# Ontology-based Framework for the Generation of Interlock Code with Redundancy Elimination

Michael Steinegger, Martin Melik-Merkumians, and Georg Schitter

Automation and Control Institute, Vienna University of Technology
Gußhausstraße 27-29, AT-1040 Vienna, Austria
Email: {steinegger,melik-merkumians,schitter}@acin.tuwien.ac.at

***Abstract*—In this paper, a framework for the automatic generation of redundancy-free interlock code is proposed. The core component is represented by an ontology-based integration framework that holds the relevant information from a developed piping and instrumentation diagram (P&ID) editor as well as the PLC program where the interlocks should be added. The integration framework is mainly based on AutomationML concepts as glue model between the different internal models of the P&ID editor and the PLC programming tool. In order to enable the co-existence of manual and automatically generated interlocking code, the approach is extended by additionally applying a theorem prover. Here, the theorem prover Z3 is utilized to eliminate redundant interlock logic whereas the redundancy-free logic is then automatically added as interlock code to the IEC 61131-3 compliant program. The framework is evaluated by application to a laboratory process plant.**

## I. Introduction

The effort for design and implementation of process automation systems increases with the complexity of modern production plants [1]. Typically, the control applications for process plants are developed based on plant-specific engineering documents [2]. In order to support software engineers or even partially automatize the implementation process, several attempts have been made to achieve integration solutions to enable the semantic integration of design data distributed over the individual involved engineering tools [3]. For instance, there exist approaches to enable a completely integrated tool chain and bridge the aforementioned semantic gaps like PlantXML [4], the international standard ISO 15926 [5], or the engineering service bus concept [6].

In recent years, several approaches have been developed to support and simplify the design process of industrial control applications. For instance, a markup language for component-based design of industrial control applications is proposed by Estévez et al. [7], where *programmable logic controller* (PLC) code is then automatically generated from the application model designed with the proposed markup language. For interlock code Drath et al. [8] proposed a rule-based generation process generating a cause-and-effect table in a first step, which is then translated to PLC code. However, since usually there exists additional control code within a PLC program which could also have been implemented manually by the software engineer, the combination of the generated (interlock) code and eventually existing PLC code is not addressed.

In this paper, an ontology-based framework to combine manually defined and automatically generated interlock code for PLCs is proposed. The required data is retrieved from an own implemented *piping and instrumentation diagram* (P&ID) editor, whereas the internal tool model is mapped to the ontologies based on AutomationML. Additionally, to enable the co-existence of user-defined code and automatically added interlock logic code, a redundancy elimination algorithm utilizing the theorem prover Z3 [9] is proposed, which prevents redundant interlock logic within the PLC program.

The remainder of the paper is organized as follows. An overview on the interlock generation framework and its core component is given in Sec. II. The code generation process and the proposed redundancy elimination approach is described in Sec. III. In Sec. IV, the approach is evaluated by application to a laboratory process plant. Section V concludes the paper.

978-1-5090-6505-9/17/$31.00 © 

## II. Concept Overview

The proposed framework is depicted in Fig. 1 and is based on an integration framework composed out of different ontologies. Parts of the ontologies have already been published in [10]. Here, the integration framework acts as the glue model between a subset of the different domains typically involved in the design of process plants. Furthermore, it represents the basis for the automatic generation process for interlock code for PLCs. Throughout the paper, an interlock is considered as a piece of binary logic which locks an associated hardware component at a specific state if activated. For instance, an inlet valve of a tank should be locked to its closed state whenever the maximum level of the tank is reached in order to prevent an

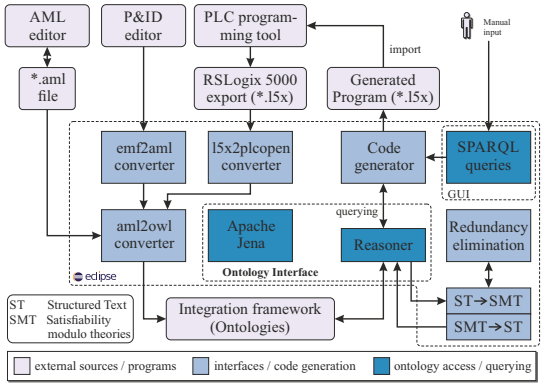



Fig. 1. Overview on the proposed automatic interlock code generation process. The required engineering data is extracted either from AutomationML files or out of an implemented P&ID editor and mapped to the ontology-based integration framework. Based on the formalized engineering data, the interlock generation process automatically generates interlock PLC code for the PLC programming tool RSLogix 5000 based on its XML interface.

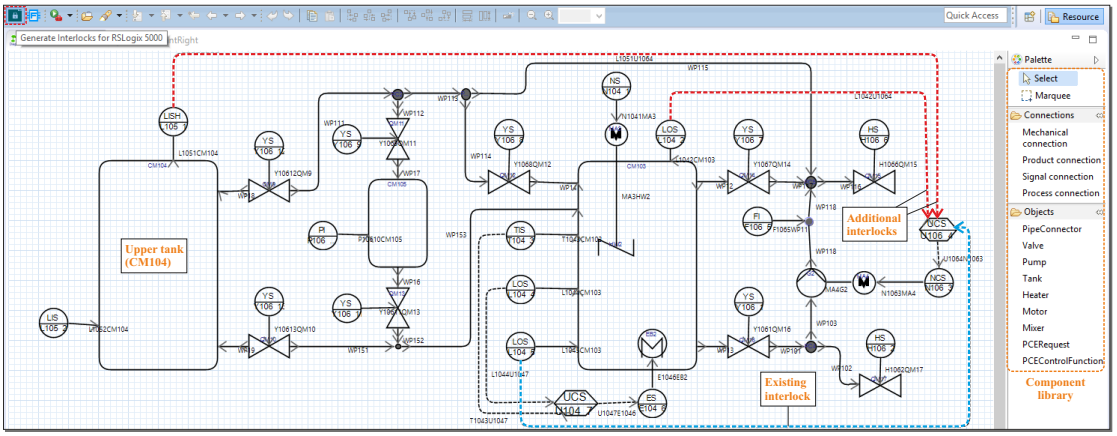



Fig. 2. Developed editor for piping and instrumentation diagrams (P&IDs). The P&ID can be developed by applying components from the library (right side). The interlocks defined for the considered use case for pump G2 are highlighted, whereas the interlock originating from the level sensor L104_5 (blue dashed line) has also been defined in the existing PLC program.

overflow. The individual components of the proposed approach are briefly introduced in the following subsections.

### A. Knowledge from Engineering Processes

In the process domain, the core *process control engineering* (PCE) document is represented by the P&ID as depicted in Fig. 2. In general, the P&ID contains information about the plant components, their interconnection for example via pipes, as well as first information about automation aspects like interlocks (dashed lines in Fig. 2). To access the engineering data contained in the P&ID, an editor has been developed based on the Eclipse graphical tooling infrastructure Graphiti [11], which is also shown in Fig. 2. It enables the component-based engineering of P&IDs, whereas the underlying editor model is given by the Ecore model as depicted in Fig. 3. The P&ID editor has been developed for several reasons. One reason is that this enables full access to the internal tool model and the semantics of the individual components used for drawing a specific P&ID. Additionally, the design of P&IDs can be restricted to specific rules as suggested by Schüller and Epple [12] and focused on standardized concepts as, for instance, defined in the international standard IEC 62424.

Referring to Fig. 3, the main concepts for interlock modeling within P&IDs are `PCEControlFunction` to model multiple interlocks acting, for example, on a single hardware component, `PCERequest` for representing interfaces between the hardware component and the superior process control system as well as signal interface for interlocks, and `SignalConnection` as the interlock connection between involved components. To improve semantics, the signal source and sink elements of a `SignalConnection` are an element of the PandIX interface elements (cf. Schüller and Epple [12, Table 1]). This enables also the explicit modeling of logical negated signals within the P&ID, which is required for the interlock generation process. In order to cope with even more complex interlock logic, the `PCEControlFunction` can be applied to group different interlock signals acting on one component and the logical relation between the individual signals can be defined as an attribute of the concept as IEC 61131-3 *Structured Text* (ST). Additionally, to distinguish between a binary or analog signal source, the processing function convention standardized in the international standard IEC 62424 [13] is consistently applied (*I* for analog values, *O* for binary values). Therefore, each `PCERequest` representing a component providing an analog signal and generating a binary signal has to define the `switchlimit` element.

### B. Semantic Integration

Since all the basic information for the proposed automatic interlock generation process is contained in the individual engineering documents, the main challenge is to gather all the required information and bring them together in an integrated and machine-readable plant model. Since some of the engineering tools applied in the process engineering domain support the export of the internal plans in the XML (*eXtensible Markup Language*) format, the goal is to bring together the individual exported XML files. One of the commonly applied strategies to semantically integrate individual XML-

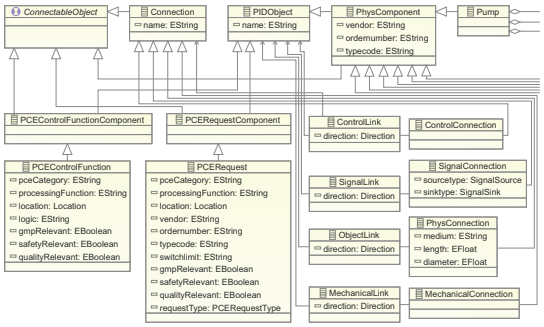



Fig. 3. Ecore model of the piping and instrumentation diagram editor. This internal tool model is mapped to the CAEX model of AutomationML.

Post-print version of the article: Michael Steinegger, Martin Melik-Merkumians, and Georg Schitter, "Ontology-based framework for the generation of interlock code with redundancy elimination," *2017 IEEE 22st International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2017. DOI: 10.1109/ETFA.2017.8247710

Listing 1. SWRL RULE TO ASSIGN A CONTROL FUNCTION BLOCK.

```
ProcessControlElement(?processControlElement) ^                1
name(?processControlElement, ?name) ^                          2
Block(?controlBlock) ^                                         3
instanceName(?controlBlock, ?instanceName) ^                   4
swrlb:equal(?instanceName, ?name)                              5
      -> hasControlFb(?processControlElement,                  6
          ?controlBlock)                                       7
```

based sources is to map them onto ontologies as, for example, shown by Cruz et al. [14]. An ontology-based integration of the individual aspects within the domain-specific engineering documents enables the reasoning about desired information with a standard interface. According to Fig. 1, the internal tool model of the developed P&ID editor is converted to an instance of the PandIX model [12] defined upon the CAEX ontology as submodel of AutomationML [10]. This enables the post-processing of the converted P&ID model with an additional tool supporting the AutomationML interface or the mapping of an individually designed AutomationML file.

For the PLC interlock code generation process, an additional mapping for the Rockwell RSLogix 5000 XML-based import and export interface was developed. It converts the required sub-set of the tool-specific data format to the PLCopen XML which is in the end converted to the corresponding ontology (see Fig. 1). This enables the manipulation of the existing PLC code within the ontologies. The manipulated PLC program is then mapped back to the XML representation after the generation process. In order to link both instances of the ontologies, containing the engineered P&ID on the one hand and the PLC program on the other hand, SWRL (*semantic web rule language*) rules are applied. Listing 1 contains a SWRL rule which automatically adds a reference between a process control element (line 1 and 2), representing a hardware device and denoted as *PhysComponent* in Fig. 3, and the controlling *function block* (FB) in the PLC program (line 3 and 4). In this case, the identification is based on the instance name of the FB and the name of the process control element (Line 5). Thus, the interlock logic specified in the P&ID for a specific hardware component can be (automatically) added to the corresponding control FB identified by the SWRL rule.

## III. GENERATION OF INTERLOCK CODE

As described in the previous section, the engineering data of the P&ID editor as well as the existing PLC program are mapped to the AutomationML-based ontologies of the integration framework. Thus, the ontologies already contain the information of existing interlocks as well as interlocks which are designed within the P&ID editor and should be added to a control FB of a physical component. However, the mapping between the PCE request tag and the corresponding variable name of an input or output is given by a manually engineered mapping table, since this information is usually contained in the input/output configurations for PLCs.

### A. Interlock Redundancy Elimination

In order to describe the redundancy elimination process, two logical expressions L1 (existing interlock) and L2 (interlock to be added) are considered, which are defined as follows:

$$\mathrm{L1} := (\mathtt{A}\ \textbf{AND}\ \mathtt{B})\ \textbf{OR}\ (\mathtt{B}\ \textbf{AND}\ \mathtt{C})\ , \tag{1}$$

$$\mathrm{L2} := (\textbf{NOT}\ \mathtt{A})\ \textbf{AND}\ \mathtt{C}\ . \tag{2}$$

In the current implementation, the interlock L2 to be added would be stored as plain ST expression in the ontology within the corresponding PCE control function instance (*logic* data element), while the existing interlock L1 is represented by a FB network as depicted in Fig. 5. Since different IEC 61131-3 languages can be used to describe the interlock logic, the specified language SMT-Lib [15] (satisfiability modulo theories) for the solver Z3 is applied as intermediate language. Therefore, two different grammars for parsing ST code as well as SMT-Lib expressions have been developed. To eliminate redundant logic, the workflow is as follows:

- In the first step, the logical FB network connected to the interlock input of the control FB, representing the existing logic L2, and given by the `Block` instances in the PLCopen TC6 ontology is retrieved via recursive SPARQL queries. Afterward, the query results are processed in order to translate the interlock logic encoded into the FB network stored in the ontology to an IEC 61131-3 compliant ST expression.
- In a second step, the interlock logic L2, which should be added to the PLC program, is retrieved from ontology instances generated from the P&ID editor. Therefore, each interlock element in the P&ID is processed individually and, in case of a instantiated `PCEControlFunction` combined with an logical *OR* expression. Since the implementation of the signal connections follows the semantics of the PandIX model, the logical effect on the connected component can be determined (e.g., sink type NLOCK represents a logical negation of the interlock signal of the source). Additionally, the type of a `PCERequest` instance can be determined by the applied processing function convention according to the international standard IEC 62424. The result of this step is a logical interlock expression given in ST.
- After retrieving both logical expressions, the existing as well as the interlock logic to be added, the resulting expression L := L1 **OR** L2 is translated to the Z3 input language SMT-Lib. This is achieved by traveling the parse tree given by the implemented ST parser and translating the entire ST expression step-by-step to the SMT-Lib language, while automatically defining the identifiers and their datatypes in SMT-Lib. Here, the specification of the basic datatype can be supported by the knowledge from the P&ID editor stored in the ontologies, since therein it is defined whether a *PCERequest* is an analog or binary one (see Section II-A).
- Finally, the resulting SMT expression is analyzed for logical redundancies by applying an extended simplify tactic within the Z3 solver. This, even though it is computationally expensive, results in a redundancy-free logical SMT-Lib expression. For the example given above, the resulting expression is given by

  $$\mathrm{L}_* := (\mathtt{A}\ \textbf{AND}\ \mathtt{B})\ \textbf{OR}\ ((\textbf{NOT}\ \mathtt{A})\ \textbf{AND}\ \mathtt{C})\ . \tag{3}$$

  Applying the standard simplify tactic would only remove redundant sub-strings in the logical expressions and thus the final result would be $\mathrm{L}_* := \mathrm{L}$, which still contains logical redundancy. With the applied Z3 tactic it is guaranteed that all redundancies are recognized and eliminated, since it works on the logical level.

With the described procedure, the redundancy-free interlock logic is retrieved and can be added as PLC code afterward.

Post-print version of the article: Michael Steinegger, Martin Melik-Merkumians, and Georg Schitter, "Ontology-based framework for the generation of interlock code with redundancy elimination," *2017 IEEE 22st International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2017. DOI: 10.1109/ETFA.2017.8247710

### B. Generation of the Interlocks

To add the redundancy-free interlock logic to a FB, the existing interlock logic is removed initially from the interlock input of the FB since the existing interlock logic is contained in the reduced set. This is achieved by automatically remove the instances of the relevant blocks in the PLCopen TC6 ontology as well as all interconnections (wires) between the removed FBs. The resulting SMT-lib expression representing the new interlock logic, which should be added to the PLC program, is converted to an instance of a FB network in the PLCopen TC6 ontology. To achieve this transformation, the SMT (*satisfiability modulo theories*) parse tree is converted stepwise to the corresponding logical FB network representation by a bottom-up approach. Finally, the entire generated and adapted representation of the PLC program contained in the PLCopen TC6 ontology is transformed to the XML representation required for importing the program into RSLogix 5000.

## IV. Evaluation

The proposed automatic generation approach for interlock code is evaluated by applying it to a laboratory process plant.

### A. Description of the Target System

The target system is a Festo Compact Workstation consisting of two tanks, which are interconnected via different pipes (see Fig. 4). The flow between the two tanks can be controlled via one pump and different switching valves. Furthermore, the process plant contains a heater to heat up the contained liquid, an agitator in the lower tank, as well as several sensors like level or flow sensors for process feedback. The plant itself is controlled by a Rockwell CompactLogix 1769-L35E programmed via the PLC programming tool RSLogix 5000.

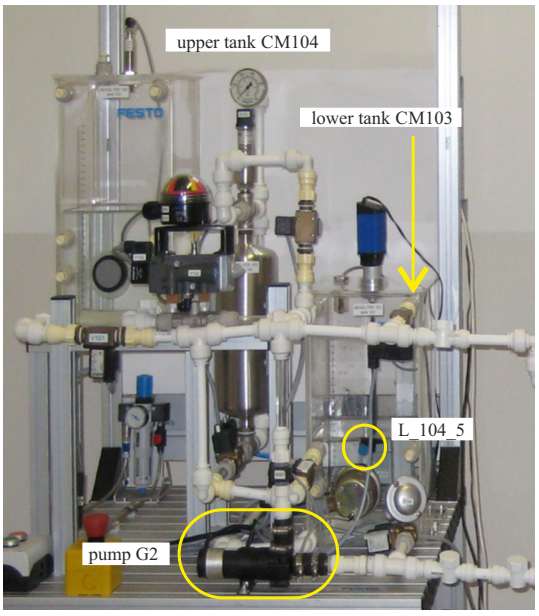
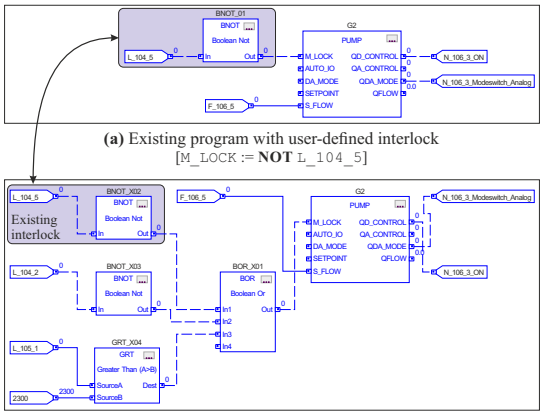



Fig. 4. Laboratory process plant of Festo applied for evaluation.




**(a)** Existing program with user-defined interlock
[M_LOCK := **NOT** L_104_5]




**(b)** Generated program with added interlock
[M_LOCK := (**NOT** L_104_5) **OR** (**NOT** L_104_2) **OR** (L_105_1 > 2300)]

Fig. 5. PLC program snippet (a) before and (b) after the generation process. It can be seen that the already existing interlock (**NOT** L_104_5) at the interlock input *M_LOCK* is not additionally added due to the proposed redundancy elimination algorithm.

### B. Use Case: Adding Interlock to Existing PLC Code

The results of the code generation process is described based on the control of the pump G2 as depicted in Fig. 2. For simplicity reasons, it is assumed that the process recipe in form of a *sequential function chart* (SFC) as well as the component control has already been defined by the user within the PLC program. Furthermore, as depicted in Fig. 5, the controlling FB for the pump is identified by the corresponding IEC 81346 [16] component designation. However, this does not represent a limitation for the underlying approach since it could be either assumed that the instance name of the corresponding FB equals the IEC 81346 tag or the mapping is given by an additional user-defined mapping table that has been previously be defined by the user. In the current implementation, it was assumed that the FB instance name equals the IEC 81346 tag applied in the P&ID for simplicity reasons.

Considering Fig. 5(a), it can be seen that there already exists a user-defined interlock logic for the controlling FB of the pump G2, which prevents the pump to be switched on in case there is not enough liquid in the lower tank (float switch L104_5 is not activated). The existing interlock L1, represented as a simple FB network in Fig. 5(a), can be translated to an ST expresion and written as

$$\text{L1} := \textbf{NOT}\ \texttt{L\_104\_5}\ . \tag{4}$$

This existing interlock could have been added by the PLC programming engineer as an a-priori known interlock function. Since all required concepts of the RSLogix 5000 program is mapped to the developed PLCopen TC6 ontology, the information about the existing interlock as well as the control FB corresponding to the hardware component is available. Additionally, all interconnections of the instantiated FBs are contained in the ontology as instances of the Wire concept which provides a formalized model of the entire PLC program and, as a result, of the interlock logic in form of a FB network. On the other side, additional interlock logic can be defined

Post-print version of the article: Michael Steinegger, Martin Melik-Merkumians, and Georg Schitter, "Ontology-based framework for the generation of interlock code with redundancy elimination," *2017 IEEE 22st International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2017. DOI: 10.1109/ETFA.2017.8247710

within the P&ID as, for instance, depicted in Fig. 2. This interlock logic is combined with the desired interlock logic engineered in the P&ID diagram given by

$$\text{L2} := (\textbf{NOT}\ \texttt{L\_104\_5})\ \textbf{OR}\ (\textbf{NOT}\ \texttt{L\_104\_2})\ \textbf{OR}\ (\texttt{L\_105\_1} > 2300)\ . \tag{5}$$

Afterward, the redundancy-free interlock logic is retrieved via the algorithm described in Sec. III-A.

The resulting and automatically added interlock logic for the pump control FB is depicted on the bottom of Fig. 5. It can be seen that the redundant expression (**NOT** `L_104_5`) is eliminated in the combined expression L during the generation process and only added once to the interlock input of the pump control FB. However, the user-defined FB network representing the manually implemented interlock have been removed and a new instance of the negation FB was added, which can be seen by the different instance names of the FBs in Fig. 5. Nevertheless, the current implementation of the proposed framework represents the foundation for the co-existence of user-defined and automatically added interlock PLC code.

## V. Conclusion

This paper presents a knowledge-based approach to automatically generate interlock PLC code solely based on the information contained in typical engineering documents in the process domain, like a P&ID. In order to achieve the integration of automatically generated PLC code into existing code, different ontologies together with mappings from XML sources onto the ontologies have been developed. Here, the ontologies hold the information of the desired interlocks contained within a P&ID and the already existing interlock logic given by a PLC program engineered in RSLogix 5000. The elimination of redundant code fragments is achieved by applying the theorem prover Z3 to the combined interlock logic previously translated to the solver language SMT.

It is shown, that the generation of interlock code and integrating the generated code into existing PLC programs is possible with the proposed framework. However, the future work is concerned with the generation of redundancy-free code while avoiding the entire removal of existing interlock code. This would enable the entire co-existence of user-defined and automatically generated code. Furthermore, the current implementation only deals with existing interlock logic code represented by a FB network composed out of standard logical FBs. The scalability issue of how to deal with even more complex structures and retrieving the logical expression from them is also part of the future work.

## Acknowledgments

Financial support by Festo AG (Esslingen, Germany) is gratefully acknowledged.

## References

[1] M. Steinegger and A. Zoitl, "Automated code generation for programmable logic controllers based on knowledge acquisition from engineering artifacts: Concept and case study," in *IEEE Int. Conf. on Emerging Technologies and Factory Automation*, 2012, pp. 1–8.

[2] M. Felleisen, *Prozessleittechnik für die Verfahrensindustrie*. Oldenburg Verlag, 2001.

[3] T. Moser and S. Biffl, "Semantic integration of software and systems engineering environments," *IEEE Transactions on Systems, Man, and Cybernetics – Part C: Applications and Reviews*, vol. 42, no. 1, pp. 38–50, 2012.

[4] F. Anhäuser, H. Richert, and H. Temmen, "Degussa PlantXML - integrierter Planungsprozess mit flexiblen Bausteinen [german]," *atp*, vol. 46, no. 10, pp. 62–72, 2004.

[5] Int. Org. for Standardization, "*ISO 15926: Industrial automation systems and integration – Integration of life-cycle data for process plants including oil and gas production facilities*," 2007.

[6] S. Biffl, A. Schatten, and A. Zoitl, "Integration of heterogeneous engineering environments for the automation systems lifecycle," in *Proc. of the 7th IEEE Int. Conf. on Industrial Informatics*, 2009, pp. 576–581.

[7] E. Estévez, M. Marcos, and D. Orive, "Automatic generation of PLC automation projects from component-based models," *Int. Journal of Advanced Manufacturing Technology*, vol. 35, pp. 527–540, 2007.

[8] R. Drath, A. Fay, and T. Schmidberger, "Computer-aided design and implementation of interlock control code," in *Proc. of the IEEE Int. Conf. on Computer-Aided Control Systems Design*, 2006, pp. 2653–2658.

[9] L. de Moura and N. Bjørner, "Z3: An efficient SMT solver," in *Tools and Algorithms for the Construction and Analysis of Systems*, ser. Springer Lecture Notes in Computer Science. Springer Verlag, 2008, vol. 4963, pp. 337–340.

[10] M. Steinegger, M. Melik-Merkumians, J. Zajc, and G. Schitter, "Automatic generation of diagnostic handling code for decentralized PLC-based control architectures," in *IEEE Int. Conf. on Emerging Technologies and Factory Automation*, 2016, pp. 1–8.

[11] W. M., "Eclipse Graphiti – building graphical editors the easy way," https://projects.eclipse.org/projects/modeling.gmp.graphiti, 2011.

[12] A. Schüller and U. Epple, "PandIX – exchanging P&I diagram model data," in *IEEE Int. Conf. on Emerging Technologies and Factory Automation*, 2012.

[13] Int. Electrotechnical Commision, "*IEC 62424: Representation of process control engineering – Requests in P&I diagrams and data exchange between P&ID tools and PCE-CAE tools*," 2008.

[14] I. R. Cruz, H. Xiao, and F. Hsu, "An ontology-based framework for XML semantic integration," in *Int. Database Engineering and Applications Symposium*, 2004.

[15] C. Barrett, A. Stump, and C. Tinelli, "The smt-lib standard: Version 2.0," in *Int. Workshop on Satisfiability Modulo Theories*, 2010.

[16] Int. Electrotechnical Commision, "IEC 81346 – Industrial systems, installations and equipment and industrial products – Structuring principles and reference designations – Part 1: Basic rules and Part 2: Classification of objects and codes for classes," 2010.

Post-print version of the article: Michael Steinegger, Martin Melik-Merkumians, and Georg Schitter, "Ontology-based framework for the generation of interlock code with redundancy elimination," *2017 IEEE 22st International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2017. DOI: 10.1109/ETFA.2017.8247710