TOWARDS AUTOMATIC GENERATION OF FAIL-SAFE PLC CODE COMPLIANT WITH FUNCTIONAL SAFETY STANDARDS

A. Germinario[*], B. Fernández, E. Blanco, CERN, Geneva, Switzerland

Abstract

In agreement with the IEC 61511 functional safety standard, fail-safe application programs should be written using a Limited Variability Language (LVL), that has a limited number of operations and data types, such as LD (Ladder Diagrams) or FBD (Function Block Diagrams) for safety PLC (Programmable Logic Controller) languages.

The specification of safety instrumented systems, as part of the Safety Requirements Specification document, shall unambiguously define the logic of the program, creating a one-to-one relationship between code and specification. Hence, coding becomes a translation from a specification language to PLC code. This process is repetitive and error-prone when performed by a human.

In this paper we describe the process of fully generating Siemens TIA portal LD programs for safety applications from a formal specification. The process starts by generating an intermediate model that represents a generic LD program based on a predefined meta-model. This intermediate model is then automatically translated into code.

The idea can be expanded to other equivalent LVL languages from other PLC manufacturers. In addition, the intermediate model can be generated from different specification formalism having the same level of expressiveness as the one presented in this paper: a Cause-Effect Matrix.

Our medium-term vision is to automatically generate fail-safe programs from diverse formal specification methods and using different LVLs.

Keywords: functional safety, safety instrumented systems, PLC

INTRODUCTION

Programmable Logic Controllers (PLCs) are widely recognized as the standard for industrial process control. One of the main reason is that PLCs are robust and reliable devices that can work in harsh environments. The Mean Time Between Failures (MTBF) of PLCs are normally very high. For example, a Siemens CPU S7-1515-2 PN has a MTBF of 27.7 years (data extracted from [1]).

PLC manufacturers also provide solutions for safety systems. When in an industrial process, a particle accelerator, a machine or any other kind of system, a failure may lead to a risk for humans, the environment or a big economic and reputations loss, the project responsible must reduce the risk the to the tolerable levels defined by the organization, company or regulations. Fail-safe PLCs are devices certified by organizations like TÜV SÜD [2], that have been designed to follow the IEC 61508 [3] Functional Safety standard. These devices are used to deploy the Safety Instrumented Functions (SIF) in a way of a software program meant to reduce the risks mentioned above. The MTBF of fail-safe PLCs is also high but lower than the one of a standard PLC. For example a S7-1515F PLC has a MTBF of 24.5 years. This distinction arises from its internal architecture and increased complexity, which enables it to substantially decrease the occurrence of dangerous undetected failures when compared to a typical PLC. In discussions concerning safety-critical systems, only dangerous undetected failures are pertinent.

Fail-safe PLCs are able to detect most of their failures as stated by their Safe Failure Fraction (SFF) ≥ 99%. This means that the dangerous undetected failures are ≤ 1%. This number is so low because the safety systems internal to a safety PLC cover most of the internal failures of the controller, including hardware, operating system, firmware, etc. On the contrary, it clearly does not cover the user PLC application program (AP) for the SIFs.

The IEC 61511 standard [4] provides the guidelines to develop Safety Instrumented Systems (SIS) for industrial processes. The Clause 12 of the IEC 61511-1 focuses in the SIS AP development requirements. Among other things, it recommends the usage of Low Variability Languages (LVL) to write these programs.

Code generation is a common practice in software engineering. It has a lot of advantages in quality and efficiency like reducing the amount of coding errors introduced by the programmer, speeding up the development process, etc. For PLC programs, there are not many available tools for code generation. A good example is *PLC coder* from MathWorks [5], where ST (Structured Text) programs from the IEC61131-3 [6] can be generated automatically from Simulink models. At CERN, we use the *UNICOS* [7] framework to generate PLC programs from high level specifications.

However, most of the available tools cannot generate safety PLC programs compliant with the IEC 61511 standard. In addition, most of PLCs brands did not allow to generate safety PLC programs with external tools, import them in their programming environment and compile them as safety programs. Very recently the Totally Integrated Automation Portal (TIA portal) [8], the programming environment of Siemens PLCs, opened the door for code generation of safety PLC programs. Now source code files written in Ladder Diagrams (LD) or Function Block Diagrams (FBD) with certain restrictions to be compatible with the LVL requirements, can be imported and compiled as safety programs.

There are a very few references in the Functional Safety standards about code generation. Since this is a relative new feature in devices like PLCs, we believe that future releases of the standards will address this specific topic with more

[*] andrea.germinario@cern.ch

details.

This paper proposes a method to automatically generate ready-to-use Safety code, written in safety Ladder Diagram (LD) language, compliant with the LVL requirements of the IEC 61511. The LD programs are generated from Cause-Effect-Matrices (CEM), a specification method also recommended by IEC61511.

The paper is structured as follows: Section *Requirements for SIS AP design and development* briefly introduces some concepts related to Functional Safety, Model Based Design and discusses the recommendations of the Functional Safety standards about model based design and code generation. Section *From CEM to safety LD programs for Siemens TIA Portal* introduces the theoretical background of CEM and mentions the full workflow used to generate LD code from this formalism. Section *CEM-to-LD code generator* gives more insights about the design of the tool developed to generate LD and its implementation. Section *CERN case study* shows an case study to potentially use this tool in future upgrades of the project. Finally, Section *Conclusions and Future developments* outlines the conclusions of this work and some future developments.

REQUIREMENTS FOR SIS AP DESIGN AND DEVELOPMENT

Functional safety standards provide the guidelines to develop a SIS that reduces the risks to tolerable levels. The SIS comprises one or several Safety Instrumented Functions (SIF) where each one targets a specific risk. The requried risk reduction for each risk is determined as part as the risk analysis and assessment. This is directly related to the Safety Integrity Level (SIL), as shown in Table 1. SIL indicates how critical the risk is and sets the requirements for each SIF. This includes the requirements for the selection of hardware devices, which relates to the calculation of PFD_{avg} (average Probability of Failure on Demand) or PFH_{avg} (Probability of Failure per Hour), the SIF architecture, the testing activities and also the SIS AP.

Table 1: Relationship Between SIL, PFD, PFH and RRF

SIL	$\mathbf{PFD_{avg}}$	$\mathbf{PFH_{avg}}$	RRF
4	$\geq 10^{-5}$ to $< 10^{-4}$	$\geq 10^{-9}$ to $< 10^{-8}$	10000 to 100000
3	$\geq 10^{-4}$ to $< 10^{-3}$	$\geq 10^{-8}$ to $< 10^{-7}$	1000 to 10000
2	$\geq 10^{-3}$ to $< 10^{-2}$	$\geq 10^{-7}$ to $< 10^{-6}$	100 to 1000
1	$\geq 10^{-2}$ to $< 10^{-1}$	$\geq 10^{-6}$ to $< 10^{-5}$	10 to 100

In this section of this paper, we will present some of the most relevant requirements that are related to automatically generate the SIS AP.

The IEC 61511-1 Clause 12 defines the requirements for the SIS AP development. For example, it enforces the usage of LVL or Fixed Program Languages (FPL) for the SIS AP, excluding the usage of Full Variability Language (FVL). If FVL is required from the SIS AP functionality, the programmer must refer to IEC 61508-3 to extract the relative requirements.

An important aspect of the SIS AP is that it shall be consistent with and traceable back to the SRS (Safety Requirements Specification) from the phase 3 of the safety life-cycle. The AP should be modular and keep the complexity of each SIF to the minimum. The functionality of the SIS AP should be testable and there should be a one to one correspondence between the hardware architecture and the AP architecture.

The IEC 61511-2 Annex B (Example of SIS logic solver AP development using function block diagram) states that the traditional text based approach of safety AP specification is not efficient enough to handle the advanced, complex safety requirements commonly found in SIF specifications. The most efficient tool to address these challenges is Model-based design (MBD).

The IEC 61511-2 Annex D (Example of how to get from a piping and instrumentation diagram (P&ID) to application program) shows an example where the Cause and Effect (CEM) formalism was used to specify a SIF.

The IEC 61511 does not mention explicitly the use of code generation tools for the SIS AP. However the IEC 61508-3 Clause 7.4.4 (Requirements for support tools, including programming languages), states that support tools like code generation tools can be integrated into the safety program development process, provided they enhance software integrity by diminishing the chances of introducing faults or failing to detect them during the development phase.

In conclusion, PLC brands now allow automatic generation of safety PLC programs. In addition, the functional safety standards recommend the usage a MBD for the design of SIFs. They recommend to produce modular and traceable software and accept the use of code generation tools if they help to increase the reliability of the final SIS AP.

For these reasons, we believe that the future of SIS AP will be linked to code generation tools that are able to generate safety PLC programs from models like CEM, state machines or logic diagrams. This is already the case in safety critical industries like the aircraft industry [9]. These models can be used in simulation and validated. Once the desired behaviour is met, the SIS AP can be automatically generated from the model and imported as safety PLC program.

FROM CEM TO SAFETY LD PROGRAMS FOR SIEMENS TIA PORTAL

This section introduces the basic theoretical background on the methods and programming languages that have been used in this project.

Cause-Effect-Matrix Model

A Cause-Effect-Matrix is a formalism used to represent unambiguously relations between causes and effects in a system. It is used in stateless systems, where an output only depends on some combination of the inputs at a given time. In SIFs, normally what happens is that an actuator or an alarm triggers when a logic combination of sensors also triggers. In CEM, the rows represent the causes, which are normally the sensors, the column represents the effects which are normally the actuators and the cells of the matrix represent the relations between the two. An extensive explanation of CEM is present in [10], but here we only present the basic semantic of the logic expressions inside the cells.

- **X:** the cause, when active, triggers the effect (OR logic);
- **N:** the cause, when inactive, triggers the effect (OR NOT logic);
- **(N)A$_i$:** the effect is triggered when all the causes with the A$_i$ entry (where i = 1, 2, ...) are simultaneously active, or inactive if the prefix N is present (AND logic);
- **TON$_x$:** the cause, if active for more than x seconds, triggers effect (IEC61131-3 [6] TON logic);
- **TOF$_x$:** the cause, when active, triggers the effect and the effect remains active for x seconds after the cause becomes inactive (IEC61131-3 [6] TOF logic);
- Multiple entries in a single column, or separated by "," in a single cell, are combined with OR logic;
- The same effect may appear in multiple matrices, the resulting expression for this effect is an OR logic between the activations of each matrix.

As an example, Table 2 represents a CEM whose first column encodes the logic requirements implemented in the Ladder Diagram in Fig. 1.

Table 2: CEM Example

Effect	Q01	Q02
Cause		
I01	X	
I02	TON(20)	A1, A2
I03	NA1	A1
I04	A1	NA2

Ladder Programming

Ladder Diagrams (LD) is a graphical programming language for PLCs that resembles circuit diagrams used to design relay logic hardware. Its inputs resemble electric contacts and its outputs electric coils. It is particularly efficient for coding stateless logic, mainly containing AND, OR and negations. All details about LD can be found in IEC61131-3 [6]. Safety LD is standard LD with several restrictions on the datatypes and the variable operations to meet the functional safety standards requirements for LVL. For example, *REAL* datatypes (floating point number variables in PLC programming) cannot be use in Safety PLC programs.

Using LD to represent the logic expressed in the first column of Table 2, the result is shown in Fig. 1.

The one-to-one unambiguous relation between the first column of the CEM and the LD makes very clear and simple the translation rules. Figure 2 shows a simple example of translation from a CEM to an LD program in TIA portal. The LD code consists on a *FUNCTION_BLOCK* (*FB_M1*) and its *instance DATA_BLOCK* (*FB_M1_DB*).

The difference between this example and a real-case scenario is not in the complexity of the logic, but in the size and quantity of CEM(s) to specify and consequently in the amount of LD to code. In a real control or safety system, the LD programs can be very large and an engineer should carefully read the specifications in the CEMs and slowly translate them to LD, with high likelihood of introducing coding mistake during this manual process. A well-tested code generation tool would solve this problem.

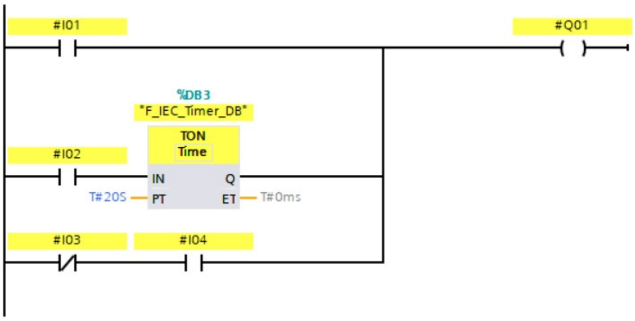
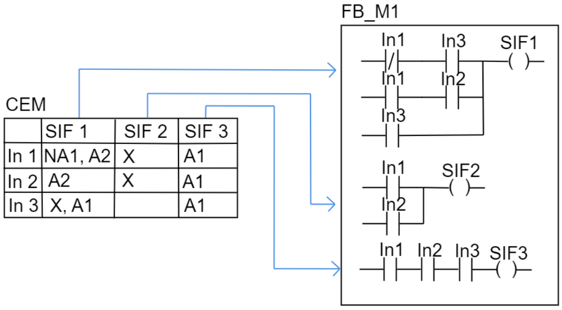


Figure 1: Ladder Diagram specified by the first column of the CEM shown in Table 2 represented in TIA Portal V17.



CEM

	SIF 1	SIF 2	SIF 3
In 1	NA1, A2	X	A1
In 2	A2	X	A1
In 3	X, A1		A1

Figure 2: Example of CEM to LD one-to-one bi-univocal relation.

Full Workflow

At CERN, we have developed a tool to specify interlock logics based on CEMs. The tool is called *SISpec* [10] and it is still in the prototype phase. Using this tool, we can write CEMs reducing the specification errors, thanks to syntax and specification checks. Once the CEM is ready and validated, it can be exported into *XML* format. This file can be imported into a *Python* tool called *CEM-to-LD*, which will extract the information from the CEM into an intermediate model to then translate it into LD for Siemens PLCs. The output of the tool is a file in XML format that represents the LD program in a textual format. This file can be imported and compiled in TIA Portal (V16 or later), and then downloaded to a Siemens PLC, using the Siemens tool called OpenessScripter. All this complexity is hidden to the engineer, since once the specification file is created with SISpec, the rest of the process can be fully automatized. The full workflow is shown in Fig. 3.

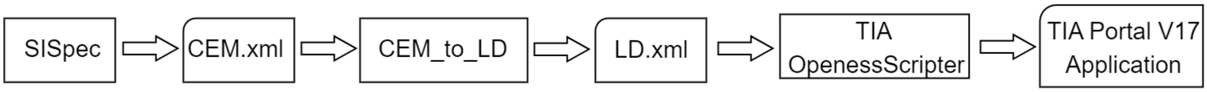


Figure 3: Full workflow from specification to generated code.

CEM-TO-LD CODE GENERATOR

The *CEM-to-LD* tool is fully written using Python 3.9 [11] only using standard library tools. It is designed as a Python package and can be run by command line on any computer with Python 3.9 installed.

The idea behind this tool is to create a generic platform where more specification languages and PLC programming languages can be added. Starting from the program specification, instead of translating it directly into LD, an intermediate model is generated. Then the intermediate model is translated into LD. In this scalable approach, we can add new specification methods that will be translated to the intermediate model and new PLC programming languages that can be generated from the intermediate model. By employing an Intermediate Model, it is possible to generate Function Block Diagram (FBD) programs as well, only the translation from IM to FBD is required. Or even generating LD or FBD programs for other PLC brands, since the textual representation is different from the TIA portal one. Same applies for new specification methods. The IM can be obtained from other types of specification methodologies (e.g. state machines or logic diagrams). Therefore the tool can be enlarged to accept new specification methods. A representation of this idea is shown in Fig. 4.

This is our vision for the future of the tool. Multiple specification methods and multiple programming languages in a more generic tool: *CEM-to-TIA*. This enforces the architecture of the tool based on this intermediate model. The advantage in terms of scalability of the Intermediate Model is clearer with an example. Considering a case where 3 specification methods and 3 output PLC languages are accepted by the tool, the number of translations to implement increases from 6 to 9 when not using the IM.

Users can be concerned about the introduction of errors in the generated code. Even the standards states clearly that a code generation tool should be added to the SIS development workflow if it is proven that it can increase the reliability of the final program. In safety critical industries like avionics or aerospace, the use of certified code generation tools is already common. An example is the ANSYS SCADE Suite [1] code generator. However, obtaining certification for our tool is an extensive and expensive endeavor and, currently, there are not plans to pursue it in the near future. Our approach is shown in Fig. 5, where in addition to generating the LD from CEM, also formal verification properties and test cases are automatically generated from it. This allows us to formally verify the generated PLC program against formal properties with PLCverif[2] [12, 13] and run the test cases to guarantee that the safety PLC program is compliant with the logic expressed in the CEM. This minimizes the possibility of the code generator introducing bugs.

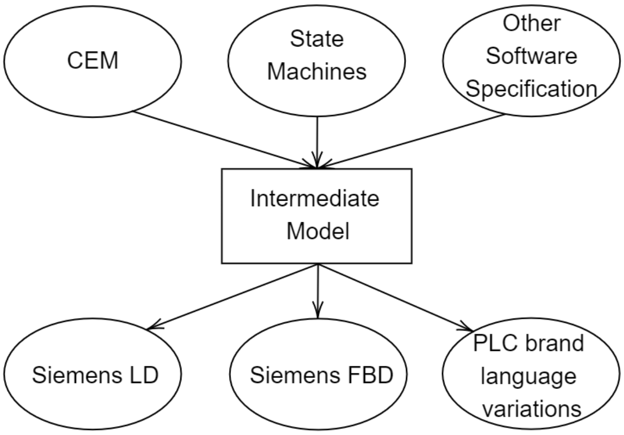


Figure 4: Conceptual representation of role of the Intermediate Model in the CEM_to_LD tool.

CERN CASE STUDY

In its current status, *CEM-to-TIA* can be employed in any scenario that utilizes CEM for program specification and LD programs in TIA Portal V17 as programming environment for PLCs. The tool was tested on one of the safety control systems that protects the SM18 cluster F superconducting magnet test bench facility at CERN. The PLC program for this project was specified using CEMs. This specification contains 350+ variables and 19 matrices. By applying this tool, the safety PLC program was generated. 19 *FUNCTIONS_BLOCKS* and *DATA_BLOCKS* were automatically generated. The generated PLC program will be deployed in a future upgrade of the control and safety of the SM18 facility.

The current safety PLC program of the SM18 facility was written manually by the PLC programmer for the former programming environment for Siemens PLCs: SIMATIC Step7. This was an enormous time consuming task. However, future upgrades will leverage the *CEM-to-TIA* tool to expedite the PLC program development process and reduce the occurrence of coding errors.

In addition, a large set of generated LD programs where validated to test the tool and validate the correctness of the translation.

[1] ANSYS SCADE Suite webpage `https://www.ansys.com/products/embedded-software/ansys-scade-suite`.

[2] PLCverif is available on `https://gitlab.com/plcverif-oss`.

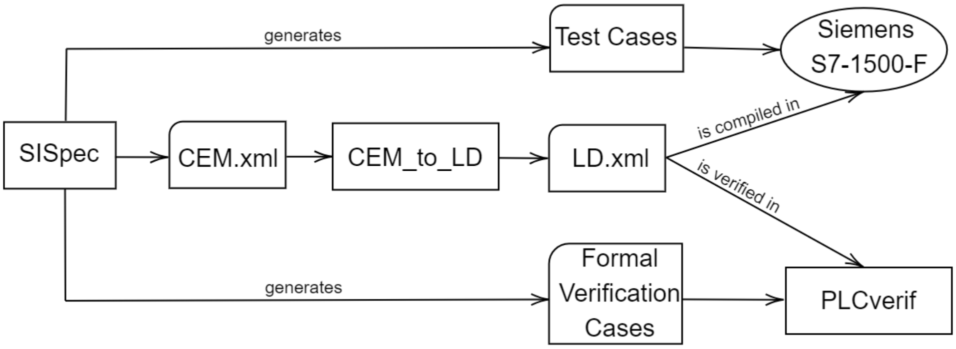


Figure 5: Workflow to prove the compliance between the generated LD and the CEM.

CONCLUSIONS AND FUTURE DEVELOPMENTS

This paper presents a novel approach to generate Safety Ladder Diagram programs for Siemens TIA portal projects, using Cause-Effect-Matrices to define the program specification. The underlying logic behind this concept is that, in safety-critical applications, the specifications provide a comprehensive and unambiguous description of the software to be created. Therefore, with the specification in hand, it becomes possible to generate the code instead of having to write it manually. This method dramatically reduces the development time and the amount of errors, yet is compliant with the IEC 61508 Standard.

The tool follows the guidelines of the Functional Safety standards regarding the rules for LVL programs, Model Based Design and usage of external tools for code generation.

The paper presents the background of Cause-Effect-Matrices, Ladder Diagrams and the insides of the *CEM-to-LAD* tool based on an intermediate model. The tool was applied to a real CERN case study: the SM18 superconducting magnet test facility at CERN. The results were positive, confirming the potential of code generation in safety applications.

The future development of the tool should mainly focus on enlarging the input specification methods and output PLC programming languages. The tool will be applied to new safety projects at CERN and can also be applied in other organizations and companies designing safety PLC programs.

REFERENCES

[1] Siemens webpage - Mean Time Between Failures (MTBF) - list for SIMATIC products, https://support.industry.siemens.com/cs/document/16818490/mean-time-between-failures-(mtbf)-list-for-simatic-products?dti=0&lc=en-WW

[2] TÜV SÜD webpage, https://www.tuvsud.com/en

[3] Functional safety of electrical/electronic/programmable electronic safety-related systems, https://webstore.iec.ch/publication/5515

[4] Functional safety - safety instrumented systems for the process industry sector, https://webstore.iec.ch/publication/24241

[5] MathWorks PLC code generation from simulink models, https://www.mathworks.com/help/plccoder/code-generation-for-plcs.html

[6] Programmable controllers - part 3: Programming languages, https://webstore.iec.ch/publication/4552

[7] UNICOS webpagef, https://unicos.web.cern.ch/

[8] Siemens TIA Portal webpage, https://www.siemens.com/global/en/products/automation/industry-software/automation-software/tia-portal.html

[9] G. Walde and R. Luckner, "Bridging the tool gap for model-based design from flight control function design in simulink to software design in SCADE", in *2016 IEEE/AIAA 35th Digit. Avion. Syst. Conf. (DASC)*, Sacramento, CA, USA, 2016, pp. 1–10. doi:10.1109/DASC.2016.7778044

[10] B. F. Adiego *et al.*, "Cause-and-Effect Matrix Specifications for Safety Critical Systems at CERN", in *Proc. ICALEPCS'19*, New York, NY, USA, 2020, pp. 285–290. doi:10.18429/JACoW-ICALEPCS2019-MOPHA041

[11] Python webpage, https://www.python.org/

[12] J.-C. Tournier, B. F. Adiego, and I. Lopez-Miguel, "PLCverif: Status of a Formal Verification Tool for Programmable Logic Controller", in *Proc. ICALEPCS'21*, Shanghai, China, 2022, paper MOPV042, pp. 248–252. doi:10.18429/JACoW-ICALEPCS2021-MOPV042

[13] D. Darvas, E. B. Viñuela, and V. Molnár, "PLCverif Re-engineered: An Open Platform for the Formal Analysis of PLC Programs", in *Proc. ICALEPCS'19*, New York, NY, USA, 2020, pp. 21–27. doi:10.18429/JACoW-ICALEPCS2019-MOBPP01