A Framework for Modeling and Formal Verification of SIS Control Programs Based on the IEC61511 Standard

Rodrigo Cesar Ferrarezi, Reinaldo Squillante Júnior, Jefferson A. L. Souza, José Reinaldo Silva, Fabrício Junqueira, Paulo Eigi Miyagi, Lucas Antonio Moscato, Jun Okamoto, Jr., Diolino J. Santos Filho

Abstract— As productive systems are becoming more complex, their control solutions are also increasingly becoming more complex. The processes of understanding and developing such systems have also become highly complex. Thus design flaws are intrinsic to their development. Even the most innovative systems are error prone and faults or accidents may cause severe damage to the operators, the plant or the environment, as no system has no fault risk. The concepts of Safety Instrumented Systems (SIS) might be a solution to this problem. However, critical systems – such as oil and gas refineries where faults may cause severe accidents – controlled by PLCs also demand a formal verification processes of their control programs and must be developed according to the safety control program development cycle defined on the IEC 61511 standard. On this work we propose a framework for the model based development of SIS control programs that is based on the cycle defined on the IEC 61511 standard and where the interaction between the prevention and mitigation programs is considered. The framework was applied to the development of the SIS control program of a natural gas compression plant (ECOMP). The framework allowed the resulting program to present a modular structure and to have several properties properly verified – considering that the final model represented the entire SIS program of a real world plant.

Keywords— framework, IEC 61511, prevention and mitigation SIS, Model Checking, GHENeSys.

I. INTRODUCTION

A

s per society demand, increasingly complex control systems are being developed for a great variety of applications, ranging from productive to service systems [1] [2]. Together with the physical complexity of these control systems, their control programs also became more complex, not only because some tasks, which were performed by hardware, are now performed by software, but also because software allows the implementation of much more complex tasks [3].

Even the most innovative systems are error prone and faults or accidents may cause severe damage to the operators, the plant or the environment. Although fault detection and treatment is subject of several studies, faults still happen as (i) no physical component or device has no fault risk, (ii) human operators does not have no risk, (iii) no tool can predict or model all states reachable by a system [4], [5], [6]. The concepts of Safety Instrumented Systems (SIS) might be a solution to these problems. Risk reduction layers based on hierarchical control solutions can manage risks through prevention and mitigation layers designed to lead the system to a safe state in case of faults [7]. On this context, safety standards as the IEC 61508 [8] and IEC 61511 [9], provide guidelines for the SIS life cycle, ranging from development to decommissioning.

The processes of understanding, specifying and developing such complex and critical systems has become a highly complex task, thus design flaws are intrinsic to their development [10], [3]. Critical processes controlled by PLCs – such as oil and gas refineries, where faults may cause severe accidents – requires great reliability from their control programs and thus, they not only need to be subject of formal verification processes [11], [12], but also have to be developed according to the safety control program development cycle defined on the IEC 61511 standard [13].

We found distinct approaches to deal with these problems on the literature. Approaches as presented in [14] and [15] propose frameworks for safety control software development as well as the control software architecture, however they do not include formal verification phases or were proposed according to the IEC 61511 standard. It is worth mentioning that the framework proposed in [15] presents modularity concepts and is one of the few that distinguish between prevention and mitigation control programs. Approaches as the ones presented in [16] and [17] only focus on proposing methodologies for the formal verification of exiting control programs. In [18] is presented a proprietary approach for the model based development of safety control programs according to the IEC 61508 standard, no details are given on how each phase of the cycle is implemented. In general terms,

The authors would like to thank the Brazilian governmental agencies CNPq, FAPESP, and CAPES for their financial support to this work.

Authors with Polytechnic School – Department of Mechatronics Engineering and Mechanical Systems, University of São Paulo - São Paulo, Brazil. (+55 11 30916025; e-mail: rferrarezi@usp.br, reinaldo.squillante@usp.br, jeferson.souza@usp.br, reinaldo@usp.br, fabri@usp.br, pemiyagi@usp.br, lamoscat@usp.br, jokamoto@usp.br, diolinos@usp.br).

the safety control software is modelled as blocks diagrams and state machines and automatically translated to control code. The control code is then verified through observers, where only if a given input produces the expected output is verified. In [19] a process is presented for the model based development of control programs and their formal verification from their informal specifications. It is also discussed the necessity of including the plant behavior model on the control model for proper verification. However this approach was not proposed for safety programs and does not present solutions to minimize the state space explosion problem.

On this work we propose a framework for the development of SIS control programs. The proposed framework: First, will be based on the phases of the safety control program development cycle of the IEC 61508 and IEC 61511 standards. Second, will take into consideration and analyze the relationships between the prevention and mitigation layers. Third, will be based on the Model Based Design (MbD) [3] approach. Considering the approaches on the literature, with this work it is expected to put together the best practices for the development of safety control programs based on the cycle of the IEC 61508 and IEC 61511 standards, as well as to deal and propose solutions for the shortcomings arriving in a framework that can be used for control engineers to develop solutions for real world problems.

This work is organized as follows: Section II will present a short review of the main concepts used in this work. Section III will present a description of the proposed framework. Section IV will present an application of the framework to a real case scenario. Section V will present the results and conclusions.

II. MATERIALS AND METHODS

A. *The GHENeSys environment*

SIS control systems can be viewed as an event driven system, presenting functional characteristics as asynchronism, reset possibility, parallelism, concurrence, etc. Thus they can be classified as discrete event systems (DES) and modelled by Petri Nets, [5], [20] and its extensions.

The GHENeSys environment is being developed with the goal of representing, in a unified way, classical Petri Nets, its extensions defined on the ISO/IEC 15909 standard, and High Level Petri Nets [21]. The GHENeSys environment is composed of the following basic modules: The GHENeSys nets and the Editor tool, the simulation module and the verification tool [22].

The GHENeSys environment implements several concepts to aid the modeling process, such as: Pseudoboxes that allow easier modelling of the exchange of information between different parts of the system; Hierarchy that allows the encapsulation of subnets without losing any properties through the use of macro elements; The representation of non-deterministic time periods, where lower and higher bound time intervals can be set for transitions and places.

The GHENeSys net is the tuple $G = (L, A, F, K, \Pi, C_0, \tau)$, where:

- $L = B \cup P$ is the set of places, which can be boxes or pseudoboxes;
- A are the activities, or active elements;
- $F \subseteq (L \times A \to \mathbb{N}) \cup (A \times L \to \mathbb{N})$ is the flux relation;
- $K : L \to \mathbb{N}^+$ is the capacity function;
- $\Pi : (B \cup A) \to \{0,1\}$ is the function that identify the macro elements or the hierarchy;
- $C_0 = \{(l, \sigma_j) | l \in L, \sigma_j \in \mathbb{R}^+ | l | < K(l)\}$ is the set of initial marks;
- $\tau : (B \cup A) \to \{\mathbb{Q}^+, \mathbb{Q}^+ \cup \{\infty\}\}$ is the function that maps the dense time intervals for each element.

The set of markings is the pair (l, σ_j) with $l \in L$, defining the place each token can be found and σ_j defining for how long this token will remain in the place. The time measurement is globally synced and updated after each transition.

The GHENeSys verification tool performs the formal verification of real time concurrent systems modelled by GHENeSys net through Model Checking [23] techniques. The state space is constructed using the enumerative approach based on state classes [24] concept. The tool has options to build SCG, SSCG and CSCG state graph types. Checked properties are specified through TCTL [25].

The GHENeSys environment will be used on this work due to several reasons: (i) Due to the characteristics of the SIS, the amount of checked properties may be very large, so it is desirable that the state space is generated through the enumerative approach once instead of being generated on-the-fly several times. The state space generation is done in exponential space and time, and the verification of a property is performed in polynomial time, if the state space is already constructed. (ii) The use of the dense time approach, as several SIS properties are time dependent. (iii) PNML [26] is the default transfer format. This allows the interchange of information between all GHENeSys tools, as well as with external tools that support the standardized format. (iv) All modeling and verification tasks are performed with no need of external modules or tools. (v) The state space generation and the specification of the tested properties are performed by the same tool.

III. PROPOSED FRAMEWORK

The proposed framework will be presented according to the "V-model" from the IEC 61511 standard [9]. The main components of the framework and their relations with the phases of the "V-model" are displayed on Fig. 1. Each framework component was proposed to comply with one or more phases of the safety program development cycle.

The first phase of the development cycle is related with each SIS subsystem requirement elicitation. This phase is out of the scope of the framework and must be addressed by the SIS control program development methodologies, which will be discussed later.

The methods, techniques and tools that will be used during

the control program development must be chosen on the first part of the second phase. As already discussed on previous sections, the GHENeSys environment was chosen for the modelling and formal verification of SIS control programs.

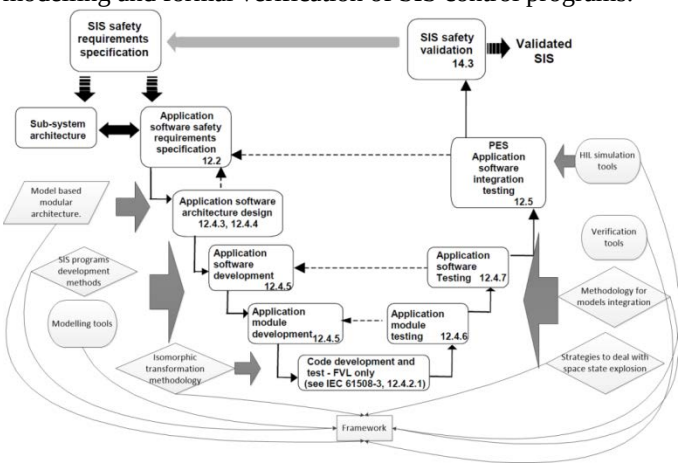

Fig. 1 - Framework + "V-model"

The control program development on the framework will be performed according to the Model Based Design (MbD) approach [3]. Although this approach is not referred in the IEC 61511 standard, as according to it, all control programs shall be developed in an implementation language. The standard requires that a final validation must be performed by the end of the development cycle, and modeling is one of the recommended tools for this validation. Thus by adopting the MbD, the framework will be not only complying with the standard but also improving the modularity of the SIS control programs.

A. Control program architecture modelling methodology

Besides the modeling tools, a modeling methodology also must be chosen or proposed. This methodology must comply with the modularity requirements of the IEC 61511 standard and allow a high-level view of the modeled system together with the necessary refinements.

There are several methodologies on the literature for the modelling and development of control programs as in [27], [28] and [29] among others. However these methodologies were developed for productive / manufacture systems and are not suitable for SIS development. As opposed to a usual productive system, a SIS is designed to oversee a system and to react on events received at any point of the control code execution, degenerating the system to a safe state. Also, control programs for productive systems usually present a "linear" structure, that is, the system receives some input – or raw material –, this material is then processed or transformed resulting on an output – or product – by the end of the program execution. SIS control programs are not designed to produce outputs in this sense, but to decide which control actions are necessary based on the current state of the overseen system. As a productive system process goods, a SIS process information.

Based on the discussed distinctions between a productive system and SIS we are proposing a methodology for SIS control program architecture modelling in GHENeSys nets:

i. Main activity definition: The highest level activity is defined.
ii. Definition of the independent macro processes: On this step the main SIS processes must be defined. A SIS might be composed of processes of fault prevention and mitigation for example. Also a coordination process might be proposed to coordinate other processes.
iii. Sub-processes or functionalities definition: The processes must be refined on the necessary functionalities. For instance, a process of fault prevention might need at least the fault detection and actuation functionalities.
iv. Basic operations definition: The functionalities are further refined in the necessary basic operations. For instance, detection functionality might require Boolean logic operations for fault detection, filters for spurious readings from the sensors, among others. Basic operations can be shared between functionalities.
v. Basic operation refinement: All operations must be refined according to the rules for Petri Net reduction or refinement presented in [30] in order to execute the logic they were designed to. A GHENeSys net box must be defined as the holder of the binary result of the modelled logic.
vi. External signals representation: All necessary external signals for the execution of each operation must be represented. External signals can be sensors, actuators or even other operation output signals.

By using the presented methodology we expect to generate the maximum possible amount of generic operations, being the control program designer responsible to choose the operations necessary for the developed system. New functionalities can also be developed if needed.

B. Control program architecture

The second, and final, part of the second phase of the development cycle regards the control program architecture. The architecture is developed according to the methodology introduced on the last section.

i. Main activity definition: The main, and highest level, activity is the whole SIS control program.
ii. Definition of the independent macro processes: After refining the SIS activity we have the prevention and mitigation macro processes.
iii. Sub-processes or functionalities definition: Both prevention and mitigation activities need at least the fault detection and fault treatment functionalities.
iv. Basic operations definition: On this step we hope to refine the most common basic operations needed for SIS development found in several related works as [5] e [31].
v. The detection functionality is refined in four operations. The spurious event filter operation

avoids spurious readings from poorly calibrated sensors, for instance, thus avoiding unwanted SIS activation and system degeneration. The Boolean logic "AND" and "OR" operations can be used to implement logics for faults detection between several sensors. The voting logic operation is using to implement voting logics – like 2oo3 (two out of three, 1oo3 (one out of three) and so on. As these logics are very common in SIS development they will be implemented as a full operation even though they can be implemented through Boolean operations.

vi. The treatment functionality is also refined in four operations. The Boolean logic operations are the same of detection. The sequencing operation provides sequencing for actuator signals. The actuation operation sends signals for the actuators and receives signals from detection operations and command devices – if used.

vii. Basic operation refinement: The operations described on the last item are now further refined in order to obtain the structures that can implement the proposed logic in GHENeSys nets. The refinements of a 2oo3 voting operation are shown on Fig. 2. The "Voting-ON" box stores the information if the voting operation is true or not based on the signal from the sensors. The refinements were done for all other operations.

viii. External signals representation: Now the sensors signals can be represented on the operations structures to enable the desired logics. The complete 2oo3 voting logic is shown on Fig. 3. This operation receives three input signals – represented as the pseudo-boxes "ps-Sensor-A/B/C" – that, if active in pairs, make the "Voting-ON" box receive one mark. The box loses its mark if any pair is no longer active. Input signals can come either from sensors or any other detection operation.

C. Control program development

The third and fourth phases of the development cycle are related to the development of the control program. The framework allows the user to choose the methodologies for the development of the prevention and mitigation functionalities. These methodologies must be based on formal models and be in accordance with the IEC 61508 and IEC 61511 standards. The methodologies must supply the necessary information to build the control programs from the basic framework operations.

1) Prevention Macro Process Development Methodology

The prevention macro process development methodology is responsible to identify and select the faults that will be treated by the SIS. Faults can be identified through HAZOP reports, cause-effect matrix or other applicable technique. The methodology must then apply the techniques to discover the causal relations leading to each selected fault, that is, how – by which sensors – each fault can be detected.

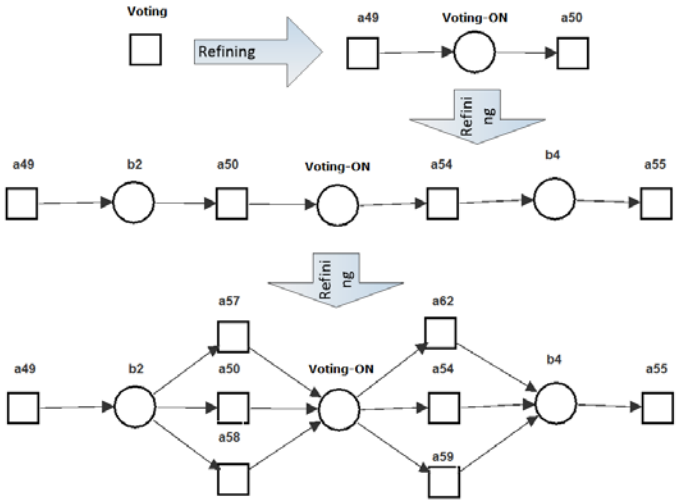

Fig. 2 - Voting operation structural refinement

Besides proposing how each fault can be detected, the methodology must also propose actions to treat each fault. The treatment is usually performed by the actuation of one or more components – as control valves – and/or the shutdown of an endangered plant equipment. The command signals, if any, must be defined and the preset times used on the spurious event filter block must be informed in case this block is implemented. All collected fault information constitutes a SIF.

From SIF information, the necessary basic operations to model the prevention macro process can be selected. Operations have their input and output signals connected until the control program model is constructed.

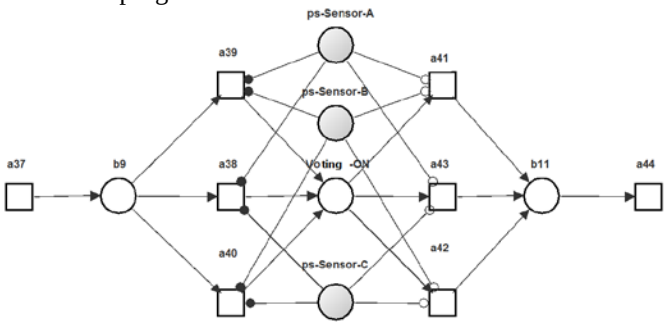

Fig. 3 - 2oo3 Voting operation

2) Mitigation Macro Process Development Methodology

Now, a mitigation macro process development methodology must be chosen, either the same methodology used for prevention or a new methodology can be chosen. As opposite to the prevention activity, the mitigation activity does not need to select the faults that will be treated by the mitigation SIS. The mitigation SIS will treat faults on the prevention SIS execution, that is, the mitigation SIS will mitigate the effects of the prevention SIS not being able to lead the plant or process to a safe state.

The mitigation methodology, as the prevention methodology, supplies the necessary information to constitute the SIF, that is, how each effect can be detected and treated. Necessary command signals and preset times must also be

provided by the methodology. From SIF information the necessary basic operations to model the mitigation macro process are selected. Operations have their input and output signals connected until the control program model is constructed.

D. Verification Process

The sixth and seventh phases of the development cycle are related to the formal verification of the models. Due to the Model based Design approach, the formal verification is performed on the control program models, not on the control code. The fifth phase of the development cycle (control code generation) will be performed on the models after the verification processes. All properties described initially in natural language shall be translated to TCTL according to the patterns presented in [32]. Models are integrated according to methodology described below. The verification will be performed in three different contexts which are detailed afterwards.

1) Integration of the models

The verified models must be integrated after each verification context. First, the prevention and mitigation models are integrated resulting on the complete treatment model of a single fault. The treatment of several faults is then integrated, resulting on the complete SIS model. The integration is performed according to the steps below:

i. Shared sensors and actuators between models are listed.
ii. Shared sensors signals are duplicated. Each model keeps its basic operations, like spurious events filters, voting, etc. Only the sensor signal is duplicated.
iii. The activation of sets of shared actuators is implemented through "OR" logic operations between the pertinent actuation operations. Sequencing between actuators must be respected and implemented through sequencing operations.

2) Models context

The first verification is performed in each prevention and mitigation model. This first context verifies if the models are capable of identifying and treating the fault they were designed to identify. The verified properties can be extracted from each SIF description. The reachability of undesired states must also be verified, that is, the reachability of states that might endanger the protected system or process must be verified. Natural language properties are translated to TCTL propositions and inputted on the GHENeSys verification tool.

3) Fault treatment context

The verified and corrected, if necessary, prevention and mitigation models are now integrated and their relationship is evaluated. The second context verifies if the integrated models can properly work together, that is, if the prevention model can jeopardize the mitigation model and vice-versa. Verified properties can be proposed from the analysis of the integrated GHENeSys model structure.

4) SIS context

The last verification step is performed on the complete SIS model, where the treatment of several faults is integrated. As the on the fault treatment context, the SIS context verifies if the integrated models can properly work together, that is, if the treatment of one fault can jeopardize the treatment of others faults. From the analysis of the integrated GHENeSys model structure the verified properties can be proposed. Desirable events such as, shared actuators being activated by all pertinent models and sequencing operations, must be verified. Undesirable events, such as one model being able to activate actuators that are not part of treatment of the fault, must also be verified.

5) Strategies to deal with the state space problem

The basic architecture of a discrete event system (DES) is composed of command and sensoring devices that send environmental information to the control module. The control module processes the information according to its control software logics and sends information to the output devices, such as monitoring and actuation devices. As a SIS can be classified as a DES, they share the same architecture.

However, although sharing the same basic architecture, they have distinct verification contexts. The verification of the control software for SED – such as manufacture systems – is aimed in checking if the control software fulfills its design requirements, that is, if the software has the desired behavior based on a fixed set of inputs (or initial states). The verification of SIS control software is aimed not only on verifying if the software fulfills a set of requirements, but also if the system does not reach undesirable states that might be dangerous. Thus the model of the plant must be attached to the SIS control program model to generate all possible sets of inputs.

As modelling the behavior of a plant is out of the scope of this work, a very simple model for the plant was chosen, on this model every input device is modelled with two states – GHENeSys boxes –, one is marked when the signal is active, and the other is marked when the signal is inactive. Although this model does not represent precisely the behavior of the plant, it can generate all possible sets of inputs.

The main downside of this approach is the explosion of the state space, thus we proposed some strategies to deal with this problem and shorten verification duration.

i. Actuators models suppression

On SIS, usually a single actuation signal activates several actuators, so the nets representing the actuators behavior can be suppressed, as only the actuation signal needs to be part of the verification. Thus, the size of the state space can be reduced without losing any important information regarding the behavior of the system.

ii. Segregation of the models

Another strategy to reduce the state space explosion problem is the segregation of the models. This segregation can be performed on mitigation models, as the mitigation model of one fault is composed of the treatment of several effects. A usual mitigation model constructed according to this framework has the structure shown on Fig. 4. On the figure, the effect "A" is detected by the sensors "A", the signal of the sensors is sent to the detection logic "A" and the logic

activates the actuation operation "A". This operation finally activates the actuators "A/B/C", which are shared among all effects. The same happens for the effect "B", except that the actuators "A/B/C" and "B/C" are activated. The effect "C" activates the actuators "A/B/C", "B/C" and "C".

Fig. 4 - Mitigation program structure

We propose to perform on this type of model four distinct verifications. On the first three verifications (Verification 1, 2 and 3) the model of each effect treatment is verified alone, with other models disabled. These first three verifications are intended to verify if the treatment of each effect has been properly implemented. The last verification (Verification 4) is intended to verify the interactions between the actuation of all effects. That is, if each effect treatment model can only activate its actuators or if some effect can jeopardize the treatment of another. By verifying only the actuation relations, the model does not need the detection blocks anymore – as each model was already verified individually – leading to smaller models and thus shorter verification durations.

iii. Segregation of the context

As the models are integrated for the verification of the next context they inevitably get increasingly bigger. However, the integrated models were already verified on the previous context, so only their actuation interactions need to be verified. Much like the "Verification 4" of the previous item, the sensors and detection logic models can be removed from the models, just by allowing the actuation operation to generate random signals all possible combinations are inputed on the actuation logic and their relation can be verified.

E. Control program codding

With the SIS control program properly verified it is possible to start the fifth phase of the development cycle that is related to the control program codding.

One of the advantages of using the model based approach is the possibility of automatically transforming the models into a language allowed by the IEC 61131-3 [33] standard. To perform this isomorphic transformation we propose to use the method described in [34]. This systematic generates modular LD diagrams, where it is possible to recognize blocks resulting from the transformation of transitions, places or outputs. The systematic is composed of the following steps:

i. Assign LD internal variables to transitions.
ii. Assign LD internal variables to places.
iii. Internal LD events must be associated with output signals as monitoring and actuation.
iv. External LD events must be associated with input signals as command and sensoring.
v. For each transition generate the corresponding LD
    a. One rung for each transition
    b. Each rung must contain AND operations between the transition preconditions and restriction conditions (external events)
    c. In case of timed transitions, TON elements must be added before the relay that represents a transition on its rung.
vi. For each local state change generate the corresponding LD
    a. First the initial marking is generated.
    b. The marking of each place is updated by set/reset relays
vii. Generate the LD for the external events
    a. Output activation LD

Normal relays without memory must be used as the local states remain active until some state transition happens.

F. Control program tests

The eighth phase of the development cycle is the integrated test of the control program and the control hardware. On this framework, the control program, now translated to an IEC 61131-3 standard language, is uploaded to the safety PLC and submitted to exhaustive plant simulation tests. These tests may be performed through HIL [35] techniques. This phase shall be further developed on future works.

IV. FRAMEWORK APPLICATION

The proposed framework was applied to the development of the SIS control program for a natural gas compression plant (ECOMP). As this type of plant works with highly flammable fluids, the process risk is too high, demanding the implementation of SIS.

A. Process description

The ECOMP plant is connected to the main gas pipeline through the suction pipeline. The incoming gas is filtered on two coalescent filters then distributed between four turbo compressors. The compressed gas then returns to the main gas pipeline through the discharge lines. There are temperature sensors, fire and gas detectors on each turbo compressor discharge lines as well as pressure sensors on the main discharge line. Several ON-OFF valves are installed on the suction lines, between the filters, on the turbo compressors, and on the discharge lines. Also there are ON-OFF valves for the emergency discharge lines, and on the CO_2 tanks connected with the suction of each turbo compressor. All equipment TAGs used are according to the SA-S5.1-1984 [36]

standard.

B. Development of the control program

Now that the process is described, the methodologies for the development of the prevention and mitigation functionalities must be chosen.

1) Development of the prevention macro process

The prevention macro process will be developed according to the methodology proposed in [5]. Briefly, the faults that will be treated by the prevention SIS are extracted from the HAZOP report. The detection models are generated from cause-effect matrixes, and then those matrixes are converted into Bayesian Networks, which are finally converted into Petri Nets. The treatment models, that is, the actuators used on the treatment of each fault, are generated from the HAZOP report.

Applying the methodology for the ECOMP results in a table where it is possible to find the faults that will be treated, their SIF number, the initializing events – the sensors that are capable of detecting such fault – and the prevention actions, that is, which valves need to be closed and/or which equipment must be shut-down, and so on. The logic relations between the sensors and command signals can be found on the generated Petri.

The first fault is "High pressure on the discharge lines". This fault can be detected by the sensors $(\mathrm{PIT}-006\mathrm{A}, \mathrm{PIT}-006\mathrm{B}, \mathrm{PIT}-006\mathrm{C})$ connected by a 2oo3 voting logic. The treatment is performed by closing six ON-OFF valves and by shutting-down all four turbo compressors. The second fault is "High temperature on the turbo compressors discharge". This fault can be detected by the sensors connected by the logic $OR(\mathrm{TIT}-209\mathrm{A}, \mathrm{TIT}-209\mathrm{B}, \mathrm{TIT}-209\mathrm{C}, \mathrm{TIT}-209\mathrm{D})$. The treatment is performed by shutting-down all four turbo compressors. Also, a command signal resetting the models was implemented.

Finally, both models will need the spurious event filter and the actuation operations. The first fault treatment model will need the 2oo3 voting operation, and the second fault treatment model will need the "OR" logic operation. With the necessary blocks, the high level GHENeSys model is assembled and refined until the full model is obtained. Unfortunately the models are too big to be presented on the paper.

2) Development of the mitigation macro process

The mitigation macro process will be developed according to the methodology proposed in [37]. The mitigation treatment is focused on the mitigation of the effects of the prevention SIS failing to lead the plant to a safe state. The detection models are generated from how to detect each effect described on the FMEA [38] report. Relations between sensors are described on the FTA [38] report. Treatment models are constructed from What-if techniques [39].

The application methodology on the ECOMP results on What-if, FMEA and FTA reports. From these reports it is possible to obtain the sensors and actuators related with each effect, as well as their logical relations. As example, fire may be one of the effects of the first prevention fault not being properly treated; as if the temperature is not lowered the pressurized gas can leak and ignite. This effect can be detected by the sensors $OR(\mathrm{BSH}-501, \mathrm{BSH}-502, \mathrm{BSH}-503, \mathrm{BSH}-504)$. The treatment is performed by first, performing the actions from the prevention SIS, second, opening the emergency discharge line, third, closing the filters outlet vales and, after all these actions are performed, finally opening the valves from the CO_2 tanks.

Finally, the example effect will need the spurious event filter and "OR" logic operations for the detection functionality. "AND" and "OR" logic, sequencing and the actuation operations for the actuation functionality. Logic operations are necessary in the actuation blocks as some actuators are shared among treatment models and there are sequencings that must be respected. With the necessary blocks, the high level GHENeSys model is assembled and refined until the full model is obtained.

C. Models verification

The strategies for the reduction of the state space problem were applied to all models, integrated or not. The amount of checked properties and the verification duration for each model can be observed on the Table I.

Table I - Verifications

Model	# of properties	# of State classes	Time elapsed
Fault 1 Prevention	3	2785	24s
Fault 4 Prevention	3	13566	9m 30s
Fault 1 Mitigation - Effect 1	3	5061	3m 11s
Fault 1 Mitigation - Effect 2	3	24827	1h 1m 24s
Fault 1 Mitigation - Effect 3	3	38648	2h 28m 16s
Fault 1 Mitigation - Interaction	2	178	1s
Fault 4 Mitigation - Effect 1	3	24827	1h 14m 18s
Fault 4 Mitigation - Effect 3	3	38648	2h 10m 6s
Fault 4 Mitigation - Interaction	2	66	1s
Fault 1 integrated	4	2148	15s
Fault 4 integrated	4	885	3s
Complete SIS	4	109963	4h 17m 33s

All verifications were performed on an Intel Core i7 3770 (@3.4 GHz), 16GB DDR3 SDRAM, 1 TB HDD machine running Windows 7 SP1 with Java Version 7 Update 67. Some examples of checked properties are given on Table II, together with their respective TCTL propositions. All formulas were checked with satisfactory results.

Where "P" stands for prevention, "M" for mitigation, "E1", "E2", "E3" stand for each effect treated by mitigation, and "ENABLE_EXT" stands for the command signal. "PSHH-

006A/B/C" is the signals from the pressure sensors.

TABLE II. CHECKED PROPERTIES

Models level – Fault 1 prevention
Actuators remain activated and the compressors remain off until the fault is not detected anymore by the sensors for 10 sec. and the command signal is active.
$\neg\exists\left(\left(((PSHH-006A \wedge PSHH-006B) \vee (PSHH-006A \wedge PSHH-006C) \vee (PSHH-006B \wedge PSHH-006C)) \wedge ENABLE\_EXT\right) \square_{\geq 10}\left((P1-Actuation) \wedge (((PSHH-006A \wedge PSHH-006B) \vee (PSHH-006A \wedge PSHH-006C) \vee (PSHH-006B \wedge PSHH-006C)) \wedge ENABLE\_EXT)\right)\right)$
SIS level – Integrated
Common actuators are activated by all models.
$\forall\square_{\geq 0}((P1-Actuation \vee P4-Actuation \vee M1-E1-Actuation \vee M1-E2-Actuation \vee M1-E3-Actuation \vee M4-E1-Actuation \vee M4-E3-Actuation) \rightarrow \forall\Diamond_{\geq 0}(SIS-Actuators))$
The activation of any prevention model never triggers any actuators exclusively related with the mitigation models.
$\neg\forall\square_{\geq 0}(((P1-Actuation \vee P4-Actuation) \wedge \neg M1-E1-Actuation \wedge \neg M1-E2-Actuation \wedge \neg M1-E3-Actuation \wedge \neg M4-E1-Actuation \wedge \neg M4-E3-Actuation) \rightarrow \forall\square_{\geq 0}(M-Actuators \wedge M-E3-Actuators))$

V. CONCLUSION

On this work we presented a framework for the model based development of SIS control programs according to the phases provided by the safety software development cycle from the IEC 61508 and 61511 safety standards. The GHENeSys environment was chosen as the main tool to aid the development; this choice was also justified as per safety standard requirement. The modular development of the control software was allowed by the proposed methodology for the architecture development. This methodology not only was used to propose the control program architecture considering the most common SIS software components, but also can be used by the control engineers to develop others architectures fitting their specific needs and still respecting the safety standards.

After the control program architecture was proposed, the development of the control program was discussed. Guidelines for the choice of the methodologies for the development of the prevention and mitigation functionalities were discussed, where these methodologies must supply the information needed to assemble the modular operations that will result on the final control program model.

The verification processes were broken into contexts and guidelines to propose the checked properties in each context were supplied. The modular approach allowed the easy integration of the simpler models until the complete SIS model is obtained. After extensive modeling and testing, some strategies to deal with the state space explosion problem were proposed. Also, a methodology for the isomorphic transformation of the SIS control program model to control code was discussed.

Finally, the framework was applied to the development of a SIS control program for a real gas compression plant. The framework was able to deal with all the specific needs of the project, the generated models presented modular architecture and, thanks to that, can be easily understood and maintained. The models were verified always considering the relationship between the integrated parts and their possible positive or negative interactions. Even with the integrated plant model the framework could also successfully decrease the verification duration by applying the proposed strategies – before the application of the strategies only the two smallest models could be verified, state space generation would not finish for weeks for the other models – and thus even the biggest model – the complete SIS model – could be successfully verified in around four hours. Thereby the framework proved applicable to the development of real scenario SIS control systems according to the safety standards.

REFERENCES

[1] M. Bani Younis and G. Frey, "Formalization of Existing PLC Programs: A survey," Kaiserslautern, 2003.

[2] C. Yamada, Y. Nakaga and M. Nakahodo, "An Efficient Model Checking Using Check-Points Extraction Method," *International Journal of Computers*, vol. 1, no. 3, pp. 95-101, 2007.

[3] M. Mazzolini, A. Brusaferri and E. Carpanzano, "An Integrated Framework for Model-based Design and Verification of discrete Automation Solutions," in *Proceedings 2011 9th IEEE International Conference on Industrial Informatics*, Milan, 2011.

[4] M. Sallak, C. Simon and J.-F. Aubry, "A Fuzzy Probabilistic Approach for Determining Safety Integrity Level," *IEEE Transactions on Fuzzy Systems*, vol. 16, no. 1, pp. 239-248, 2008.

[5] R. Squillante Júnior, D. J. Santos Filho, F. Junqueira and P. E. Miyagi, "Development of Control Systems for Safety Instrumented Systems," *IEEE Latin America Transactions*, vol. 9, no. 4, pp. 451-457, Julho 2011.

[6] J. Börcsök and P. Holub, "Consideration of common cause failures in safety systems," in *ACACOS'08 Proceedings of the 7th WSEAS International Conference on Applied Computer and Applied Computational Science*, Hangzhou, 2008.

[7] G. Florea and R. Dobrescu, "Risk and Hazard Control the new process control paradigm," in *Communications, Circuits and Educational Technologies - Proceedings of the 2014 International Conference on Electronics and* , Prague, 2014.

[8] IEC, "IEC 61508 - Functional safety of electrical/electronic/programmable electronic safety-related systems," International Electrotechnical Commission, Geneva, 2010.

[9] IEC, "IEC 61511 - Safety instrumented systems for the process industry sector," International Electrotechnical Commission, Geneva, 2003.

[10] M. Diaz, Petri Nets - Fundamental Models, Verification and Applications, London: John Wiley & Sons, 2009.

[11] M. M. Patil, S. Subbaraman and S. Joshi, "Exploring Integrated Circuit Verification Methodology for Verification and Validation of PLC Systems," in *International Symposium on Electronic System Design*, Kochi, 2011.

[12] H. Wan, X. Song, G. Chen and M. Gu, "A Refinement-Based Validation Method for Programmable Logic Controllers," in *10th International Conference on Quality Software*, Zhangjiajie, 2010.

[13] A. Mayr, R. Plösch and M. Saft, "Towards an Operational Safety Standard for Software - Modelling IEC 61508 Part 3," in *18th IEEE International Conference and Workshops on Engineering of Computer-Based Systems*, Las Vegas, 2011.

[14] S. Richter, M. Wahler and A. Kumar, "A framework for component-based real-time control applications," in *13th Real-Time Linux Workshop*, Prague, 2011.

[15] H. A. Gabbar, "Integrated framework for safety control design of nuclear power plants," *Nuclear Engineering and Design*, vol. 240, no. 10, p. 3550–3558, 2010.

[16] C. A. Sarmento, J. R. Silva, P. E. Miyagi and D. J. Santos Filho, "Modeling of Programs and its Verification for Programmable Logic Controllers," in *Proceedings of the 17th World Congress* , Seoul, 2008.

[17] G. Canet, S. Couffin, J.-J. Lesage, A. Petit and P. Schnoebelen, "Towards the automatic verification of PLC programs written in Instruction List," in *IEEE International Conference on Systems, Man, and Cybernetics*, Nashville, 2000.

[18] A. Bouali and B. Dion, "Formal Verification for Model-Based Development," in *SAE Technical Paper 2005-01-0781*, Newswire, 2005.

[19] E. I. Gergely, L. Coroiu and A. Gacsadi, "Design of Safe PLC Programs by Using Petri Nets and Formal Methods," in *Proceedings of the 11th WSEAS international conference on Automation & information*, Romania, 2010.

[20] R. Zurawski and M. Zhou , "Petri nets and industrial applications: a tutorial," *IEEE Transactions on Industrial Electronics*, vol. 41, no. 6, p. 567–583, 1994.

[21] P. M. G. del Foyo and J. R. Silva, "Towards a unified view of Petri nets and object oriented modeling," in *In 17th International Congress in Mechanical Engineering*, São Paulo, 2003.

[22] P. M. G. del Foyo, A. S. P. J. Miralles and J. R. Silva, "UM VERIFICADOR FORMAL EFICIENTE PARA SISTEMAS DE TEMPO REAL," in *X SBAI – Simpósio Brasileiro de Automação Inteligente*, São João del-Rei, 2011.

[23] E. M. Clarke, O. Grumberg and D. A. Peled, Model Cheking, 1st ed., Cambridge: MIT Press, 1999.

[24] B. Berthomieu and M. Menasche, "An Enumerative Approach For Analyzing Time Petri Nets," in *Proceedings IFIP*, Paris, 1983.

[25] R. Alur, C. A. Courcoubetis and D. L. Dill, "Model-checking for real-time systems," in *Proceedings of the Fifth Annual IEEE Symposium on Logic in Computer Science*, Philadelphia, 1990.

[26] ISO/IEC, "Software and Systems Engineering - High-level Petri Nets, Part 2: Transfer Format, International Standard WD ISO/IEC 15909. Wd version 0.9.0," 2005.

[27] M. Bonfe and C. Fantuzzi, "Object-oriented approach to PLC software design for a manufacture machinery using IEC 61131-3 norm languages," in *Proceedings of IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, Como, 2001.

[28] G. Di Orio, J. Barata, C. Sousa and L. Flores, "Control System Software Design Methodology for Automotive Industry," in *IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, Manchester, 2013.

[29] M. Ko, S. C. Park, J.-J. C. Chang and M. Chang, "New modelling formalism for control programs of flexible manufacturing systems," *International Journal of Production Research*, vol. 51, no. 6, pp. 1668-1679, March 2013.

[30] T. Murata, "Petri nets: Properties, analysis and applications," *Proceedings of IEEE*, vol. 77, no. 4, p. 541–580, 1989.

[31] A. Cavalheiro, D. Santos Filho, A. Andrade, J. R. Cardoso, E. Bock, J. Fonseca and P. E. Miyagi, "Design of Supervisory Control System for Ventricular Assist Device," in *Second IFIP WG 5.5/SOCOLNET Doctoral Conference on Computing, Electrical and Industrial Systems*, Caparica, 2011.

[32] M. B. Dwyer, G. S. Avrunin and J. C. Corbett, "Property Specification Patterns for Finite-state Verication," in *Proceedings of 2nd Workshop on Formal Methods in Software Practice*, Clearwater Beach, 1998.

[33] IEC, "IEC 61131-3 - Programmable controllers - Part 3: Programming languages," Geneva, 2003.

[34] D. J. Santos Filho, "Aspectos do Projeto de Sistemas Produtivos," Tese de Livre Docência, Escola Politécnica da Universidade de São Paulo, São Paulo, 2000.

[35] F. Gu, W. Harrison, D. Tilbury and C. Yuan, "Hardware-In-The-Loop for Manufacturing Automation Control: Current Status and Identified Needs," in *CASE 2007. IEEE International Conference on Automation Science and Engineering*, Scottsdale, 2007.

[36] ISA, "ANSI/ISA-S5.1 — Instrumentation Symbols and Identification," Research Triangle Park, 1984 (R1992).

[37] J. A. L. Souza, D. J. Santos Filho, P. E. Miyagi, R. Squillante Junior and R. C. Ferrarezi, "Critical Systems: a New Approach in Mitigation Control Layer," in *IFAC2014*, Cape Town, 2014.

[38] M. Modarres, M. Kaminskiy and V. Krivtsov, Reliability Engineering and Risk Analysis: A Practical Guide, 2 ed., New York: CRC Press, 2009.

[39] Center for Chemical Process Safety, Guidelines for Hazard Evaluation Procedures, 3 ed., New York: Wiley-AIChE, 2008.