables, thus relieving the development engineers from these monotonous tasks. Then with the unique data entering, the systems engineers can optimize their mental workload and focus their attention on cognitive tasks (Coupat, 2014). Thanks to automatic generation, the design phase of

deliverables can be achieved by SNCF's engineers with 115 hours less than with the traditional manual approach requiring about 290 hours for each new PSEEL project (Coupat et al., 2018). However, this approach does not guarantee that generated PLC code is correct with respect to the safety and functional requirements. Consequently, V&V phases are still necessary.

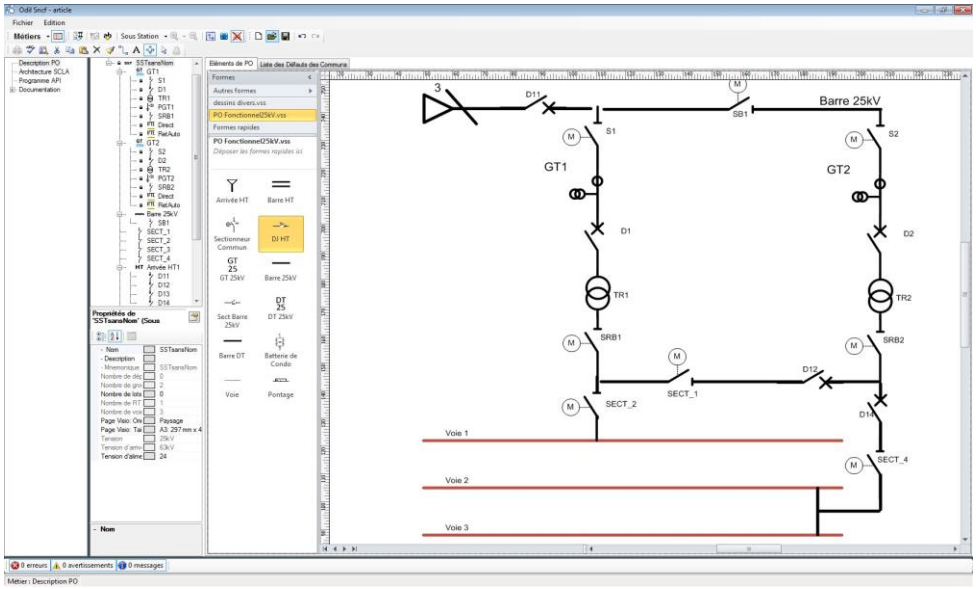


Figure 8. View of Project's description (single-line diagram) in ODIL's interface

4 FORMAL VERIFICATION AND AUTOMATIC VALIDATION OF CONTROL SYSTEM

As mentioned in Section 2.2, the most used approaches in industry for V&V of control systems are testing, virtual commissioning and formal methods. Each of these techniques has benefits and disadvantages, but they can complement each other (Fernandez et al., 2013; Constant, 2007; Rusu et al., 2004; Tretmans, 1999). The second phase of the new workflow (Figure 3) is based on the combination of these approaches to obtain a methodological approach adapted to SNCF's engineers for an efficient and reliable V&V of PSEEL's control systems. This V&V phase is presented in Figure 9. It is based on:

- Phase 2-a: formal verification of safety and functional requirements of PLC programs. This step consists of using a model checker to execute the test cases of the recipe books on the PLC programs to verify the functional requirements and help the automation engineer to correct the PLC programs in case these requirements are not satisfied. The safety requirements are verified by checking that no dangerous situations are reached for all possible inputs of the controlled plant. If the safety requirements are not satisfied, a safety-filter is introduced to synthesize a PLC program that avoids the dangerous situations. The safety and the functional requirements of the synthesized program including the filter are verified again. If the required properties are not satisfied, the filter is corrected again, and the verifications are reiterated. The use of formal verification prior to the validation phase makes it possible to detect and to correct the PLC programs as early as possible. As a result, the duration of the validation phase is reduced because any problems encountered during this phase will more likely be related to wiring errors.

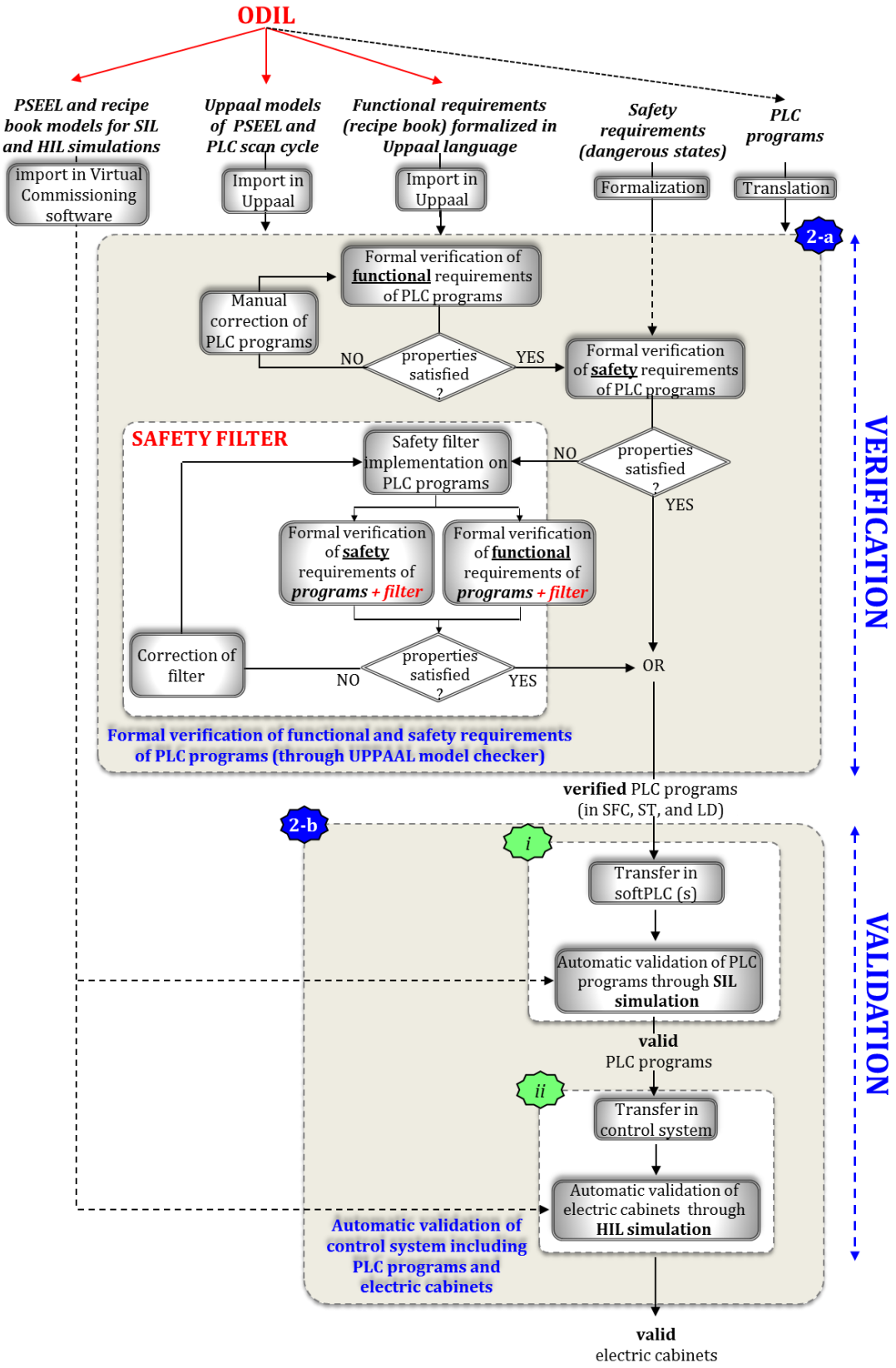


Figure 9. Formal verification of PLC programs and automatic validation of control systems

- Phase 2-b: automatic validation of PLC programs through SILS and, then, of electric cabinets in factory through HILS. The test cases are executed automatically on a virtual or real plant. PLC programs are validated by using SILS virtual commissioning. Once the control program is validated, it is transferred to the PLC based control system in the factory. The PLCs are then connected to the HILS Virtual Commissioning software through a physical interface for automatic validation of electric cabinets.

4.1 Formal verification of functional and safety requirements of PLC programs

From the various known verification techniques presented in Section 2.2.3, we focus on model checking. In this view, we have chosen to use the model checker Uppaal (Larsen et al., 1997). This tool offers a compact description language, a simulation module and a model-checker. The graphical representation of models and counter examples (or witnesses) returned after verification of properties facilitate the process of errors detection in PLC programs. In Uppaal, a system is represented by a collection of Timed Automata (TA). For each automaton, a given location represents a particular configuration of the model. A timed automaton can communicate with another one through binary synchronizations (or channels).

To formally verify the functional and safety requirements of PLC programs, the following models are required:

- Model of PLC scan cycle (formalized in TA);
- Model of the target PSEEL (in TA);
- Model of the recipe book (in TA and Boolean expressions) for formal verification of functional requirements;
- The safety requirements (representing dangerous states of the PSEEL) for formal verification of safety requirements. Boolean expressions are used to formalize these requirements;
- The target PLC programs to be verified, translated into Uppaal language (algebraic equations).

To avoid human errors during modeling, some of the above models are automatically generated by ODIL. For this, the data structure of ODIL has been extended to generate not only the traditional deliverables (PLC programs and recipe books, see Figure 7), but also the models of PSEEL electrical equipment (switch, circuit breaker…), the PLC scan cycle model and the recipe book formalized in TA (Figure 9). These models are then imported and grouped together in Uppaal to obtain the global model to be checked.

Verification of PLC programs through model checking requires the use of a behavioral model of the PSEEL's control system that matches the program execution on the target PLC. The different models indicated above are synchronized as described in Figure 10. The initialization step is executed only during the first PLC cycle to initialize the control (SFC) programs and the PSEEL state, through a function named "initialisation()". Following the initialization, the evolution of the models iterates a cycle comprising the following steps:

1- Inputs reading: all inputs of the PSEEL (sensors) are read by the PLC controlling the PSEEL. The signal "reading!" is sent by the PLC cycle's model

2- Commands reading: the PLC reads also the orders sent by operators to the PSEEL through signal "command!";

3- Timers evolution: the control programs use timing operations. This step is used to synchronize the evolution of all timers used in the PLC programs. The signal "TON!" is sent to all the models of timers (such as the one in Figure 16) so that they run in parallel with the control programs;

4- Main program execution and outputs computation using a function named "computing()". “computing()” is a stateless function which emulates the execution of PLC program by using internal variables, without interfering with the other Uppaal models;

5- Evolution of the PSEEL’s state according to the new output values of the control programs through the signal "PO!" which is sent to the PSEEL elements (such as the one in Figure 15);

6- Recipe book execution through signal "end!". The test cases contained in the recipe book are sequentially executed on the controlled PSEEL for verification of functional requirements. The state of recipe book’s execution is updated during each cycle.

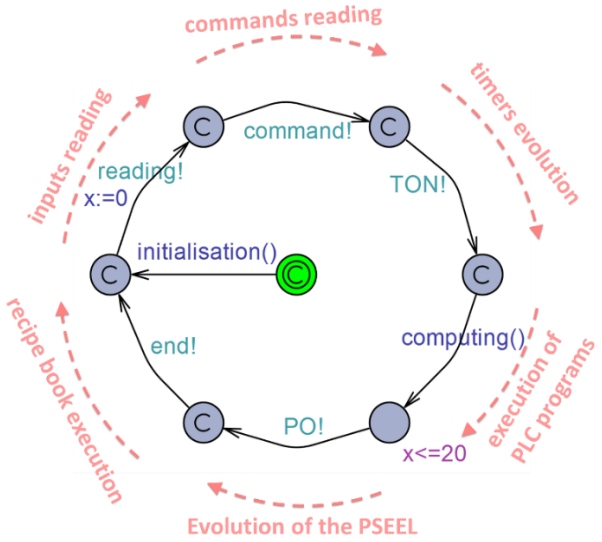


Figure 10. Main PLC – PSEEL Uppaal model

To synchronize the execution of the 6 tasks as depicted in Figure 10, the main Uppaal model, corresponding to the PLC scan cycle, is connected (binary synchronized by communicating channel) to the other models through sending messages that trigger their execution at the required time in the cycle. The structures of different models will be illustrated through the example in Section 5.

Once the cyclic evolution of the models is programmed in the model checker, we can verify the functional and safety requirements of the control programs (phase 2-a of Figure 9). The Verification of functional requirements consists in automatically executing a set of test cases of the recipe books on the PLC programs, and comparing the results with the expected ones to determine the satisfaction of each test case (Figure 11). A test case provides an initial condition and a set of sequential instruction to be executed by the operator as well as a set of expected results for each instruction. An illustration will be given in figure 19.

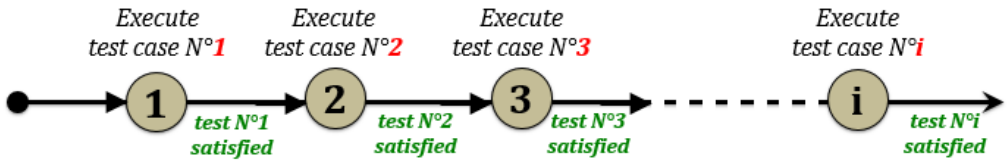


Figure 11. Principle of verification of functional requirements

The test cases of the recipe books define how the controlled plant should behave if the control programs are well designed. Therefore, the functional requirements are satisfied by the control programs only if, after test cases execution, the obtained results correspond to the expected ones. A timed automaton, synchronized with the main Uppaal model, is used to apply the test cases on the controlled plant. This timed automaton executes all the test cases sequentially, and checks that each of them is satisfied by the control programs. Otherwise, manual corrections are applied to the control programs. An example of model of recipe book will be presented in Section 5 (Figure 21).

The long-cumulated experience of SNCF engineers in the design of PSEEL control systems and the design improvement due to first phase of our methodology have shown to result in a high-quality PLC programs and recipe books. Furthermore, it has been noticed that most of the safety issues that may arise, are already implicitly taken into consideration by the engineers when they develop test cases of the recipe books. Therefore, the control errors that are not detected by a recipe book correspond to exceptional unsafe situations with minimal probability of occurrence. To avoid these residual unsafe situations, safety requirements are formulated and formally verified, without using the recipe book. Safety requirements correspond to a set of dangerous states that the controlled plant should never reach for fear of damaging the system. The verification of safety requirements consists in checking that these dangerous states are never reached for all possible inputs of the controlled plant. An additional timed automaton (see Figure 22) that generates arbitrary values of the PLC inputs (sensors, orders...) during any PLC scan cycle is therefore used to browse the whole state space of the controlled plant during the verification using Uppaal. Then, for each browsed state (characterized by a specific input/output vector), the model checker verifies whether the safety requirements, expressed as queries, are violated or not by the control program.

If a safety requirement is not satisfied, the automation engineers are not entitled to exploit the verification results to directly modify the control programs. This is due to the industrial certification constraints at SNCF, which considers that a PSEEL's control system is valid if and only if it satisfies all the testing procedures of the recipe book. Consequently, the modification of standardized PLC code, that is generated with ODIL and used by all engineers throughout the different running projects of SNCF, requires lengthy certification procedures.

Therefore, given the time constraints of the automation projects, when safety requirements are not satisfied by the control programs, a safety filter (Pichard et al., 2019) is applied to guarantee the safety of the installation (Figure 9). This safety filter is represented in the form of ~~a~~ code introduced at the end of the PLC programs, to correct (force) the PLC outputs in conformance with the safety requirements expressed as logical constraints. During each PLC cycle and before the outputs writing (evolution of the PSEEL's state in Figure 10), the filter checks whether the outputs computed by the existing control program violate the logical constraints (safety requirements) defined in the filter, and corrects the corresponding outputs.

However, knowing that the safety filter can force the PLC outputs to correct them if needed, the expected

functional behavior of a PLC program may change after the implementation of the safety filter (Marangé et al., 2009 ; Göbe et al., 2016). Therefore, it is not only necessary to verify that the safety requirements are met after the safety filter implementation, but also to check if the functional requirements (initially satisfied by the control program) are still met by the new control program including the robust filter. If these requirements are not met, the safety filter should be corrected until all requirements are satisfied by the control program (Figure 9). The correction of the safety filter may consist in modifying the priority order of the logical constraints (expressing the safety requirements) and their resolution (Pichard et al., 2019).

This iteration of the safety filter design can theoretically be non-terminating in case it is impossible to find a safety filter which is compatible with the functional requirements – for example, if the safety filter blocks outputs which are needed for the progress of the system. However, the experience conducted so far has shown that the quality of resulting PLC programs from the first design phase, when associated with a well-designed filter, should guarantee the detection and the correction of these errors in compliance with functional requirements.

The verification of the functional requirements is not subject to combinatory explosion because it is scenario-driven, through the consecutive execution of the test cases of a recipe book. Furthermore, the model of the PSEEL plant is taken into account during these executions, which limits the combinations of input values of the PSEEL sensors that are read by the PLC program. On the other hand, the verification of the safety properties requires checking the whole state space and has shown to converge in 80% of the verifications that have been conducted so far, with less than 300 seconds execution time on a "core i5" windows machine with 4Gb RAM. In the 20% remaining cases, where the safety verification does not converge, the safety property is considered to be unsatisfied. In this case, the safety filter is introduced and results in limiting the state space to browse during the subsequent iterations of the formal verification of the safety requirements.

4.2 Automatic validation of control system (PLC programs and electric cabinets)

After formal verification of PLC programs (phase 2-a), the next step (step *i* of phase 2-b, Figure 9) consists of automatic validation of PLC programs through SILS (Figure 5). For this, the verified (and corrected) PLC programs are transferred in SoftPLCs (communicating through *shared memory protocol)*, connected to a Virtual Commissioning software (Figure 12) whose input models are automatically generated by ODIL. This Virtual Commissioning software is dedicated to validation of PSEEL's control system at SNCF. The specifications of the software have therefore been established to satisfy the requirements of SNCF engineers.

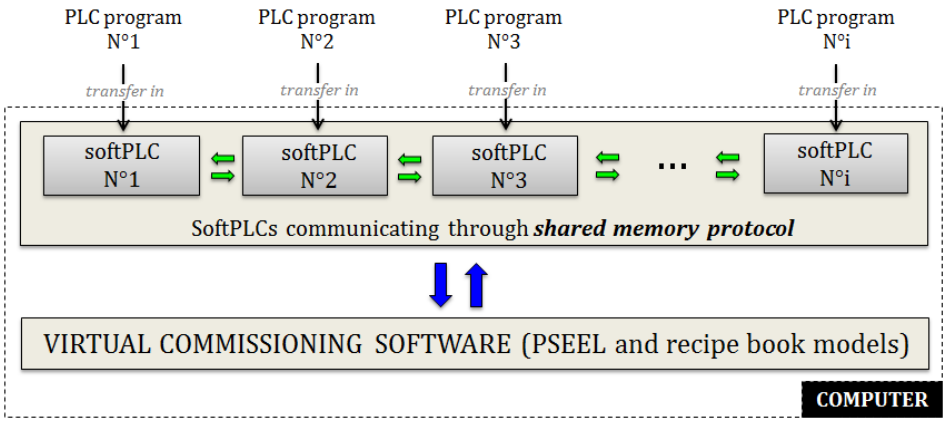


Figure 12. SILS architecture for automatic validation of PLC programs

The principle of automatic validation of PLC programs is based on automatic execution of test cases on the Virtual plant, controlled by softPLCs running the target PLC programs. For this, the generated input models for Virtual Commissioning should include not only the virtual model of the target PSEEL, but also all required tests cases contained in the recipe book. These test cases are the same as those used in the formal verification phase.

When the connection between softPLCs and the Virtual Commissioning software is established, the engineer can start the SILS for automatic validation of PLC programs. As in the case of formal verification with Uppaal Model Checker (phase 2-a), the validation phase stops when there exists a blocking instruction in the recipe book. Otherwise, the PLC programs are considered to be valid and ready for real commissioning. At the end of the validation phase, a report is generated with a trace of the test cases that are executed successfully and the blocking test cases.

In addition to automatic execution of tests cases on virtual plant, the engineer can manually control the plant by playing specific scenarios, different from tests cases included in the recipe book. In this way, he/she can, for example, verify that the dangerous scenarios – previously identified by the model checker (during formal verification of safety requirements) – cannot occur anymore further to the introduction of the safety filter.

The final step of the methodology (step ***ii*** of phase 2-b, Figure 9) is related to the automatic validation of electric cabinets, using the same Virtual Commissioning software. Once the control program is validated, it is transferred to the PLC based control system in the factory. The PLCs involved are then connected to the Virtual Commissioning software through a physical interface that is embedded in specialized computers, called cases. The architecture of the resulting HILS for automatic validation of electric cabinets is presented in Figure 13. The physical interface is composed of:

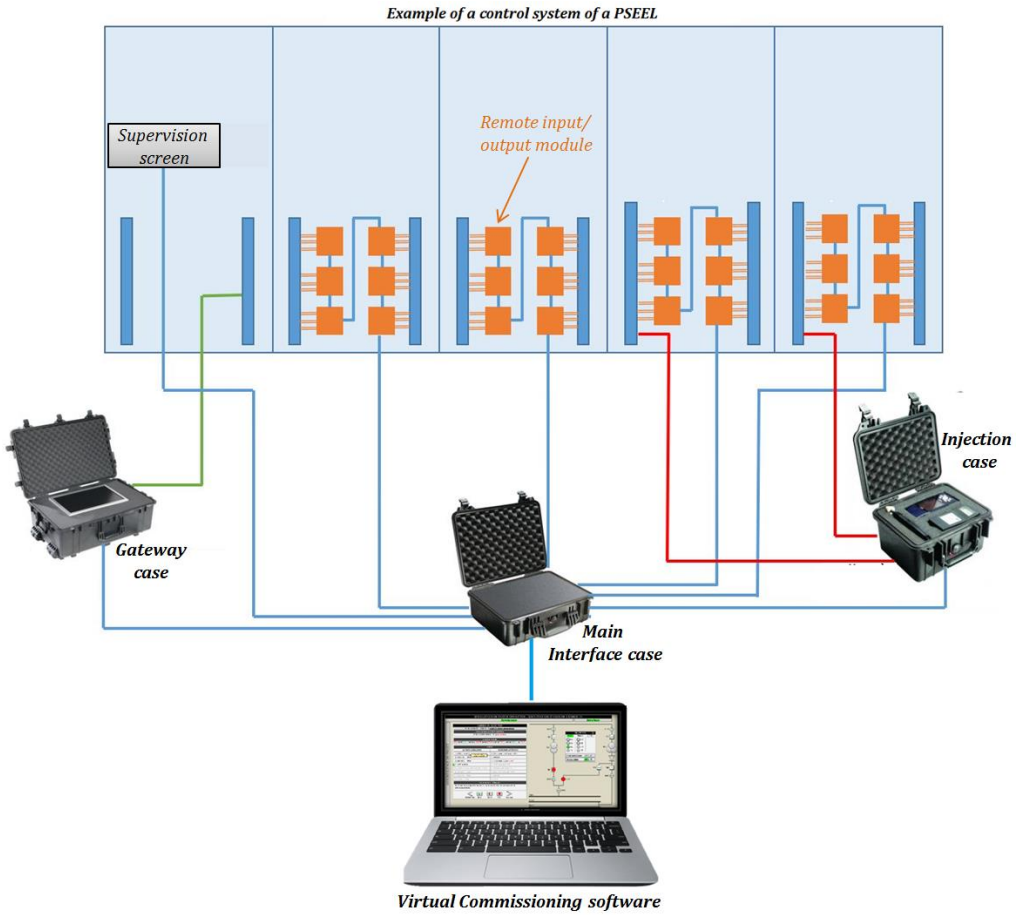


Figure 13. HILS architecture for automatic validation of cabinets

- The main case comprising a PLC and a switch. The PLC is used to collect the inputs and outputs of the control system – through remote inputs/outputs modules connected in series – and transfer them to the Virtual Commissioning software.
- An injection case used to simulate faults in the system, during the execution of tests cases.
- A gateway case used to verify the connection of the control system with the centralized supervision system, which controls remotely all PSEEL located in the same area.

The principle of the validation of electric cabinets is the same as for the SILS validation of PLC programs, through automatic execution of test cases on control system.

5 APPLICATION OF FORMAL VERIFICATION TO A TRANSFORMER GROUP

This section presents an example of application of the formal verification methodology to the Transformer Group N°1, which is controlled by one PLC and presented in Figure 1. The PLC programs are designed with STRATON software (www.copalp.com) and the programming languages used (IEC 61131-3, 2013) are Ladder Diagram (LD) and Sequential Function Chart (SFC). The objective is to formally verify these PLC programs generated by ODIL and dedicated to control the electric subsystem. The required models are presented below. To synchronize the execution of the models and tasks, the Uppaal model presented in

Figure 10 is used. This main model is represented by a timed automaton structured as a loop, which includes a clock "**x**" to measure the PLC scan cycle time (equal to 20 time units here). As the duration of inputs reading and outputs writing is negligible in the target PLC execution, six among the seven locations of the model are declared as "*committed* (C)" so that time can elapse only during the *execution of PLC programs*. Committed locations freeze time and are useful for creating atomic sequences and for encoding synchronization between multiple components.

5.1 Model of the Transformer Group

In order to verify the PLC program and its integration into the plant it has to control, a model of the Transformer Group is required. This implies a thorough knowledge of the plant, particularly the behavior of each plant element and its reaction time. In a PSEEL, there exist about 20 types of devices (switches and circuits breakers), but according to their structure and behavior (inputs/outputs), these devices can be classified in 4 generic categories. To facilitate the automatic generation of PSEEL's models, the four generic devices are modeled as discrete event systems and integrated into ODIL's data structure. For a given installation, ODIL selects and generates all required models to compose the global model of the plant.

The Transformer Group, as shown in Fig. 1, is composed of a switch "sect1", a circuit breaker "DJ1", and a transformer "Tr1". The switch "Sect1" and the circuit breaker "DJ1" belong respectively to the first and third category of generic devices. Figure 14 shows the inputs/outputs and parameters of these two devices represented as black boxes. The inputs "open" and "close" correspond respectively to the opening and closing orders of the device. These orders are sent by the corresponding PLC program, and each device is controlled by its own program. The outputs "so" and "sf" correspond respectively to the states "opened" and "closed" of the corresponding device. These outputs are fed back to the PLC programs. The parameters "TimeOP" and "TimeCL" correspond respectively to the opening and closing times of the device. Finally, "Tmin" and "Tmax" represent temporal parameters of the device.

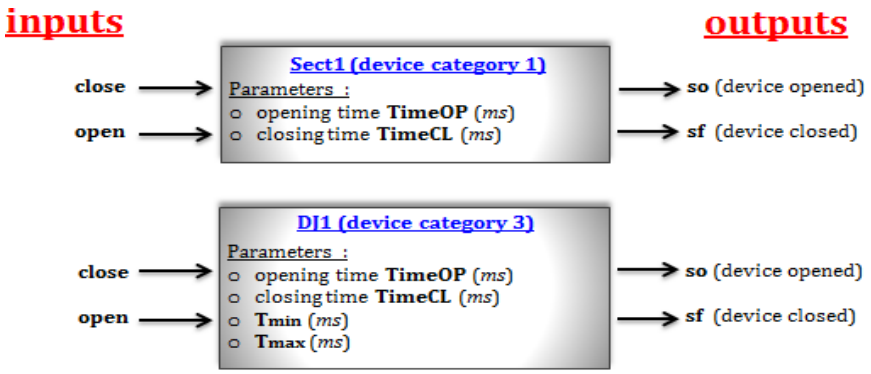


Figure 14. Structures of devices "Sect1" and "DJ1"

Though the transformer "Tr1" has a continuous behavior, it is modeled as a structure of Boolean variables representing internal faults, because these faults (overcurrent, short circuit, overheating...) are the only information that the program needs. These faults are modelled as an arbitrary fault generator (figure 22).

The above structural analysis of the Transformer Group is followed by its functional analysis to understand the behavior of the devices in order to model them in the model checker Uppaal. For example, the behavior of the switch "sect1" (device category 1) is modelled with the Uppaal timed automata presented in figure

15. The switch is initially opened, but it starts closing once it receives closing order (input "close") from its PLC program. Then, after a certain duration (TimeCL) it becomes fully closed if the order was still maintained, otherwise it returns to the initial state if the closing order was released or if opening order was activated. When the device is closed, a rising edge of input "open" is sufficient to open the device after a certain duration (TimeOP). The Uppaal model of the switch "sect1" (Figure 15) is synchronized with the signal "PO!" sent by the model of the PLC scan cycle (Figure 10). The signal "PO?" is present on each transition. The variable "x" represents the internal clock of the model, and the Boolean variable "flag" is an observer indicating the states "opening" or "closing" of the switch. Details and explanation about models of the switch, circuit breaker and transformer are presented in (Niang, 2018).

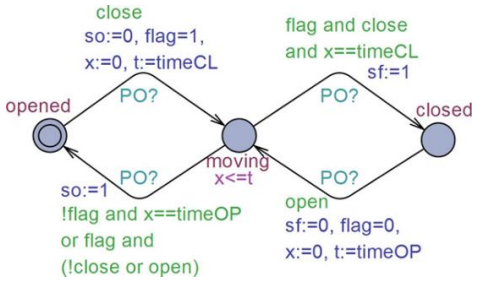


Figure 15. Uppaal model of the switch "Sect1"

5.2 Models of timers

The PLC program, mainly the SFC programs, contains timing operations described by functional blocks called TON (Timer On-delay). These TON functional blocks are described in IEC 61131-3 standard (IEC 61131-3, 2013). A TON block has:

- two input variables: 1) Boolean variable "in" to start or stop counting time; 2) time parameter "PT" indicating the timing delay (defined in Figure 18).
- and two output variables: 1) Boolean variable "Q" which equals 1 if the delay has expired; 2) time variable "x" which gives the time elapsed from the last rising edge of the input "in".

The timers are modeled in Uppaal as timed automata that run in parallel with the control programs. During the instantiation of timers, the above input/output variables are set for each instance of the timer, which is synchronized with the PLC cycle's model through the signal "**TON!**" (Figure 16). Details about this model are presented in (Niang, 2018).

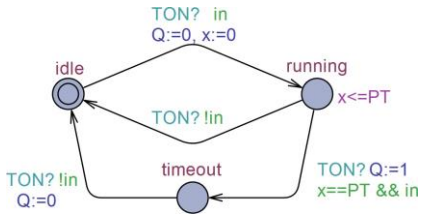


Figure 16. Uppaal model of a Timer TON

5.3 Translation of PLC programs into algebraic equations

The PLC program of the Transformer Group is translated and embedded in the model of the PLC cycle. The main program of the Transformer Group, which is executed during each PLC cycle ("computing()"

function in Figure 10), contains:

- a program controlling the switch "sect1" (in SFC language),
- a program controlling the circuit breaker "DJ1" (in SFC language),
- a program for the Transformer (in LD language).

These control programs communicate with each other through shared variables. For formal verification, the Model-Checker Uppaal does not accept Ladder Diagram or SFC language but only textual equations or representation by automata. For programs coded into Ladder Diagram (LD), the translation into algebraic equations is trivial and consists in interpreting several simple logic functions (Seabra et al. 2007). On the other hand, the SFC programs are translated into algebraic equations that are sequentially executed by computing: 1) the clearing conditions of the transitions, 2) the values of the step variables, and 3) the actions (Machado et al.).

Figure 17 presents the SFC program of the switch "sect1". The translated program, written in algebraic equations, is depicted in Figure 18. The function of a transition, *ft*, is given by a conjunction of the state of the previous steps and the logical condition associated with the transition. A step is activated if the function of one of its input transitions is true. It remains active as long as the function of one of its output transitions is false. The actions associated with a step are active when the step is active. A tool has been developed for automatic translation of SFC and LD control programs into algebraic equations in Uppaal. This tool is based on the structured methodology proposed in (Machado et al., 2006). Figure 18 corresponds to the translation (into Boolean equations) of the SFC control program of a switch. As in a real control system, this control program aims to calculate the outputs (or orders) “open” and “close”, before sending them to the model of the switch (Fig. 15). So the only variables of Figure 18, used by the Uppaal model, are the outputs “open” and “close” dedicated to open and close the switch.

Each output is maintained for a minimal time to guarantee its execution and the reception of the signal indicating the state (open or closed) of the component involved. For example, the “*close*” order associated with step *x2* of the SFC of figure 17 is activated during 80ms. This time is expressed by a Boolean information *Qtmp* which is updated by the timers’ evolution in steps *x2*, *x4*, and *x101* in figure 18 (PT=(x2: ?80:PT) …).

After the translation of the control programs, they are instantiated into the main program "computing()" which executed during any PLC cycle.

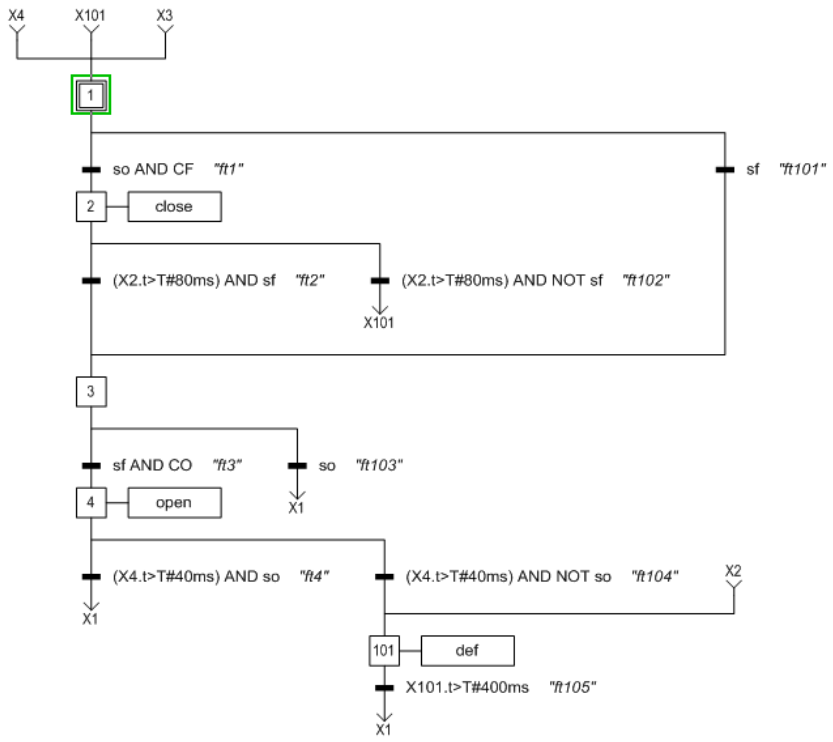


Figure 17. Control program for switch "Sect1" in SFC

```
void CmdAPPi( bool &x1, bool &x2, bool &x3, bool &x4,          // Declaration of variables
              bool &x101, bool &so, bool &sf, bool &CO, bool &CF,
              bool &open, bool &close, bool &def, bool &intmp, bool &Qtmp, int &PT)
{
     bool ft1, ft2, ft3, ft4, ft101, ft102, ft103, ft104, ft105;

 // Equations of transitions
     ft1   = x1 and so and CF;
     ft2   = x2 and sf and Qtmp;
     ft3   = x3 and sf and CO;
     ft4   = x4 and so and Qtmp;
     ft101 = x1 and sf;
     ft102 = x2 and !sf and Qtmp;
     ft103 = x3 and so;
     ft104 = x4 and !so and Qtmp;
     ft105 = x101 and Qtmp;

 // Equations of steps
     x1   = ft4 or ft105 or ft103 or x1 and !ft1 and !ft101;
     x2   = ft1 or x2 and !ft2 and !ft102;
     x3   = ft101 or ft2 or x3 and !ft3 and !ft103;
     x4   = ft3 or x4 and !ft4 and !ft104;
     x101 = ft104 or ft102 or x101 and !ft105;

 // Timers evolution in steps x2, x4, and x101
     intmp = x2 or x4 or (x101 and !ft104);
     PT=(x2 ?80:PT);  PT:=(x4 ?40:PT);  PT:=(x101 ?400:PT);

 // Equations of actions
     close  = x2;
     open   = x4;
     def = x101;
}
```

Figure 18. Control program for switch "Sect1" translated into algebraic equations

5.4 Formalization and verification of functional requirements (testing procedures)

The verification of functional requirements consists in verifying that the PLC program satisfies a set of test cases defined in the recipe book. These test cases are not exhaustive enough to formally guarantee the safety of the installation.

For the Transformer Group defined in Figure 1, ten test cases are required to validate its functional requirements. Figure 19 presents the first test case, entitled "Circuit breaker's reaction after overcurrent". It consists in verifying that if an overcurrent fault appears in the Transformer Group, the circuit breaker "DJ1" remains open as long as the fault has not disappeared.

To verify with Uppaal if the control program satisfies this test case, the latter should be formalized in the model checker. A test case is, somehow, similar to a SFC program, because it is mainly composed of: 1) initial condition, actions (instructions), and 2) transitions (Expected results). Thus, the formalization of a test case consists in translating it into a SFC program, and then into algebraic equations as we did for the SFC control program of the switch "sect1". The method of self-holding programming is also used to translate SFC program into algebraic equations. The translation of the test case of Figure 19 is depicted in Figure 20. This translation should be applied to each test case.

Circuit breaker's reaction after overcurrent	
Initial conditions	
Switch "Sect1" and Circuit breaker "DJ1" closed	
High Voltage presence (AbsU = 0)	
No defaults	
Tests	
Instructions	**Expected results**
1. Create an overcurrent default	**1**. "DJ1" opening, input "Imax" activated
2. Close circuit breaker "DJ1"	**2**. Nothingness
3. Stop overcurrent default and close "DJ1"	**3**. Input "Imax" desactivated, "DJ1" closed

Figure 19. First test case of recipe book

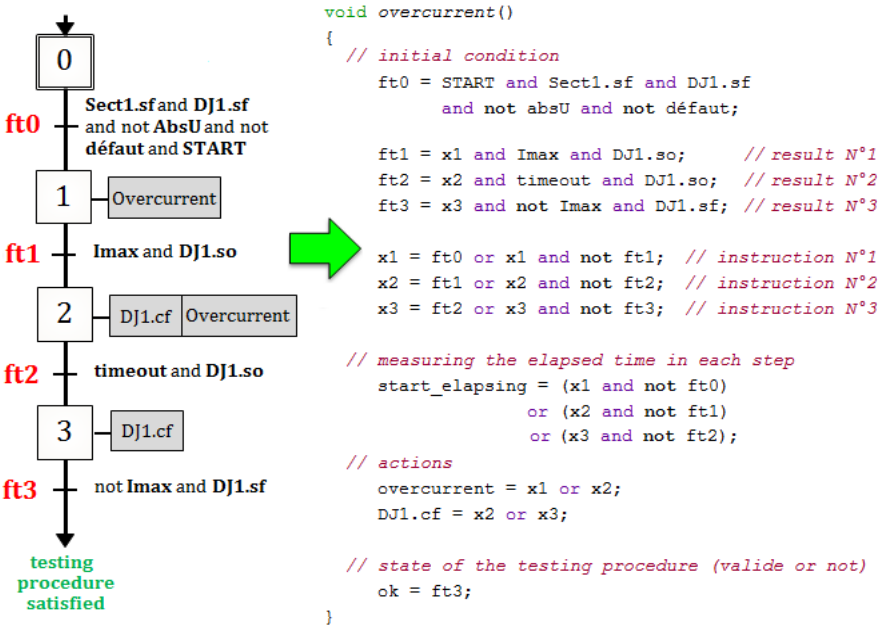


Figure 20. Translation of the first test case into SFC and then into algebraic equations

A timed automaton is then used to automatically execute the test cases of the recipe book. The recipe book for the controlled Transformer Group is represented by the timed automaton of Figure 21. Each location of the automaton (except the initial location entitled "begin" and last one "end") corresponds to one of the 10 consecutive test cases of the recipe book. A test case is synchronized with the model of the PLC cycle through the signal "end!" (Recipe book execution step in Figure 10) in such a way that the recipe book's evolution state is updated at the end of each PLC cycle. The label "ok" is a guard which signals whether the running test case has been satisfied or not. This label is updated by the last algebraic equation of the test case (Figure 20), which corresponds to the successful execution of the final transition of the corresponding SFC (transition "ft3" of Figure 20). Therefore, a test case of the timed automata representing the recipe book (Figure 21) is executed only if the previous one (if it exists) is satisfied by the control program.

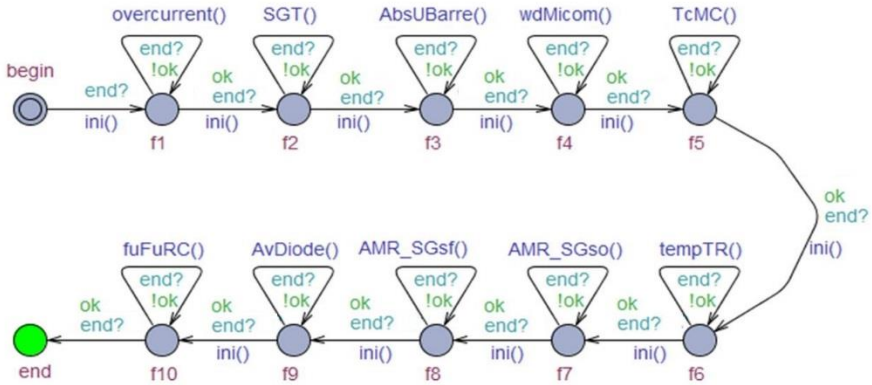


Figure 21. Timed automaton of a recipe book

To verify if the PLC program of the Transformer Group satisfies all testing procedures, it is enough to check if there exist at least one blocking instruction in the recipe book. An instruction is blocking if, after its execution on the controlled Transformer Group, the corresponding expected result is not obtained after a

certain duration. This property can be easily checked in Uppaal by ~~to~~ the following "query" which checks whether or not the timeout of the "RecipeBook" timer is signaled:

$$E <> RecipeBook.timeout$$

The "RecipeBook" timer counts, for each instruction of recipe book, the elapsed time between the execution of the instruction and the obtainment of the corresponding expected result. The *timeout* represents an elapsed time which cannot be attained if the program is valid. If the *timeout* is attained, the program presents an error because this means that the output transition of the current active step of the corresponding SFC program cannot be fired. The delay of the *timeout* is quantified by the upper threshold of the test case that has the longest execution time. The satisfaction of this query will therefore mean that there exists at least one blocking instruction. Then, the model checker Uppaal provides the path – a witness – ~~(a witness in this case)~~ leading to the blocking instruction and the corresponding testing procedure.

The verification of functional requirements of the Transformer Group's control program were performed on a "core i5" windows machine with 4Gb RAM. It takes at most 70 milliseconds for the model checker to return a blocking instruction if the program contains errors. In this case, the path returned by the model checker Uppaal helps the engineers to diagnose and correct the corresponding error of the PLC programs. The modeling and verification procedures of functional requirements are further detailed in (Niang, 2018).

5.5 Formalization and verification of safety requirements

Two examples of safety requirements of the Transformer Group (named property 1 and property 2) are defined below:

- property 1: never attempt to open the circuit breaker "DJ1" in charge, i.e., when the switch "sect1" is closed, expressed by the following query: *E<>cycle.end and tempoDef.timeout and APP2.sf*
- property 2: any fault that appears in the Transformer Group must be eliminated by the circuit breaker "DJ1" within 300 milliseconds, expressed by the following query: *E<>cycle.end and tempoDef.timeout and DJ1.sf*

Some of the 10 testing procedures dedicated to the Transformer Group (Figure 21) are intended to verify that the dangerous states (corresponding to properties 1 and 2) are unreachable. For example, the second test case in Figure 21 (named *"SGT()"*) consists in putting the Transformer Group in an initial state, then executing the predefined scenario and verifying if the circuit breaker "DJ1" can be opened in charge. However, even if this testing procedure is satisfied by the control program, it does not imply that property 1 will always be satisfied, because there may exist other scenarios of the control program that violate this property. A more exhaustive approach for formal verification of safety requirements is therefore needed. For this, the dangerous states should be expressed as queries in Uppaal. Then, the models in Figure 22 are used to generate arbitrary values of the system's inputs for exhaustive verification (as has been explained in Section 4.1).

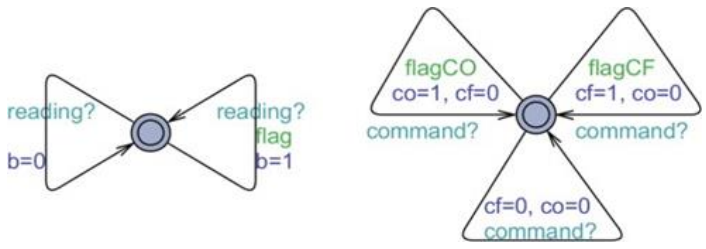


Figure 22. Arbitrary generator of input values

The left-hand side model of Figure 22 is synchronized with the PLC cycle's model through the signal "reading?" (inputs reading step of Figure 10), whereas the right-hand side model is synchronized through the signal "command?" ("commands reading" step of Figure 10). During any PLC cycle, the first model generates arbitrary values for one Boolean input (a sensor, a fault...), while the second model generates arbitrary command sent by the operator, such as the opening or closing of a switch or a circuit-breaker. These models are instantiated several times in the global model, depending on the number of inputs and devices of the plant. To limit the state space during formal verification of safety requirements, only faults or sensor inputs relative to the given test case are generated. Guards are therefore associated with the transitions of these models. For example, the guard "flag" of the first model of Figure 22 is a Boolean variable which allows the activation of the fault "b" only if it will make the control program evolve during the current PLC cycle. "flagCO" and "flagCF" are used in a similar way for the second model of Figure 22.

The formal verification of safety requirements consists in verifying in Uppaal (through a query) "*if there exists a scenario (or inputs vector) that can occur in the system and leads it to violate the safety requirements*". When the properties are violated, the model checker returns the scenario(s) leading to the dangerous state(s) to help the engineer correct the errors. Formal verification results showed that the safety requirements are not formally satisfied by the control program of the given example. In fact we have proved that, even if their occurrence is rare, there exist some scenarios that can lead the control program to violate the safety requirements, and the corresponding errors must be corrected. Due to the requirement to search the whole state space, the verification of safety properties requires much more computer time (about 30 seconds for our example using a "core i5" windows machine with 4Gb RAM) than that required for the verification of functional properties.

While this formal technique allows engineers to detect errors in PLC programs, it does not allow them to correct the program to ensure it never violates the safety requirements. For this purpose, we implement a safety filter in the PLC, as proposed by (Pichard et al., 2019) as a piece of ST code placed at the end of the program (Figure 23) to formally guarantee the safety of installation. For example, let's consider that the designed control program has violated the property 1 (opening "DJ1" in charge). When the operator needs to open the circuit breaker, the safety filter checks first (thanks to logical constraints) if the request corresponds to an opening order in charge. In this case, the opening request is filtered and not sent to the circuit breaker.

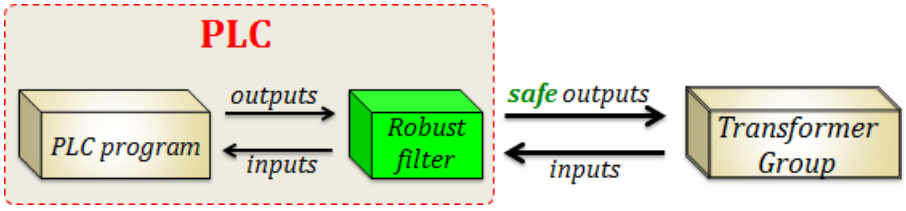


Figure 23. Principe of the safety filter

The filter is therefore used to complement (or refine) the existing control program to obtain the final and safe control program. In this way, the engineer is not required to modify the PLC program in order to guarantee its safe execution. The design method of the safety filter is presented in (Pichard et al., 2019).

To formally verify that the safety filter guarantees the safety of installation as expected, it was implemented in the

model of the PLC program controlling the Transformer Group. Then we formally verify if the safety requirements have been met. The design of the safety filter is based on algebraic equations. Therefore its implementation in the model of the main program "computing()" (Figure 10) does not need any translation. As expected, the safety requirements are no longer violated by the control program thanks to the safety filter, while the functional requirements are still satisfied in our example.

6. CONCLUSION

The aim of the methodological approach presented in this paper is to bridge the gap between formal approaches in academia and industrial applications by proposing an original workflow for automation study. This workflow focuses on the improvement of the traditional workflow of SNCF engineers, which involves a design and a V&V phases of PSEEL's control systems, without modifying the way they are used to program PLC. The traditional design phase is replaced by an automatic generation process of deliverables, and the manual testing approach for V&V of control system is replaced by a methodological approach that encompasses formal verification of PLC programs, automatic validation of PLC programs (through SIL), and automatic validation of electric cabinets (through HIL). The recipe book, which is traditionally used to perform the tests, is automatically generated and used as the requirement specification model for formal verification of functional properties.

Formal verification compensates the lack of exhaustiveness of the traditional testing approach, and the implementation of the safety filter guarantees that the control system satisfies the safety requirements without modifying the initial PLC code. The Virtual Commissioning technique allows the engineers to automate the Validation procedure of the control system, and to provide interactive traceability for critical and faulty situations. This technique is also useful for training new operators.

The new workflow has been implemented by SNCF and has improved the working conditions of SNCF engineers during automation projects while guaranteeing the safety of installation. Thanks to automatic generation of deliverables, the repetitive tasks are avoided, and human errors are reduced during the design phase.

References

ADMIN, A. Automated Testing vs Manual Testing: Which Should You Use, and When?, Nov. 2014, https://www.apicasystems.com/blog/automated-testing-vs-manual-testing/

IEEE 729 (ANSI) - IEEE Standard Glossary of Software Engineering Terminology, 1983.

BABIC, J. Model-based approach to real-time embedded control systems development with legacy components integration, Doctoral dissertation, Sveuciliste u Zagrebu, 2014.

BACCARI, S., CAMMEO, G., DUFOUR, C., IANNELLI, L., MUNGIGUERRA, V., PORZIO, M., REALE, G., AND VASCA, F. Real-Time Hardware-in-the-Loop in Railway: Simulations for Testing Control Software of Electromechanical Train Components. Railway Safety, Reliability, and Security: Technologies and Systems Engineering, 2012, pp. 221–248.

BAIER, C., KATOEN, J.-P. Principles of model checking. The MIT Press, Cambridge, Mass, 2008.

BARGER, P., THIRIET, J-M., ROBERT, M., AUBRY, J-F., Dependability Study in Distributed Control Systems Integrating Smart Devices, IFAC Proceedings Volumes, Volume 37, Issue 5, 2004, pp. 67-72, ISSN 1474-6670.

BJØRNER, D., HAVELUND, K. 40 Years of Formal Methods. Lecture Notes in Computer Science. Springer, Cham, May 2014, pp. 42–61.

CHIKOFSKY, E.J., CROSS, J.H. Reverse engineering and design recovery: A taxonomy, IEEE Software 7, 1990. pp. 13–17. doi:10.1109/52.43044

CLARKE, E. M., AND WING, J. M. Formal Methods: State of the Art and Future Directions. ACM Computing Surveys 28, 1996, pp. 626–643.

CLARKE, E. M., GRUMBERG, O., AND PELED, D. A. Model Checking. MIT Press, Cambridge, MA, USA, 1999.

CONSTANT, C., JÉRON, T., MARCHAND, H., RUSU, V. Integrating formal verification and conformance testing for reactive systems. IEEE Transactions on Software Engineering 33, 2007, pp. 558–574.

COUPAT, R. Méthodologie pour les études d'automatisation et la génération automatique de programmes Automates Programmables Industriels sûrs de fonctionnement. Application aux Equipements d'Alimentation des Lignes Électrifiées. PhD thesis, University of Reims Champagne Ardenne, Reims, Nov. 2014.

COUPAT, R., PHILIPPOT, A., NIANG, M., COURTOIS, C., ANNEBICQUE, D., AND RIERA, B. Methodology for Railway Automation Study and Automatic Generation of PLC Programs. IEEE Intelligent Transportation Systems Magazine, vol. 10, no. 3, pp. 80-93, 2018.

DELAVAL, G., RUTTEN, E., & MARCHAND, H. Integrating discrete controller synthesis into a reactive programming language compiler. *Discrete Event Dynamic Systems*, 23, 2013, pp. 385-418.

DI TOMMASO, P., FLAMMINI, F., LAZZARO, A., PELLECCHIA, R., SANSEVIERO, A. The simulation of anomalies in the functional testing of the ERTMS/ETCS trackside system, Ninth IEEE International Symposium on High-Assurance Systems Engineering (HASE'05), 2005.

DRATH, R., WEBER, P., MAUSER, N. An evolutionary approach for the industrial introduction of virtual commissioning. In 2008 IEEE International Conference on Emerging Technologies and Factory Automation, 2008, pp. 5–8.

DUFFY, D. A. Principles of Automated Theorem Proving. John Wiley & Sons, Inc., New York, NY, USA, 1991.

EN 50123 - Railway applications - Fixed installations - D.C. switchgear, 2004

EN 50128 - Railway applications - Communication, signalling and processing systems - Software for railway control and protection systems Engineering 360, 2012.

EN 50126 : Applications ferroviaires - Spécification et démonstration de la fiabilité, de la disponibilité, de la maintenabilité et de la sécurité (FDMS), 2012.

FANTECHI, A. Twenty-five years of formal methods and railways: What next? In Revised Selected Papers of the SEFM 2013 Collocated Workshops on Software Engineering and Formal Methods – vol. 8368, Berlin, Heidelberg, 2014, Springer-Verlag, pp. 167–183.

FERNANDEZ, B., BLANCO, E., MEREZHIN, A. Testing & verification of PLC code for process control,

ICALEPCS conference, San Francisco, USA," vol. 14, p. 5, Oct. 2013.

FERRARI, A., FANTECHI, A., GNESI, S., MAGNANI, G. Model-based development and formal methods in the railway industry. IEEE Softw. 30, 3, May 2013, pp. 28–34.

FERRARI, A., MAGNANI, G., GRASSO, D., FANTECHI, A. Model checking interlocking control tables. In FORMS/FORMAT 2010 (Berlin, Heidelberg, 2011), E. Schnieder and G. Tarnai, Eds., Springer Berlin Heidelberg, pp. 107–115.

FITTING, M. First-Order Logic and Automated Theorem Proving, 2 ed. Texts in Computer Science. Springer-Verlag, New York, 1996.

GOBE, F., TIMMERMANNS, T., NEY, O., and KOWALEWSKI, S., "Synthesis Tool for Automation Controller Supervision", in Proc. 2016 13th International Workshop on Discrete Event Systems (WODES)

HAJJAR, S., DUMITRESCU, E., PIETRAC, L., NIEL, E. (2015). Synthesizing safe control-command systems out of reusable components. *Control Engineering Practice*, 44, pp. 243-259

HAXTHAUSEN, A. E., LE BLIGUET, M., KJÆR, A. A. Modelling and verification of relay interlocking systems. In Foundations of Computer Software. Future Trends and Techniques for Development, Berlin, Heidelberg, 2010, C. Choppy and O. Sokolsky, Eds., Springer Berlin Heidelberg, pp. 141–153.

HOLZMANN, G. J. The SPIN model checker: Primer and reference manual. Addison-Wesley Reading, 2004.

IEC 61499-1, Function Blocks—Part 1 Architecture, International Electrotechnical Commission, Geneva, International Standard, 2005.

IEC 60870-4: Telecontrol equipment and systems. Part 4: Performance requirements Ed. 1. IEC (International Electrotechnical Commission).

IEC 61131-3, I. Programmable controllers – Part 3: Programming languages, Reference number CEI/IEC 61131, 2013.

ISA-62381 (ANSI-ISA): factory acceptance test (fat), site acceptance test (sat), and site integration test (sit), automation systems in the process industry, 2012.

KELLY, S., AND TOLVANEN, J.-P. Domain-Specific Modeling: Enabling Full Code Generation. John Wiley & Sons, Apr. 2008. Google-Books-ID: GFFtRFkuU_AC.

KESRAOUI, S. Intégration des techniques de vérification formelle dans une approche de conception des systèmes de contrôle-commande : application aux architectures SCADA. PhD. Thesis, Lorient, May 2017.

LARSEN, K. G., PETTERSSON, P., YI, W. Uppaal in a nutshell. International Journal on Software Tools for Technology Transfer 1, 1-2, 1997, pp. 134–152.

LEE, C. G., AND PARK, S. C. Survey on the virtual commissioning of manufacturing systems. Journal of Computational Design and Engineering 1, 3 (July 2014), pp. 213–222.

LEITNER, A., CIUPA, I., MEYER, B., HOWARD, M. Reconciling Manual and Automated Testing: The AutoTest Experience. IEEE, 2007, pp. 261a. [Online]. Available: http://ieeexplore.ieee.org/document/4076909/

LEVINE W.S. The Control Handbook, Second Edition. CRC Press, December 2010. ISBN 9781420073669.

LUKMAN, T., GODENA, G., GRAY, J., HERICKO, M., STRMCNIK, S. (2013). Model-driven engineering of process control software – beyond device-centric abstractions. Control Engineering Practice, 21(8), pp. 1078–1096.

MA, C. and PROVOST, J. Introducing plant features to model-based testing of programmable controllers in automation systems. Control Engineering Practice. Vol. 90, September 2019, pp. 301-310.

MACHADO, J. M., DENIS, B., LESAGE, J.-J., FAURE, J.-M., FERREIRA DA SILVA, J. C. Logic Controllers Dependability Verification using a Plant Model. IFAC Proceedings vol. 39, 17, 2006, pp. 37–42.

MACHADO, J., SEABRA, E., CAMPOS, J., SOARES, F., LEAO, C. Simulation and formal verification of industrial systems controllers. ABCM Symposium Series in Mechatronics – 2008, Vol. 3 - pp.461-470.

MANI, P., PRASANNA, M., Automatic test case generation for programmable logic controller using function block diagram, in: International Conference on Information Communication and Embedded Systems (ICICES), 2016, pp. 1-4.

MARANGE P., GOUYON D., PETIN J.F. and RIERA B., Verification of functional constraints for safe product driven control. 2nd IFAC Workshop on Dependable Control of Discrete System, DCDS'092nd IFAC Workshop on Dependable Control of Discrete System, DCDS'09, 2009, pp. 315-320.

MCMILLAN, K. The SMV Language. Cadence Berkeley Labs., 1999.

MOOR, T., RAISCH, J., O'YOUNG, S. Discrete Supervisory Control of Hybrid Systems Based on l-Complete Approximations, Discrete Event Dynamic Systems, vol. 12, No. 1, 2002, pp 83–107.

NIANG, M., PHILIPPOT, A., GELLOT, F., COUPAT, R., RIERA, B., LEFEBVRE, S. Formal verification for validation of PSEEL's PLC program. The 14th International Conference on Informatics in Control, Automation and Robotics, Madrid, Spain, 2017.

NIANG, M. Vérification formelle et simulation pour la validation des systèmes de contrôle commande des EALE (Equipements d'Alimentation des Lignes Électrifiées). PhD thesis, University of Reims Champagne Ardenne, Reims, Dec. 2018.

OVSIANNIKOVA, P., CHIVILIKHIN, D., ULYANTSEV, V. and SHALYTO, A. Closed-loop verication of a compensating group drive model using synthesized formal plant model, in: Emerging Technologies and Factory Automation (ETFA), 22nd IEEE International Conference on. IEEE, 2017, pp. 1-4.

PARK, H. T., KWAK, J. G., WANG, G. N. and PARK, S. C., Plant model generation for PLC simulation, International Journal of Production Research vol. 48 (5), 2010, pp. 1517-1529.

PICHARD, R., PHILIPPOT, A., SADDEM, R., RIERA, B. Safety of Manufacturing Systems Controllers by Logical Constraints With Safety Filter. IEEE Transactions on Control Systems Technology, Vol. 27, Issue: 4, July 2019, pp. 1659–1667.

RAMADGE, P.J., WONHAM, W.M.. Supervisory control of a class of discrete event processes. *SIAM J Control Optim.*, 25(1), 1987, pp. 206 -230.

REISIG, W. Understanding Petri Nets: Modeling Techniques, Analysis Methods, Case Studies. Springer-Verlag, Berlin Heidelberg, 2013.

ROUSSEL, J.-M., FAURE, J.-M. Designing dependable logic controllers using algebraic specifications. *Control Engineering Practice*, 21(8), 2006, pp. 1078–1096.

ROUSSEL, J.-M., LESAGE, J.-J. Design of Logic Controllers Thanks to Symbolic Computation of Simultaneously Asserted Boolean Equations. *Mathematical Problems in Engineering*, vol. 14, 2014, pp. 1143-1153.

RUSU, V., MARCHAND, H., TSCHAEN, V., JÉRON, T., JEANNET, B. From Safety Verification to Safety Testing. In Testing of Communicating Systems (Testcom) (Oxford, United Kingdom), R. Groz and R. Hierons, Eds., vol. 2978 of Lecture notes in computer science, Springer, 2004, pp. 160–176.

SEABRA E., MAADO J., SOARES F., LEAO C., SILVA J., Simulation and formal verification of real-time systems: A case study. 4th International Conference on Informatics in Control, Automation and Robotics (ICINCO'2007). 9-12th may 2007, Angers, France.

THRAMBOULIDIS, K., FREY, G. Towards a model-driven IEC 61131-based development process in industrial automation. *Journal of Software Engineering and Applications*, vol. 4(4), 2011, pp. 217–226.

TIMOTHY L, J. Improving automation software dependability: A role for formal methods? Control Engineering Practice vol. 15, 11, Nov. 2007, pp. 1403–1415.

TRETMANS, J. Testing concurrent systems: A formal approach. In CONCUR'99 Concurrency Theory, Berlin, Heidelberg, 1999, J. C. M. Baeten and S. Mauw, Eds., pp. 46–65.

VIEIRA, A. D., SANTOS E. A. P., QUEIROZ, M. H., LEAL A. B., NETO A. D. P., CURY, J. E. R. A method for PLC implementation of supervisory control of discrete event systems. *IEEE Transactions on Control Systems Technology*, 25(1), 2017, pp. 175-191.

VU, L. H. Formal Development and Verification of Railway Control Systems - In the context of ERTMS/ETCS Level 2. PhD thesis, Technical University of Denmark (DTU), 2015.

VYATKIN V., HANISCH, H.-M., Verification of distributed control systems in intelligent manufacturing, Journal of Intelligent Manufacturing February 2003, Volume 14, Issue 1, pp. 123–136.

WINTER, K. Optimising ordering strategies for symbolic model checking of railway interlockings. In Leveraging Applications of Formal Methods, Verification and Validation. Applications and Case Studies, 2012, T. Margaria and B. Steffen, Eds., Springer Berlin Heidelberg, pp. 246–260.

WITSCH, D., VOGEL-HEUSER, B. Close integration between UML and IEC 61131-3: New possibilities through object-oriented extension. In: Proceedings of the IEEE Emerging Technologies & Factory Automation, Mallorca, 2009.

ZAYTOON, J., CARRÉ-MÉNÉTRIER, V. Synthesis of control implementation for discrete manufacturing systems. *International Journal of Production Research*, 39(2), 2001, pp. 329-345.

ZAYTOON, J., RIERA, B. Synthesis and implementation of logic controllers – A review. Annual Reviews in Control 43, 2017, pp. 152–168.