7th Annual IEEE Conference on Automation Science and Engineering (CASE 2011), August 24-27, 2011, Trieste, Italy.

Towards an Automated Verification Process for Industrial Safety Applications

Kleanthis Thramboulidis[2,1], Doaa Soliman[1], and Georg Frey[1]

[1]*Saarland University, Saarbrücken, Germany,* [2]*University of Patras, Greece*

***Abstract*— Legacy systems that do not conform to the norms and regulations imposed by recent safety standards have to be upgraded to meet safety requirements. In this paper, we describe a methodology to upgrade legacy industrial applications based on the IEC61131 function block model without the need to redesign the whole application. We then describe an approach for automating the verification process of safety applications that is based on the use of the UPPAAL simulation and verification platform for embedded real-time systems. The meta-models of the source and target domains are presented and a transformation process of the PLCopen XML design specification to UPPAAL XML specification is described. A laboratory system is used as a case study to demonstrate the applicability of the proposed process.**

I. Introduction

International standard organizations, such as ISO and IEC have defined various standards on safety due to social and environmental demands. These standards, as for example the IEC61508 [1] and its implementation for the machinery sector, i.e., the IEC62061 [2], impose manufacturing industries to certify that their systems are safe for the human life and the environment. This means that they have to upgrade legacy industrial automation systems to conform to various norms and regulations. The alternative to throw away the legacy system and develop a new one from scratch to meet the requirements specification, if such a specification exists, is very expensive mainly due to the high value of the software part of these systems.

Among the challenges that the engineer faces for upgrading legacy systems are: the definition of safety requirements for the upgraded system, the definition of the requirements of the safety system, the design of the safety system, the verification of the safety application, its integration with the legacy system and the verification of the upgraded system. This makes the upgrade process significantly different from the development of a new system where an integration of the traditional development process with safety engineering is also required [3].

A methodology to systematically address the above challenges and upgrade legacy industrial systems to meet safety regulations is briefly presented in this paper. The proposed methodology has been applied to a laboratory system to demonstrate its feasibility. During this work it was found that one of the most challenging parts of this process is the verification of the safety application. There are several works to this direction and it seems that there is an increasing interest for automating the verification process of FBD based safety applications. However, all the published verification approaches are based on proprietary tools and notations and none of them can be generalized since no common intermediate platform is supported.

In this paper we describe a framework towards an automatic verification process for safety applications developed using the FBD language of the IEC61131-3 [4]. More specifically, we describe an approach to upgrade legacy systems to conform to safety regulations defined by safety standards such as IEC61508 and IEC61511 [5] and we focus on the automatic transformation of the safety application design models, which are expressed in PLCopen XML, to UPPAAL models to semi-automate the verification process. The meta-models of the source and target domains have been defined as well as the required mapping rules from one domain to the other. XML is used as a notation for expressing both domain models, since it is a flexible standardized format for the description of PLC programs [6]. The properties of the safety application are verified using the transformed model. Depending on model checking results, the design model of the safety application is iteratively modified for the system to meet all safety requirements.

The remainder of this paper is organized as follows: Section 2 discusses related work. In Section 3, the proposed approach for upgrading industrial applications to conform to safety standards is presented. Section 4 presents the verification process of the safety application. The meta-models of source and target domains are presented and the transformation rules are given. The paper is concluded in the last section.

II. Related Work

Nowadays, there is an increasing interest of automating the verification process of applications in industrial automation. Several research groups have already published the results of their work to this direction. These works are mainly based on the transformation of the FBD models of the safety application to formal models, which are next used for verification purposes with the help of a model checking tool. For example, authors in [7], present a method for the automated formal verification of PLC software written in

FBD using the NUSMV model checker. The FBD should be constructed from basic FBs such as bit operations, comparators and jumps. Moreover, the IL representation of the FB body which is generated from the corresponding FBD with the help of the PLC programming tool is the one to be transformed to a NUSMV model. This method cannot be applied for the verification of safety applications built up from PLCopen SFBs, since SFBs are not basic FBs.

Authors in [8] propose a way to formally verify FBD programs. They adopt an IEC 61131-3 compliant syntax for the FBD and propose the translation of this specification into Verilog models. The so generated Verilog models are verified using the Cadence SMV model checker. A tool, FBD2V, has been implemented to automatically perform this transformation, and a case study is used to show the effectiveness of the proposed approach. As in [7], the transformation approach is presented only for systems composed finally of basic FBs, so safety FBs of PLCopen are not supported.

A formal verification approach of a safety procedure of a nuclear power plant written in FBD language is presented in [9]. The approach is based on the Colored Petri Net (CPN) representation and it is similar to our approach in that it uses a pre-developed library of CPN subnets to represent the FBD specification in an identical structural way. However, the transformation of FBD diagrams to CPN is carried out manually. An automatic transformation process is not supported.

In [10], authors propose and describe an automatic generation of timed automata (TA) models from FBD models. This allows testing of FBDs against their specifications in the case that there is no PLC available to run them. The UPPAAL TRON tool is used to test some simulation scenarios on the generated TA models. The FBD elements that this approach deals with, are logical AND, OR, SR/RS flip-flops and time elements TON, TOFF and TP. The test of the model is evaluated by feeding some simulation traces to UPPAAL TRON tool. Traces are automatically generated from cause/effect matrix specified by experts. This approach can be considered similar to the one presented in this paper, but it is very different if we consider the details. Moreover, this approach cannot be applied to the safety application used as a case study in this paper, since it only supports logical and time FBs.

We are not aware of any other approach that automatically transforms IEC61131 FBD specifications to formal models for their automatic verification, as part of a systematic approach for upgrading legacy industrial systems to meet safety regulations defined by standards.

III. The Proposed Approach

This section provides a brief description of the proposed methodology for upgrading legacy industrial systems to meet safety regulations. The XY coordinated table of the Automation laboratory at Saarland University, shown in Fig. 1 is used as a case study in this paper. This type of precision-controlled automated movement is broadly used in mechanical processes and applications including material handling and pharmaceutical. The system consists of an automatically moved XY drawing table with manually adjustable marking pen mounted above an Aluminum base. The two axes are operated with Siemens synchronous motors and precision linear guides from Bosch Rexroth. A soft-PLC WinLC-T from Siemens equipped with S7-technology is used and the motors are controlled using Sinamics S120 servo drives. The user interface is based on Wincc HMI that allows the operator to perform the required motion tasks through simple graphical screens. The control system equipments, i.e., PLC and Servo drivers, are connected through Profibus DP. For safety reasons the system is protected by a glassy cell with two doors on the top.

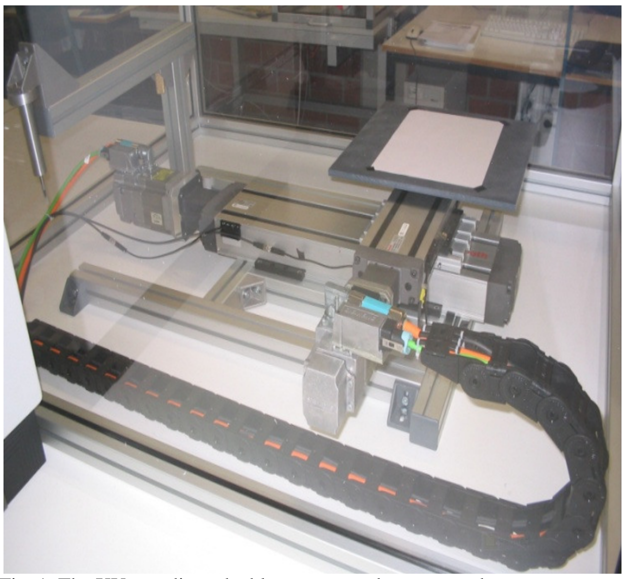
Fig. 1. The XY coordinated table system used as case study.

The main steps of the methodology we adopt in this paper are: 1) Definition of safety requirements for the upgraded system; 2) Definition of the requirements of the safety system; 3) Design of the safety system; 4) Verification of the safety application; 5) Integration with the legacy system; and 6) Verification of the upgraded system. A detailed description of the methodology, part of which is the proposed in this paper verification process, can be found in [11]. Our approach is consistent with the high level one described in IEC62061 and it proceeds further in refining the defined by that standard processes. We start by creating a list of abstract safety requirements for the upgraded system in informal text notation. These requirements will be used in step 6 to verify the system. This list contains requirements imposed by standards, such as those mentioned by IEC62061, that include among others the following:

R.1. provide complementary protective measures, e.g. emergency stop;
R.2. provide information for safe use, e.g. warning signals;
R.3. reduce risk under conditions of failure or misuse e.g. automatic monitoring, overload protection.

System specific requirements defined by the developer are also included in this list. The following requirements are an example of this category:

R.3. Cell's doors should be released for opening under normal conditions only when the machine is not in the "perform task" mode;
R.4. Cell's doors should be locked before the system enters the "perform task" mode;
R.5. Motor's speed should not exceed a given safe speed when in setup mode.

Non-functional requirements such as SILs of the system functionalities, often considered as a measurement of performance required by the expected safety functionalities, should also be defined in this step. The SysML requirements diagram can be used to capture these requirements and benefit of the requirements handling support that is provided by this system engineering language. Requirements imposed by IEC61511-1 for the requirements specification process are satisfied by SysML. A specific SysML profile for safety will help the engineer to capture and handle all the safety related model elements, and also relate them with the other elements of the system development process.

The objective of the second step, that uses as inputs the System safety requirements and the legacy system essential use cases, is to define the functionality that is expected from the safety system. This is what we call functional requirements for the development of the safety system. Even though we are dealing with a legacy system, we adopt the approach of Preliminary Hazard Analysis (PHA) on the use case level as defined in [3], since it is more effective to perform first a safety analysis on an abstract level of system description, i.e., the one that captures the system as a black box and emphasizes its interactions with its actors. Fig. 2 presents the basic sub-steps of this step. Hazards that emanate from the actor-system interaction and are caused by actors and/or system misbehavior are identified. The legacy system use cases and system safety requirements constitute the input activity parameters for this PHA, while the upgraded system use cases are the output. Hazard analysis (HA), risk analysis and safety measure definition are the main sub-processes of the PHA. To calculate the risk level of every defined hazard, risk analysis is applied. The unacceptable hazards have to be mitigated by defining safety measures. HA is complemented by solution dependent and component hazard analyses. Based on the results of HA, the system use cases are upgraded to meet safety requirements.

The so constructed upgraded use cases constitute the inputs of the refine and split sub-process of this step. Sequence diagrams are an excellent tool to refine use cases. The captured in this way system behavior is distributed to the legacy system and the target safety one using again sequence diagrams. These diagrams are next used to identify and extract the safety system's requirements, following the MTS-V model [12]. This results to functional and non-functional requirements for the safety system in the form of SysML requirements diagrams. Functional requirements will be refined using use cases. SIL which is considered as a non-functional requirement should be defined for every safety related control function. It should be clear that by the term safety system we mean not only the safety application (that is the software) but also the required hardware, i.e., emergency buttons, guard switches, PLC, etc.

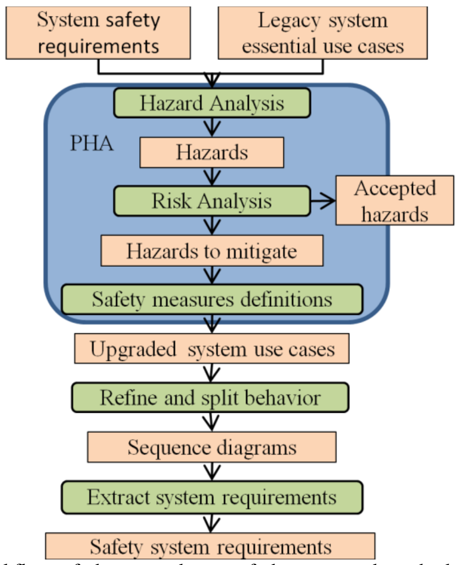
System safety requirements
Legacy system essential use cases
Hazard Analysis
PHA
Hazards
Risk Analysis
Accepted hazards
Hazards to mitigate
Safety measures definitions
Upgraded system use cases
Refine and split behavior
Sequence diagrams
Extract system requirements
Safety system requirements

Fig. 2. Workflow of the second step of the proposed methodology for upgrading a legacy system.

An example sequence diagram that was used to refine and split the functionality of the upgraded use case among the legacy and the safety system of our use case system, is shown in Fig. 3 which refers only to the automatic mode of the system operation. The other two modes of operation are initialization and set up mode. Three actors, i.e., the operator, the legacy system and the mechanical process, are captured in. Fig. 3.

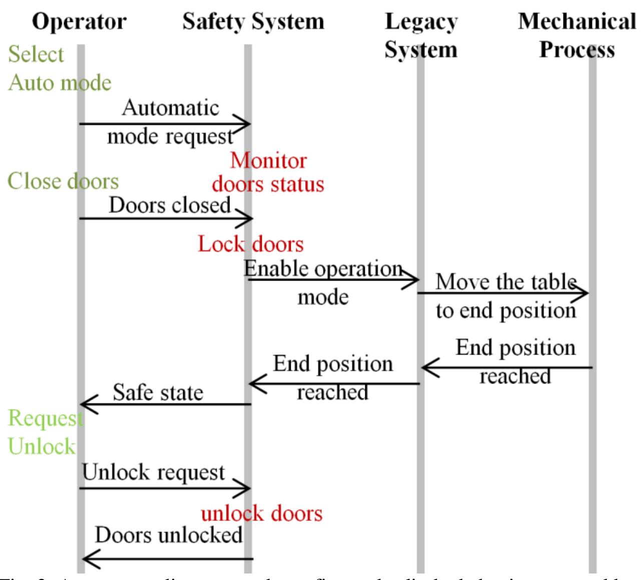

Fig. 3. A sequence diagram used to refine and split the behavior captured by an upgraded use case (automatic mode).

All the functionality of initialization mode is assigned to the legacy system which is responsible for handling initialization requests, initiated by the operator, who is responsible for selecting the task to be executed, defining the home position of the drawing table and the operating speed for both motors. For safety reasons, in setup mode, the motors speed should not exceed a defined safely limited speed (SLS), while the drawing table moves to the home position. As far as an acknowledgement of SLS is accepted by the safety system, a safe state is assumed and the operator is able to access the mechanical part safely. If the acknowledge signal is lost while in setup mode, the safety system assumes an unsafe state and safety measures are executed. During the automatic mode that is the main operation mode of the system, the selected task by the user is executed. This operation mode can be executed only when the system is in a safe state, that means that the doors must be closed and locked to start operation. Any unlock request is ignored while being in this mode of operation.

It is the responsibility of the safety system to pass emergency stop requests to the legacy system, execute safety measures and assume an unsafe state until a safe stop is received. When the legacy system, which controls the motors speed, accepts an emergency stop, it drives the motors to zero speed according to the defined stop category.

The objective of the third step of the proposed approach is to design a safety system to meet the functional and safety integrity requirements specified in the safety requirements specification. The proposed architecture for the safety system of our case study satisfies the requirements set on software architecture definition by 61511-1. SysML is used to express the system design which may next be automatically transformed through specific model-to-model transformers to IEC61131 FBD models [13]. However, in this case study we have skipped the SysML design and we have directly drawn the FBD design model. Before moving to the implementation of the proposed design we have to verify that the proposed design meets the requirements. The next section describes an approach to automate the verification process, i.e., step 4.

IV. Verification Process

The so created design model of the safety application has to be verified before implementing it. UPPAAL was selected to be used for the verification process of the safety application. This means that the FBD design models of the safety application have to be transformed to UPPAAL models. In this section a generic approach to semi automate this verification process is proposed. PLCopen XML was used as standard source format to allow the approach to be used by any IEC61131 compliant development tool that may supports the PLCopen XML specification [6]. UPPAAL [14] is a good choice for formal verification of systems that can be modeled as a collection of non-deterministic process with real valued clocks. That makes it suitable for the verification of FBD built from FBs triggered by timers. The presented approach utilizes the UPPAAL model checker that supports modeling (via graphical editor), validation (via graphical simulation) and verification (via verifier) of real time systems. The transformation process is described, source and target domain meta-models are given and the transformation rules are presented.

A. *Transformation process description*

The proposed transformation process may be used with any PLC development tool that is complaint with the IEC61131-3 programming standard, it provides support for the certified PLCopen SFB library, and may export the design models in PLCopen XML specification. The basic idea of the transformation process is depicted in Fig 4. Safety FBs of the PLCopen safety library have been transformed to corresponding UPPALL TA forming the safety UPPAAL library. For every SFB, a TA model was graphically constructed via the UPPAAL editor, validated via simulator and finally verified against formal properties via the verifier [15]. This work simplifies the building of the required model-to-model transformer that will use as source information the design model of the safety application, which will be exported in PLCopen XML format, to automatically generate the equivalent UPPAAL model in XML format. The result of the transformation process is imported to the UPPAAL model checker for verification against the safety requirements of the safety application. Information on the unsatisfied properties is forwarded to the simulator to identify the TA state in which the property is not satisfied. This helps in finding the source of the problem and addressing it.

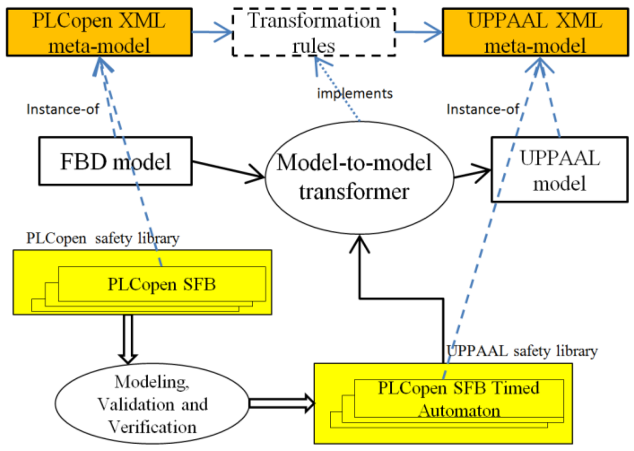

Fig. 4. The proposed transformation process.

The PLCopen XML source file captures except from the structure of the safety application part of the behavior that is defined by the specific collaboration scheme and the priorities of the application's constituent components. It also captures the connections of the application to the environment. The execution flow is not directly captured but this is extracted from the configuration of the application. The remaining part of the behavior is directly inserted in the UPPAAL XML file from the corresponding UPPAAL safety library.

The transformation process was successfully performed on the real safety application of our case study using

MultiProg from KW software. For the development of the safety application the PLCopen Safety Function Blocks library [15] was used.

B. Source and target domain meta-models

The meta-models of the source and target were constructed to automate the transformation process. Fig. 5 presents part of the PLCopen XML meta-model that was constructed to capture all the model elements that are used for constructing safety applications.

The *project* element captures project specific information and a set of types, where each type is composed of a pous element that is an aggregation of 1 to many pou. A pou is the IEC61131-3 program organization unit which is composed of an interface and a body. For safety applications only FB instances can be used as building blocks. These FB instances may be SFBs or logic gate FBs. For every FB, the list of input variables is defined under the inputVariables element. Every variable has a predefined formalParameter, specified by the vendor, and a connectionPointIn. If this variable is connected, then the reference local Id (refLocalId) number of the other connected variable is defined. The refLocalId may be for other FB output variable or a variable defined in inVariable/outVariable elements of the FBD. Local and external variables declarations are found under the localVars and externalVars elements. For the list of FB output variables defined in outputVariables element, only a formalParameter name is associated. To define connections, a local Id number is given to every FB instance, input variable and output variable. An input variable (inVariable) may be expressed as an external input variable, a local variable or a constant. An output variable (outVariable) may be expressed as an external output variable or a local variable. FB instances are defined in the localVars element, which contains the variable name defined by user and used as name of the FB instance and a derived name that is vendor predefined information.

Since the UPPAAL XML consists simply of TA templates which communicate via channels or variables defined in the global declaration part, the structure of the meta-model is straight forward. The UPPAAL XML specification is split into three main elements, which are declaration, template and system. In the global declaration part, all external variables and input/output variables of SFBs are declared. In our case there is no need of channel definitions since the communication between TAs is realized via shared variables. A template consists of locations and transitions between them. It has a name, local declaration part and an initial location. For every location a name and a label are associated. The label may be an invariant and/or comment. A transition has a source and target locations and defined nail positions. Labels of kind assignment, guard and channel can be defined on transitions. In the system part, TA template names are defined according to their execution priority.

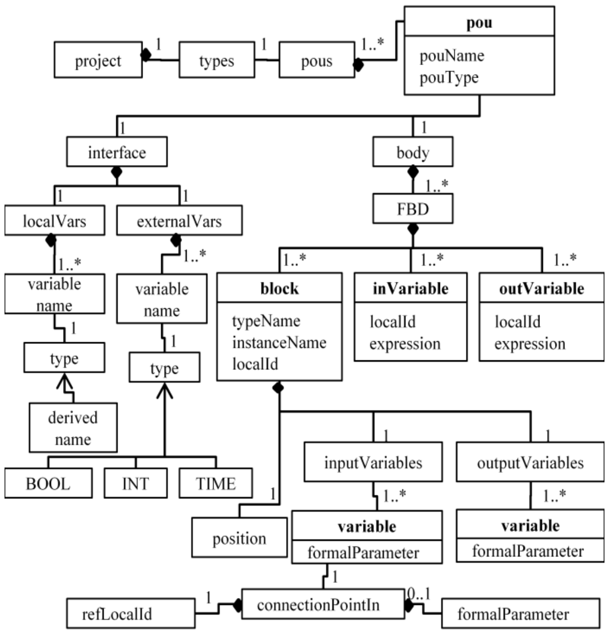

Fig. 5. The PLCopen XML meta-model used in the context of this work (part).

C. The transformation rules

The information that has to be extracted from the PLCopen XML file and mapped to the UPPAAL XML file is the used SFBs, and their interconnections, the external input/output variables, and information on the execution flow. Based on the following four rules have been defined:

Declaration Rule: The objective of this rule is to map variable declaration statements. Variables defined in the externalVars element and the input/output variables of each safety FB defined in the localVars element of PLCopen XML are declared as global in UPPAAL XML.

SFB Rule: The objective of this rule is to map the SFBs. For every safety FB defined in the localVars element of PLCopen XML, the corresponding TA template with the same name is inserted to the UPPAAL XML file.

Connections Rule: This rule maps the connection of FBs. For every FB input variable in the inputVariables element, the refLocalId element associated to a connection TA template is recognized and the corresponding connection is constructed in UPPAAL XML.

Execution Flow Rule: The objective of this rule is to define the execution flow of the UPPAAL model. The execution flow numbers of FB instances are extracted by their XY position in the FBD. Based on this, the execution priority of every TA is defined in the UPPAAL model.

Fig. 6 presents part of the FBD model that realizes the sequence diagram given in Fig 3. This part of the model monitors the status of doors and responds to an unlock request. Direct connections, as well as indirect connections via logic gates and direct connections to external input and output variables are also shown. Fig. 7 presents the corresponding UPPAAL TA. It captures one safety FB and its connection TAs. OC1 implements the direct connection to the external variable S_UnlockGuard_k1, C3 implements the direct connection between FBs and ORC1 implements the indirect connection via the OR gate. The system of TAs is arranged so as ORC1 is assigned the highest priority based on the IEC61131-3 notation regarding execution flow.

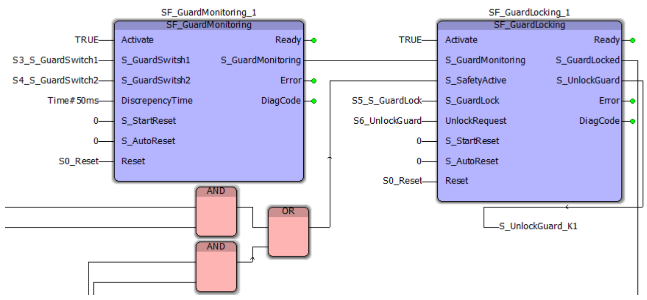

Fig. 6. Safety application FBD model (part).

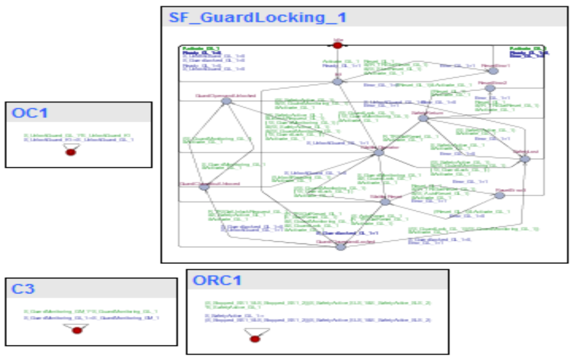

Fig. 7. Corresponding TA (part).

Appling validation via simulation on the UPPAAL model it is assured that this model is identical to the designed safety application. After that, model checking was used for verification. It was found that the property “Under normal conditions, when Automatic mode is selected and doors are closed and locked, then motors start performing task”, which is expressed in Computation Tree Logic (CTL) as “A[] S_SafetyActive_GL_1 & S_Mode0_MS_1 & S_GuardSwitch1_GM_1 & S_GuardSwitch2_GM_1 & S_GuardLock_GL_1 imply S_GuardLocked_GL_1”, is not satisfied. The source of the problem was identified to be the SF_GuardMonitoring_1 safety FB. The safety application was modified to satisfy the property.

V. Conclusion

A framework to automate the verification process of IEC61131 based industrial safety applications has been presented in this paper. This is mainly based on a methodology to upgrade legacy industrial systems to be compliant with rules and regulations imposed by safety standards. PHA is applied using a high level description of the system, the one expressed by essential use cases and a very abstract safety requirements specification, to identify hazards, their causes and their risk, and the corresponding safety measures to mitigate the risk of the unacceptable ones. Application of the solution depended and component hazard analysis complements the hazard analysis process and results to the definition of the requirements of the safety system. SysML is used to create the requirements model of the system and capture the safety analysis model elements. It is consequently used to express the system architecture too. The UPPAAL model checking tool was selected to be used for verification purposes. The meta-models of the source, i.e., PLCopen XML, and target, i.e., UPPAAL XML, domains have been defined and the transformation rules were constructed based on these meta-models. The proposed approach was successfully applied to a laboratory system to upgrade it to meet safety regulations. The safety system was designed, verified, implemented and integrated with the legacy system so as the upgraded one to meet the defined safety requirements. We are currently working: a) on the automation of the transformation of the safety application specification in PLCopen XML format into a readable format by the UPPAAL model checker tool, and b) on the model transformation from SysML to IEC 61131.

References

[1] International Electrotechnical Commission (IEC), IEC 61508 – Functional Safety of Electrical/Electronic/Programmable Electronic Safety-Related Systems, December 1998
[2] Inter. Electrotechnical Commission (IEC), IEC 62061 – Safety of machinery - Functional safety- Electrical, electronic and programmable electronic control systems, Committee draft, 29-09-2000
[3] K. Thramboulidis and Sven Scholz, “Integrating the 3+1 SysML view model with safety engineering,” IEEE International Conference on Emerging Technology and Factory Automation, Bilbao, Spain, 2010.
[4] K. John, and M. Tiegelkamp, “IEC61131-3: Programming Industrial Automation System: Concepts and Programming Languages, Requirements for Program,” Systems, Aids to Decision-Making Tools, 2001.
[5] International Electrotechnical Commission (IEC), IEC 61511 – Functional safety: Safety Instrumented Systems for the process industry sector, 2003/2004.
[6] M. Bani Younis and G. Frey, “Visualization of PLC Programs using XML,” Proceeding of the 2004 American Control Conference, Boston, Massachusetts, 2004.
[7] O. Pavlovic´ and H. Ehrich, “Model Checking PLC Software Written in Function Block Diagram,” International Conference on Software Testing, Verification and Validation, pp. 439-448, Paris, France, 2010.
[8] J. Yoo, S. Cha, E. Jee, "A Verification Framework for FBD Based Software in Nuclear Power Plants," 15th Asia-Pacific Software Engineering Conference, 2008, pp. 385-392.
[9] E. Németh and T. Bartha, “Formal verification of safety functions by reinterpretation of functional block based specifications,” Formal Methods for Industrial Critical Systems, LNCS 5596, 199-214, 2009.
[10] L. Silva, L. Barbosa, K. Gorgonio, A. Perkusich, A. Lima, “On the automatic generation of timed automata models from function block diagrams for safety instrumented systems,” 34th Annual Conf. of the IEEE Industrial Electronics Society, (IECON 2008), pp. 291-29, 2008.
[11] D. Soliman, K. Thramboulidis, G. Frey, “A Methodology to Upgrade Legacy Industrial Systems to Meet Safety regulations”, 3rd Inter. Workshop on dependable control of discrete systems, June 15-17, 2011, Saarbrucken, Germany.
[12] K. Thramboulidis, “The 3+1 SysML View-Model in Model Integrated Mechatronics”, Journal of Software Engineering and Applications (JSEA), vol.3, no.2, 2010, pp.109-118.
[13] K. Thramboulidis, and G. Frey, “Towards a Model-Driven IEC 61131-based Development Process in Industrial Automation,” Journal of Software Engineering and Applications (JSEA), Vol. 4, No. 4, April 2011.
[14] K.G. Larsen, P. Pettersson, and Y. Wang. Uppaal in a nutshell. Software Tools for Technology Transfer, 1(1-2):134-152, 1997.
[15] D. Soliman and G. Frey, “Verifications and validations of safety applications based on PLCopen safety function blocks,” Control Engineering Practice, (in press).