Available online at www.sciencedirect.com

ScienceDirect

IFAC PapersOnLine 54-1 (2021) 420–425

An FMEA-based Methodology for the Development of Control Software Reliable to Hardware Failures

Hussein David Tafur * **Giacomo Barbieri** *
Carlos Eduardo Pereira **

* *Department of Mechanical Engineering, Universidad de los Andes, Bogotá, Colombia (e.mail: {hd.tafur10,g.barbieri}@uniandes.edu.co)*
** *Universidade Federal do Rio Grande do Sul, Porto Alegre, Brazil (e.mail: cpereira@ufrgs.br)*

Abstract: In automation systems, a high number of faults is induced by hardware failures. Their control software can be utilized to mitigate this problem by making it detect and manage the different failure events that may occur in the system. However, control software design methodologies have mainly focused on the system nominal behavior, marginally consider the generation of software reliable to hardware failures. In response to this challenge, this paper presents a methodology for the development of reliable automation systems which integrates the following tools: (i) Failure Mode and Effect Analysis (FMEA): to identify the different failure modes, and the strategies for their detection and management; (ii) AutomationML: to model the hierarchy and interfaces of automation system's components; (iii) Virtual Commissioning and Fault Injection: to assess – before system deployment – the reliability of the control software in the presence of hardware failures. Through its application to a case study, it is demonstrated that the methodology enables the identification of failure modes, the elicitation of requirements for their detection and management, and the generation of control software reliable to the identified failure modes.

Keywords: FMEA, AutomationML, Control Software, Hardware Failure, Virtual Commissioning, Fault Injection.

1. INTRODUCTION

In today's automation systems, an increasing number of functionalities is being shifted from pure mechanics and electronics into software. In particular, the control software is being enhanced with functionalities for handling *failures in hardware components*. For instance, Kormann and Vogel-Heuser (2011) adopted it to detect and manage the different failure events that may occur in the system.

Concerning the development of control software, different approaches have been proposed spacing from models for abstracting its behavior towards design pattern for the deployment (Jack, 2010). *Control software design methodologies* have mainly focused on the software as a mean to satisfy the functional requirements of automation systems; see (Bonfe and Fantuzzi, 2001; Barbieri, 2016). Whereas, the literature lacks of approaches concerning the development of functionalities for the detection and management of hardware failures (Vogel-Heuser et al., 2015).

One of the reasons for this is the *complexity* of the problem. In fact, the inclusion of functionalities for the detection and management of hardware failures within the control software requires: (i) an interdisciplinary approach due to the intrinsic nature of the problem that merges hardware and software components, including electrical, mechanical and pneumatic devices among others (Vogel-Heuser et al., 2020); (ii) verification techniques able to check the behavior of the control software in response to hardware failures (Rösch and Vogel-Heuser, 2017). With the goal of bridging this gap, this paper proposes a methodology that combines a hardware failure analysis process, a modeling tool tailored for automation systems, and a verification method consisting of virtual commissioning and fault injection.

Since this work deals with hardware failures, a process must be selected for their identification and for the definition of strategies for their management. *Failure Mode and Effect Analysis* (FMEA) is the process of reviewing the maintainable items of a physical asset to identify potential failure modes, and their causes and effects (Carlson, 2012). Within this paper, FMEA is selected as hardware failure analysis process and is adopted to elicit requirements based on the recommended actions to detect failures and mitigate / eliminate their effects.

With the goal to gather and sort out the information needed for the FMEA analysis, it is fundamental to represent all the different views of the automation system and their interrelationships. For instance, it is estimated that about 50% of the total failures of automation systems occur in the interfaces between hardware components (Carlson, 2012). *AutomationML* (AML) is a modeling language that provides a hierarchical representation of automation systems, integrating mechanical, electrical and software views, and their interrelationships (Drath et al.,

10.1016/j.ifacol.2021.08.047

2008). Due to these features, AML is applied within this work as a tool to support the execution of the FMEA analysis.

Finally, a strategy must be utilized to verify the behavior of the control software in response to hardware failures. The integration of virtual commissioning and fault injection is applied in this paper to assess the behavior of the control software once the detection and management functionalities have been included. *Virtual Commissioning* (VC) is a verification strategy performed by interfacing a discrete-event behavior model of the physical plant to the hardware or emulated controller which contains the control software to be validated (Lee and Park, 2014). Whereas, *Fault Injection* consists in the system verification by forcefully making it enter failure states (Svenningsson et al., 2011). By injecting faults and analyzing the system reaction through the VC simulation, the reliability of the control software to hardware failures can be verified (Orive et al., 2019). For reliable, we mean a control software able to detect hardware failures, and able to mitigate or eliminate the effects of the detected failures.

In the literature, few works propose the utilization of tools applied within this paper to perform some of the activities involved in the development of reliable automation systems. For instance, modeling languages have been used to support the FMEA analysis (David et al., 2010; Scippacercola et al., 2015), the integration of simulation and fault injection has been adopted to check the reliability of control software to hardware failures (Rösch and Vogel-Heuser, 2017; Orive et al., 2019), etc. However, the *FMEA-based methodology* proposed in this work constitutes a complete approach to the problem since specifies all the different phases involved in the generation of control software reliable to hardware failures, starting from the definition of the necessary information towards the code deployment.

Given the above, the paper is structured as follow: section 2 illustrates the proposed FMEA-based methodology and section 3 applies it to a case study. Obtained results are discussed in section 4 and finally, section 5 presents the conclusions and sets the directions for future work.

2. FMEA-BASED METHODOLOGY

The methodology for the development of control software reliable to hardware failures is depicted in Figure 1. The methodology is meant to be applied to an already designed / operating automation system in which the control software must be enhanced with functionalities for the detection and management of hardware failures.

2.1 Information gathering

The objective of this phase is to gather all the information necessary for the FMEA analysis and for the subsequent system modifications. When this operation is implemented, it must be considered that (Kernschmidt et al., 2018): (i) systems are composed as assemblies of components; (ii) components have properties and interfaces; (iii) components of automation systems are characterized with mechanical, electrical and software views. Therefore,

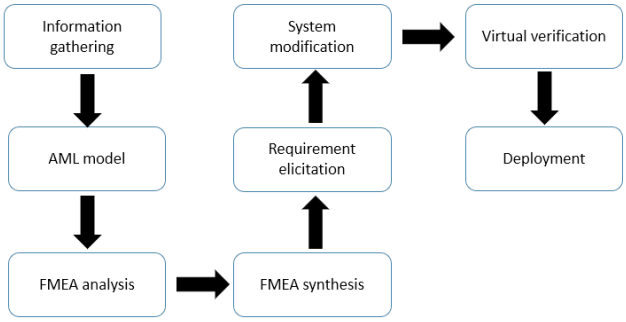

Fig. 1. Phases for the development of control software reliable to hardware failures.

the following subdivision is proposed for the information gathering:

- *Component*: (i) mechanical view: mechanical properties (e.g. dimensions, material, etc.) and mechanical interface (e.g. screws, pipes, etc.); (ii) electrical view: electrical interface (e.g. DC, AC, etc.): (iii) software view: software interface (e.g. digital, analog, etc.);
- *System*: (i) process view: P&ID diagram to indicate how the components interact for the implementation of the automated process; bill of materials to obtain the list of components utilized within the system; requirement list to detail the information concerning the system operation; (ii) mechanical view: CAD models to represent the spatial configuration of the components and how they are interfaced; (iii) electrical view: electrical drawings to indicate how the automated components are electrically assembled and supplied; (iv) software view: control code and eventual models that abstract it (e.g. state machine diagrams, etc.) to identify the implemented control logic and the interface variables.

2.2 AutomationML model

Once the information has been gathered, it must be *integrated and organized* to generate a complete representation of the system. This representation will be utilized for the implementation of the FMEA analysis. AutomationML enables the modeling of (Wimmer, 2018): (i) *Components* with their properties and interfaces; (ii) *Systems*: hierarchical view showing the constituting components and how they are interfaced both from a mechanical, electrical and software perspective. Therefore, the information gathered in section 2.1 is sorted out within this phase through the implementation of an AML model.

2.3 FMEA analysis

The FMEA process is generally utilized for identifying potential failure modes of maintainable items, and their causes and effects (Carlson, 2012). Whereas, the FMEA analysis proposed in this phase is mainly focused in defining: (i) *Failure modes*: the specific manner by which a failure occurs in the hardware of the system; (ii) *Effects*: the effects and severity of the identified failure modes; (iii) *Actions*: actions to mitigate or eliminate the failure effects. In fact, the information obtained from this analysis will be utilized to define strategies for the detection and

management of failures modes, and not to identify their causes and possible solutions to them – as in traditional FMEA processes.

The information within the *AML model* constitutes the basis for the implementation of the FMEA analysis. In fact, the AML model represents the interrelationships between the different views of the system components – mechanical, electrical and software – enabling the quantification of the effects of the identified failure modes. Furthermore, properties can be associated to components facilitating the identification of abnormal behaviors.

2.4 FMEA synthesis

This phase synthesizes the information generated with the FMEA analysis to define software strategies for detecting and managing the failure modes of the inspected hardware components. Figure 2 illustrates the operations implemented within this phase and their relationship with the FMEA analysis. These operations can be described with the following sequential steps: (i) *Failure Mode Set*: since different failure modes may be detected with the same method, this step consists in grouping failure modes based on the detection method; (ii) *Detection method*: starting from the generated effects, detection methods are defined by comparing the nominal and exceptional behavior of selected control variables. When performing this operation, it must be considered that the nominal behavior is dependent from the system operational condition; (iii) *Management strategy*: based on the severity of the generated effects, a strategy to mitigate or eliminate the effects of the considered failure modes is identified; e.g. alarm / warning activation, switch off of actuators, etc.

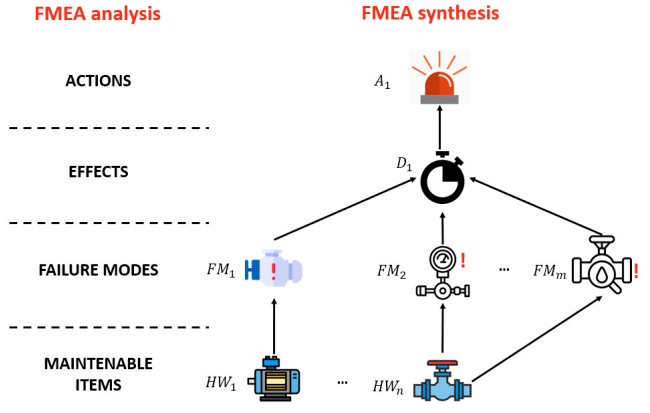

Fig. 2. Representation of the relationship between the FMEA analysis and the operations implemented within the FMEA synthesis. The following nomenclature is utilized: HW stands for hardware, FM for failure mode, D for detection method, and A for alarm.

2.5 Requirement elicitation

The results obtained in the previous phase are converted into requirements for the detection and management of hardware failures through the control software. For instance, it can be defined that the system must be able to monitor selected control variables and to implement specific actions based on the values acquired from the control variables. Different strategies can be utilized to formally specify the requirements (Fantechi et al., 1994); e.g. temporal logic, etc.

2.6 System modification

In this phase, the identified requirements must be fulfilled. *Sensors* are selected to monitor the necessary control variables and *actuators* for the implementation of the actions defined for the management of the hardware failures. These sensors and actuators may already be present in the original system or may be integrated to it. Furthermore, the *control software* is modified to include functionalities for the detection and management of the identified hardware failures by using the defined sensors and actuators.

2.7 Virtual verification

Once the modifications to the original system have been identified, these must be checked before the deployment. A virtual environment for the verification of the system reliability to hardware failures is proposed by integrating VC and fault injection. *Virtual commissioning* is implemented by developing a simulation model of the physical plant described at the level of sensors and actuators, and by connecting it to the hardware or emulated controller which contains the control software. Within the VC simulation, hardware failures are emulated with *fault injection*; i.e. hardware defects are simulated during runtime. By inspecting the VC simulation, the system reaction to hardware failures is verified.

2.8 Deployment

The control software is deployed once its modifications and the integration of the additional sensors and actuators have been verified through the described virtual environment.

3. CASE STUDY

In this section, the proposed methodology is validated by applying it to a case study. The selected case study is the *filtration unit* of a flexible test-bench for hydroponics (Barbieri, 2019). The filtration unit is depicted in Figure 3 and contains an inverse osmosis filter for reducing the water chloride and total suspended solid content. Digital level sensors are utilized to detect the state of the system, and a solenoid valve and a pump to respectively input tap water and provide the necessary pressure to make the filter properly work. The FMEA-based methodology proposed in section 2 was applied to integrate functionalities for the detection and management of hardware failures to the control software of the system.

3.1 Information gathering

In this phase, the information listed in section 2.1 was gathered. In particular, the *technical specifications* of the components provided information concerning their interfaces and the relevant properties for identifying abnormal behaviors. Examples of these properties are operative

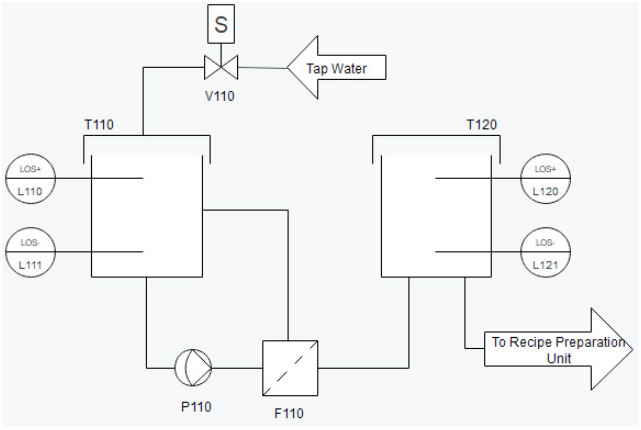

Fig. 3. P&ID diagram of the utilized water filtration unit.

pressure and flow rate, maximum tolerated current, etc. Furthermore, the different *domain-specific models* were collected; i.e. P&ID diagram, CAD model, electrical drawings and PLC code.

3.2 AutomationML model

The AML model was built in AutomationML Editor [1]. A class was generated for each component. In AML, a component is defined with custom *properties and ports* (viz. interfaces). In particular, key properties for the identification of abnormal behaviors were associated to the created components. Then, the components were instantiated to build the hierarchical representation of the system. Finally, the components were interfaced both from a mechanical, electrical and software perspective as shown in Figure 4. The CAD model, the electrical drawings and the PLC code respectively provided the information concerning the mechanical, electrical and software interfaces of the different components.

3.3 FMEA analysis

The FMEA process was implemented by following the phases of preparation, procedure and execution defined in (Carlson, 2012). *Failure modes* were identified by individually inspecting all the system components. Then, the domain-specific and the AML models were analyzed to define the *effects* and to quantify the *severity* of each failure mode. Finally, an *action* was recommended to mitigate / eliminate the effects of each failure mode. Part of the implemented FMEA analysis is represented in Figure 5.

3.4 FMEA synthesis

By analyzing the effects encountered with the FMEA analysis, a *detection method* was defined for each failure mode, and a nominal and exceptional behavior was identified for its detection. For instance, different failure modes can be detected based on the time taken to fill a tank as shown in Figure 6. Then, failure modes that presented the same detection method were grouped together into *failure mode sets*. Finally, a *management strategy* was defined by analyzing the severity of the generated effects.

[1] https://www.automationml.org/o.red.c/dateien.html?cat=1

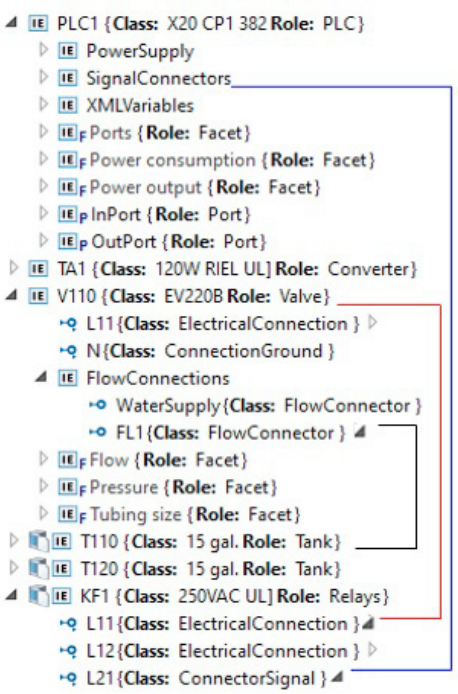

Fig. 4. AML model: hierarchical representation and interface of the V110 solenoid valve. A software interface is represented with a blue line, an electrical interface with a red line, and a mechanical interface with a black line.

Maintainable Item	Failure Mode	Failure Effect	Failure Severity	Action
V110 Solenoid Valve	Failure to open (FO)	T110 tank does not fill. L110 sensor remains false	3	Stop the system
	Failure to close (FC)	T110 tank overflows. L110 sensor remains true even if P110 pump is active	3	Stop the system
	Leakage (L)	Reduction in the system performance. L110 takes more time to become true	2	Inspect the system
L110 Level Sensor	False positive (FP)	T110 tank is assumed full of liquid when it is not. P110 pump may be damaged. L121 does not become true even if P110 pump is active	4	Stop the system
	False negative (FN)	T110 tank overflows. L110 sensor remains false	3	Stop the system

Fig. 5. FMEA analysis for the V110 solenoid valve and the L110 level sensor.

Failure Mode Set	Failure Modes	Detection			Mangagement strategy
		Method	Nominal behavior	Exceptional behavior	
1	V110 (FO) + L110 (FN)	Time to fill T110 tank	25s	30s	A110 Alarm and switch off all the actuators
2	V110 (L)	Time to fill T110 tank	25s	27,5s	W110 Warning

Fig. 6. FMEA synthesis for some of the failure modes of the V110 solenoid valve and the L110 level sensor.

3.5 Requirement elicitation

In this phase, the table generated within the FMEA synthesis process was converted into requirements. Even if different formal methods can be utilized to specify requirements, natural language was utilized since their formal specification was outside the scope of the paper. Two requirements were obtained for each failure mode set; i.e. one for the detection and one for the management. For

instance, the following requirements were elicited for the first failure mode set (see Figure 6): (i) *Detection*: "If L110 sensor is false, A110 alarm must trigger after 30s from the activation of V110 solenoid valve"; (ii) *Management*: "When A110 alarm is triggered, all the actuators must be switched off, and the operator must inspect the V110 solenoid valve and the L110 level sensor".

3.6 System modification

Based on the identified requirements, it was not necessary to integrate additional sensors and actuators to the original system. In fact, most of the failure modes were detected with timers, and most of the management strategies consisted in switching off all the actuators. Then, the *PLC software* was modified to integrate the conditions for detecting the failure modes, and to implement the actions identified within the management strategy. Furthermore, the system *HMI* (Human-Machine Interface) was enhanced with leds indicating the different alarms and warnings, and with pop-up windows suggesting to the operator the components to inspect; see Figure 7.

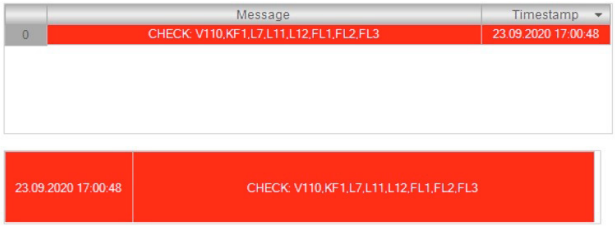

Fig. 7. Example of failure message implemented within the HMI. The suggestion of the components to inspect can be appreciated.

Finally, few failure modes did not generate consequences in the system functionality – at least in the short term. Some examples of these not critical effects are noises, vibrations, etc. In this case, we chose to insert their periodic inspection within the *system maintenance plan*, and not to integrate sensors and software functionalities for their detection. Therefore, it can be noticed that a 'side effect' of the proposed FMEA-based methodology is the refinement of the system maintenance plan.

3.7 Virtual verification

In VC, a simulation model of the physical plant described at the level of sensors and actuators must be built. Considering that a *model* constitutes an approximation of the physical system, the correct *abstraction* must be identified to properly reproduce the investigated properties (Lee and Sirjani, 2018). In the presented methodology, failures are manually injected through fault injection. Therefore, the model only needed to replicate the sequence of events that occur in the physical system; viz. its discrete-event behaviour. Whereas, the model did not need to perfectly mimic the continuous-time behaviour of the physical system. For instance, it was not necessary to model the chemical reactions that occur during the filtering of chloride. Since kinematic consists in the description of the motion without considering the efforts that cause it (Meriam and Kraige, 2012), a kinematic model was selected.

Given the above, a *kinematic model* of the filtration unit was implemented in Simulink [2]. Then, the VC simulation was performed by interfacing Simulink and CoDeSys Win Control V3 [3] (i.e. the PLC emulated controller) using the OPC protocol. Interrupters and gauges were inserted in the Simulink model to *inject* failures during runtime; see Figure 8. Finally, the system reaction to the injected failures was analyzed to evaluate its reliability.

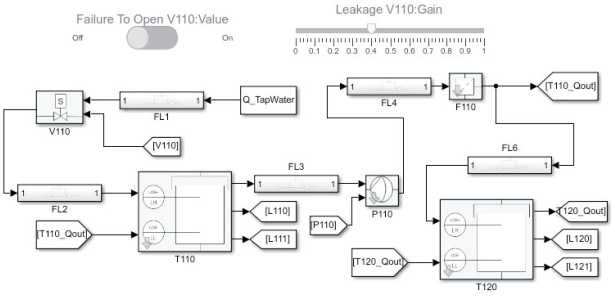

Fig. 8. Kinematic model and fault injection elements for the failure to open and leakage failure modes of V110 valve.

3.8 Deployment

The deployment of the control software in the PLC of the physical filtration unit was not included within the scope of this work and is left as future work.

4. RESULTS AND DISCUSSION

The following results were obtained by applying the proposed FMEA-based methodology to the case study of the water filtration unit:

- *FMEA and AML*: by representing all the system interfaces, the AML model facilitated the quantification of the failure effects and severity, and consequently the implementation of the FMEA analysis;
- *Requirements*: by following the FMEA synthesis process, the generation of requirements consisted in a conversion of the FMEA synthesis table into natural language. After the inclusion of the detection and management of hardware failures, the total number of requirements was approximately increased by 40%, showing the importance of considering these aspects within the control software;
- *Virtual verification*: test cases were generated for evaluating the reliability of the control software to the hardware failures. The integration of VC and fault injection enabled the simulation of all the identified failure modes and the evaluation of the system reaction to them;
- *Maintenance Plan*: a 'side effect' of the methodology was the refinement of the system maintenance plan by including the periodic inspection of not critical failure modes.

Finally, it must be considered that the methodology provided all these benefits when applied to a simple automation system. Its *scalability* to an industrial application

[2] `https://mathworks.com/products/simulink.html`
[3] `https://www.codesys.com/`

should be tested before certifying its ability to generate control software reliable to hardware failures.

5. CONCLUSION AND FUTURE WORK

Considering the high percentage of faults caused by *hardware failures*, the inclusion of functionalities for their detection and management within the control software can bring several benefits in terms of reliability and performance of automation systems.

In this context, the objective of this research work was to explore the role of FMEA, AutomationML, virtual commissioning and fault injection for the generation of control software reliable to hardware failures. By integrating these tools, an *FMEA-based methodology* was proposed and applied to a case study of a water filtration unit to be validated. The methodology resulted promising since its application enabled the identification of failure modes, the elicitation of requirements for their detection and management, and the generation of a control software reliable to the identified failure modes.

Notably, the proposed approach constitutes a preliminary concept that in future should be better refined and validated. Some future works identified are:

- *Trigger failure mode*: different failure modes are detected with the same method. The integration of variables from different sensors should be investigated for the identification of the failure mode responsible for the alarm generation;
- *Root alarm*: in industrial automation, it is common the scenario in which different alarms trigger after the occurrence of a certain abnormal behavior. Strategies for the identification of the root alarm (i.e. the one related to the root cause) should be included in the methodology;
- *Further validations*: the methodology should be deployed to be further validated. Furthermore, it should be applied to an industrial case study to certify its ability to generate control software reliable to hardware failures.

REFERENCES

Barbieri, G. (2016). Platform-based design: methodology refinement and application to cyber-physical production systems. *PhD thesis*.

Barbieri, G. (2019). A small-scale flexible test bench for the investigation of fertigation strategies in soilless culture. *International Journal of Agricultural and Biosystems Engineering*, 13(1), 5–9.

Bonfe, M. and Fantuzzi, C. (2001). Object-oriented approach to plc software design for a manufacture machinery using iec 61131-3 norm languages. In *2001 IEEE/ASME International Conference on Advanced Intelligent Mechatronics. Proceedings (Cat. No. 01TH8556)*, volume 2, 787–792. IEEE.

Carlson, C. (2012). *Effective FMEAs: Achieving safe, reliable, and economical products and processes using failure mode and effects analysis*, volume 1. John Wiley & Sons.

David, P., Idasiak, V., and Kratz, F. (2010). Reliability study of complex physical systems using sysml. *Reliability Engineering & System Safety*, 95(4), 431–450.

Drath, R., Luder, A., Peschke, J., and Hundt, L. (2008). Automationml-the glue for seamless automation engineering. In *2008 IEEE International Conference on Emerging Technologies and Factory Automation*, 616–623. IEEE.

Fantechi, A., Gnesi, S., Ristori, G., Carenini, M., Vanocchi, M., and Moreschini, P. (1994). Assisting requirement formalization by means of natural language translation. *Formal Methods in System Design*, 4(3), 243–263.

Jack, H. (2010). *Automating manufacturing systems with PLCs*. Lulu. com.

Kernschmidt, K., Feldmann, S., and Vogel-Heuser, B. (2018). A model-based framework for increasing the interdisciplinary design of mechatronic production systems. *Journal of Engineering Design*, 29(11), 617–643.

Kormann, B. and Vogel-Heuser, B. (2011). Automated test case generation approach for plc control software exception handling using fault injection. In *IECON 2011-37th Annual Conference of the IEEE Industrial Electronics Society*, 365–372. IEEE.

Lee, C.G. and Park, S.C. (2014). Survey on the virtual commissioning of manufacturing systems. *Journal of Computational Design and Engineering*, 1(3), 213–222.

Lee, E.A. and Sirjani, M. (2018). What good are models? In *International Conference on Formal Aspects of Component Software*, 3–31. Springer.

Meriam, J.L. and Kraige, L.G. (2012). *Engineering mechanics: dynamics*, volume 2. John Wiley & Sons.

Orive, D., Iriondo, N., Burgos, A., Saráchaga, I., Álvarez, M.L., and Marcos, M. (2019). Fault injection in digital twin as a means to test the response to process faults at virtual commissioning. In *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, 1230–1234. IEEE.

Rösch, S. and Vogel-Heuser, B. (2017). A light-weight fault injection approach to test automated production system plc software in industrial practice. *Control Engineering Practice*, 58, 12–23.

Scippacercola, F., Pietrantuono, R., Russo, S., and Silva, N.P. (2015). Sysml-based and prolog-supported fmea. In *2015 IEEE international symposium on software reliability engineering workshops (ISSREW)*, 174–181. IEEE.

Svenningsson, R., Eriksson, H., Vinter, J., and Törngren, M. (2011). Generic fault modelling for fault injection. In Springer (ed.), *Formal methods for components and objects*, 287–296.

Vogel-Heuser, B., Böhm, M., Brodeck, F., Kugler, K., Maasen, S., Pantförder, D., Zou, M., Buchholz, J., Bauer, H., Brandl, F., et al. (2020). Interdisciplinary engineering of cyber-physical production systems: highlighting the benefits of a combined interdisciplinary modelling approach on the basis of an industrial case. *Design Science*, 6.

Vogel-Heuser, B., Fay, A., Schaefer, I., and Tichy, M. (2015). Evolution of software in automated production systems: Challenges and research directions. *Journal of Systems and Software*, 110, 54–84.

Wimmer, M. (2018). Multi-level modeling in the wild with automationml. In *Multi Level Modeling Workshop*.