Integrated Safety and Security by Design in the IT/OT Convergence of Industrial Systems: A Graph-Based Approach

Amirali Amiri
Institute of Computer Engineering
TU Wien, Vienna, Austria
amirali.amiri@tuwien.ac.at

Gernot Steindl
Institute of Computer Engineering
TU Wien, Vienna, Austria
gernot.steindl@tuwien.ac.at

Ian Gorton
Khoury College of Computer Sciences
Northeastern University, Seattle, USA
i.gorton@northeastern.edu

Siegfried Hollerer
Institute of Computer Engineering
TU Wien, Vienna, Austria
siegfried.hollerer@tuwien.ac.at

Wolfgang Kastner
Institute of Computer Engineering
TU Wien, Vienna, Austria
wolfgang.kastner@tuwien.ac.at

Thilo Sauter
Inst. of Computer Technology, TU Wien
Integr. Sensor Syst., Danube Univ. Krems
thilo.sauter@tuwien.ac.at

***Abstract*—The convergence of Information Technology (IT) and Operational Technology (OT) in Industry 4.0 poses fresh challenges, demanding innovative strategies to ensure the safe execution of production processes. With the increasing significance of production system integrity, any security breaches can lead to severe consequences like production downtime, equipment damage, or human harm. Our prior research on Austrian industrial automation stakeholders highlighted the necessity for a cost-effective, all-encompassing approach to the integrated safety and security. We introduced an extensive ontology for safety, security, and operational requirements in IT/OT convergence. This paper presents an approach of Model-Based Systems Engineering (MBSE) for the integrated safety and security by design of industrial systems. We employ the Systems Modeling Language (SysML) 2.0 for precise modeling. We define metadata information that are used as tags for SysML 2.0 model instances. Afterwards, we create a graph-based model of the system. These graphs are used to validate safety and security standards and requirements. Finally, we automatically generate artifacts, such as code or documentation, which adhere to the standards. Our approach is extensible and supports reusability already after covering two standards. Having provided support for the standards IEC 62443-3-3 and IEC 61508, we reuse our approach to validate the standard ISO 13850:2015.**

***Index Terms*—Safety and Security by Design, Industry 4.0, MBSE, Standard Conformance, IT/OT Convergence, SysML 2.0, Automatic Artifact Generation, Graph-Based Validation**

I. Introduction

Ensuring security in production systems is vital for safety, reliability, and availability [5]. However, the convergence of IT and OT has created new vulnerabilities, exposing production systems to cyber-attacks [2]. Attacked IT infrastructure may lead to safety issues, system failures, and equipment damage. For instance, exploiting IT vulnerabilities can target a safety instrumented system, resulting in its failure to respond when necessary, or executing safety functions at the wrong time. This can lead to injuries, damage to the production facility, or intentional triggering of safety functions, causing operational stoppages and economic losses [13].

Monitoring the state of production systems and detecting unexpected attacks pose a growing challenge. The complexity arises from the diverse resources, components and the isolated design of safety and security in production systems [5]. Additionally, different perspectives of engineering experts widen the gap in safety and security coverage, complicating the analysis. Although various approaches exist [11], [12], [14], there is a demand for a standardized, flexible framework, and generic tools that seamlessly integrate safety and security in the context of industrial systems [14].

Previously [6], we examined how stakeholders in Austrian industrial automation firms handle security and safety risks to prevent unfavorable outcomes. The analysis covered product or component vendors, integrators, and asset owners of industrial systems, focusing on safe and secure infrastructures, system architectures, and risk management. Moreover, we identified inter-dependencies between safety and security requirements.

Our findings indicated that the industrial stakeholders under examination expressed a consensus that achieving an economically feasible, comprehensive approach to safety and security by design was an important objective. Thus, we set out to answer the following research question:

How can we automatically validate safety and security requirements according to related standards, e.g., IEC 62443[1], IEC 61508[2], and ISO 13850:2015[3] for industrial systems?

In [7], we introduced a comprehensive ontology for safety, security and operation requirements in IT/OT convergence of

Supported by #SafeSecLab Research Lab for Safety and Security in Industry, a research cooperation between TU Wien and TÜV AUSTRIA.

[1]https://www.iec.ch/blog/understanding-iec-62443
[2]https://webstore.iec.ch/publication/5515
[3]https://www.iso.org/standard/59970.html

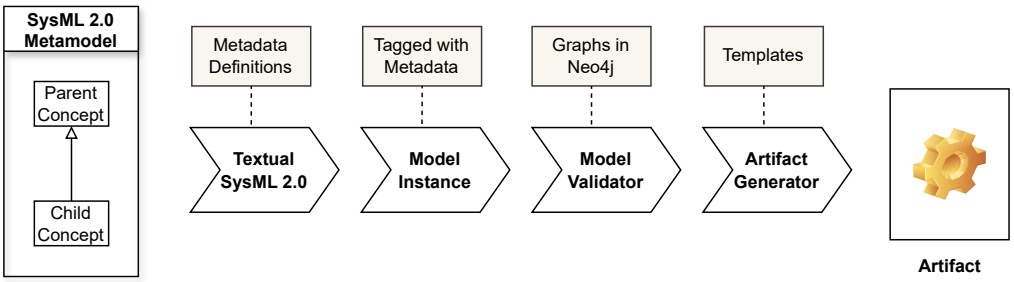

Fig. 1: The Workflow of Model-Based Systems Engineering

industrial systems. We identified eight domains to achieve a holistic view on safety and security, i.e., OT domain-specific model, operations and quality, hazard identification, threat identification, safety functions and requirements, security controls and requirements, risk evaluation and prioritization, and risk treatment. In this paper, we build on top of previous works and introduce a Model-Based Systems Engineering (MBSE) approach [16] to facilitate the generation of artifacts regarding the integrated safety and security of industrial applications.

We introduce metadata that describe industrial systems with integrated safety and security as a design-time requirement. Our proposed approach adheres to the following steps. Firstly, we create tagged model instances based on our metadata using the SysML 2.0 specification[4]. Secondly, we verify requirements and validate our model instances. Finally, we generate artifacts informing about the requirements validation.

II. Related Work

De Saqui-Sannes et al. [4] explore the evolution of Systems Engineering towards MBSE, which shifts from document-centric approaches to using models for system development. While existing literature discusses the benefits and limitations of MBSE, this paper focuses on providing industry practitioners with decision criteria for selecting MBSE languages, tools, and methods. Vogel-Heuser et al. [18] present a SysML-based approach for MBSE of manufacturing automation software projects. SysML is adapted to create SysML-AT, a specialized language profile covering functional and non-functional requirements, software applications, and hardware properties. This approach supports automated software generation for runtime environments complying with IEC 61131-3.

Japs [8] discusses the importance of interdisciplinary cooperation in developing Cyber-Physical Systems (CPS) like autonomous vehicles, emphasizing the need for system understanding among stakeholders to identify security threats and safety hazards during requirements engineering. The paper emphasizes that a lack of integrative consideration of these threats and hazards can lead to compromised safety compliance and increased development costs. Quamara et al. [15] study the crucial need to integrate safety and security considerations early in the design of critical systems like CPS, employing MBSE. The contribution shows the complexity in translating high-level mission-centric specifications into detailed architectures, especially for engineers less experienced in safety and security. A multi-layered design approach is proposed, combining techniques such as MBSE and formal methods.

Ahlbrecht and Bertram [1] consider the need for new by-wire system architectures, especially for segments like autonomous driving, which pose significant safety challenges not typically encountered in the automotive industry. To address this, a concept is proposed to incorporate safety considerations into early architecture selection using a safety trade-off approach. This involves employing system-theoretic process analysis on a SysML model developed through model-based systems engineering. Japs and Anacker [9] emphasize the interdisciplinary collaboration needed for the development of CPS, such as autonomous vehicles, due to their intelligent and networked nature. They highlight the importance of system understanding among stakeholders to identify and address security threats and safety hazards in the early engineering phases. Failure to do so can lead to increased costs in production and potential compromises in CPS safety compliance.

Kharatyan et al. [10] discuss the increasing integration of information and communication technologies in technical systems as digitization progresses, presenting both value potentials (e.g., autonomous driving) and challenges in system development. To address these challenges, MBSE approaches are employed to manage the growing complexity and networking of products. However, ensuring the reliability of future systems requires early consideration of security and safety aspects. Subramanian and Zalewski [17] discuss the importance of safety and security considerations in CPS, which are crucial for controlling critical infrastructures. These systems combine computer-based and network-based technologies for monitoring and controlling physical processes. While it is widely acknowledged that safety and security should be integrated into system design at the architectural level, there is a lack of techniques that address both aspects simultaneously.

III. Approach Overview

Model-Based Systems Engineering Figure 1 shows the workflow of our approach [16]. We define our *metadata* (based on our previous ontological concepts [7]) to describe the industrial systems that must adhere to safety and security guidelines. An example guideline is the IEC 62443[1], which is a series of standards concerning the cybersecurity of industrial

[4]https://www.omg.org/spec/SysML/2.0/Beta1

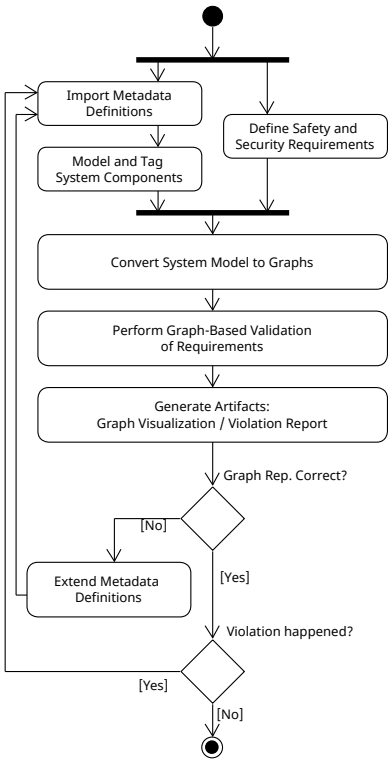

Fig. 2: High-Level Activities of Our Approach

automation and control systems. We define metadata in SysML 2.0, which are used as tags when creating a *Model Instance*. We convert these instances into graphs and store them in a Neo4j database[5]. A *Model Validator* checks the defined safety and security requirements by traversing the graphs. The model instances are then passed to an *Artifact Generator* to generate artifacts, e.g., code, automatic test cases, or documentation, which are traced back to the requirements. These artifacts are based on predefined templates, e.g., graph visualizations.

Figure 2 shows the high-level activities of our approach, which are detailed in the next section. The only extra step that system architects must perform to use our approach is to tag system components using our pre-defined metadata definitions in SysML 2.0. With safety and security requirements defined, our approach converts the system model automatically to graph representations and performs graph-based validation of requirements. We generate a graph visualization and inform the architect or system designer of the violating system parts.

IV. Approach Details

Metadata Our metadata for safety and security of industrial applications are shown in Figure 3. A *System under Consideration* consists of at least one *Asset* that can be *Information*, *Communication*, *Software*, *Hardware*, or a *User* with different *Roles*. A *Requirement* models safety or security requirement that relates to assets and *Risks*. A *Counter Measure* is an action to be performed to reduce the effects of risks. A *Safety Counter Measure* is taken against *Hazards* and *Security Counter Measure* against the *Threats*. *Test Cases* assess the effectiveness of counter measures against risks and fulfillment of requirements. A system can also consist of *Physical Sites* and *Zones*, e.g., security zones in a smart factory.

We have defined specific counter measures that we use for requirements checking in the next sections, i.e., *Stop Function* and *Authentication* with an attribute *ByPassSafety*. This attribute plays an important role in the interplay of safety and security so that a safety function, e.g., a stop button, is not blocked by security measures, such as authentication. This concept is discussed further in the next sections. Note that system architects can easily extend our metadata, e.g., by adding more counter measures.

SysML 2.0 Metadata Definitions We provide metadata definitions that must be imported in a SysML 2.0 system model. Listing 1 shows an excerpt of our metadata.

```
package ReqCheckerMeta {
    enum def UserRole;

    metadata def Asset {
        import ScalarValues::*;
        attribute RequiresSafetyFunction : Boolean;}

    metadata def User :> Asset {
        import ReqCheckerMeta::UserRole::*;
        attribute Role : UserRole;}

    metadata def StopFunction
        :> SafetyCounterMeasure;

    metadata def Authentication
        :> SecurityCounterMeasure {
            import ScalarValues::*;
            attribute ByPassSafety : Boolean; }
    ...
}
```

Listing 1: Excerpt of Metadata Definitions in SysML 2.0

Graph Representation of Model Instances System architects only need to import our metadata definitions in a SysML 2.0 textual representation and tag their system components using the imported metadata. Having tagged the system components with metadata, we convert the system model into a graph. This representation is the components-and-connectors view of the system [3]. Section V provides an illustrative sample case for further explanations.

We traverse the graph representation of model instances for requirements validation. For example, the standard IEC 62443-

Algorithm 1: Graph-Based Validation of Security Requirement: *Requests must be authenticated.*

Input: $G \leftarrow$ **Graph**$(SystemModel)$

$violation \leftarrow$ false

foreach *path in G* **do**

  if *path does not include authentication* **then**

    $violation \leftarrow$ true

    break

  end

end

return $violation$

[5]https://neo4j.com/

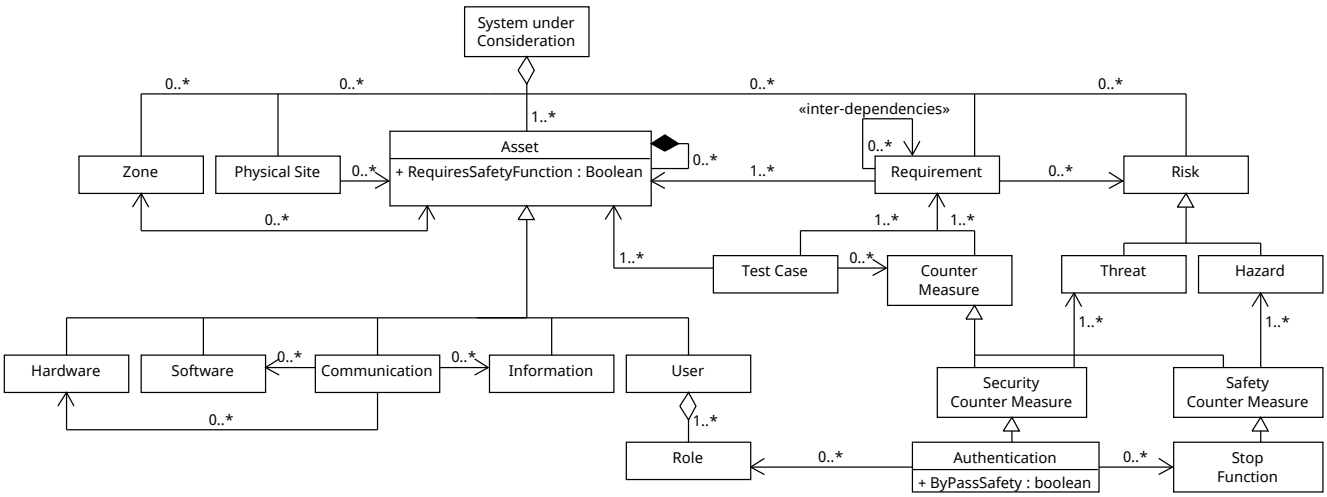

Fig. 3: Metadata Definitions in the Domain of Safety and Security of Industrial Systems

3-3[1], requires a secure industrial automation and control system to "Identify and authenticate all users (humans, software processes and devices) by mechanisms which protect against casual or coincidental access by unauthenticated entities." We check to find all paths from users to system components and make sure that all paths go through a security authenticator. If there exists a path that the request can be passed to a system component without authentication, our approach indicates a violation of security requirements. Algorithm 1 presents our validation of security requirements regarding authentication.

Different standards focus on safety of industrial applications. The IEC 61508 standard[2] requires that safety functions be implemented, e.g., a stop button triggers the system component to stop as a safety function when it is dangerous to human operators. Similarly, we traverse the graph model to make sure that for each system component (graph vertex), at least one path is available with the stop function. In this case, our approach indicates a violation of safety standards if a system component exists, to which no stop function is connected. Algorithm 2 presents our algorithm to validate the safety requirements including stop functions.

Algorithm 3 validates the integrated safety and security. The standard ISO 13850:2015[3] requires that "The emergency stop function shall be available and operational at all times". We check that requests coming from the stop function by-pass the authentication. Note that security requires that all paths

Algorithm 2: Graph-Based Validation of Safety Requirement: *Safety functions must be available.*

Input: $G \leftarrow$ **Graph**$(SystemModel)$
function stop_function_exists(G)
begin
  $SafetyComponents \leftarrow$ **find_safety_components**(G)
  $StopFunctions \leftarrow$ **find_stop_functions**(G)
  $violation\_array \leftarrow []$
  foreach *component in SafetyComponents* **do**
    foreach *function in StopFunctions* **do**
      if *path exists from function to component* **then**
        $violation\_array[component] \leftarrow$ false
        break
      else
        $violation\_array[component] \leftarrow$ true
      end
    end
  end
  return $violation\_array$
end
$violation\_array \leftarrow$ **stop_function_exists**(G)
$violation \leftarrow false$
foreach *element in violation_array* **do**
  if $element == true$ **then**
    $violation \leftarrow true$
    break
  end
end
return $violation$

Algorithm 3: Requirements Validation of Integrated Safety and Security:
The stop functions must be operational at all times.

Input: $G \leftarrow$ **Graph**$(SystemModel)$
$SecurityComponents \leftarrow$ **find_authentication_components**(G)
$violation \leftarrow$ false
foreach *component in SecurityComponents* **do**
  if *component.ByPassSafety == false* **then**
    $violation \leftarrow$ true
    break
  end
end
if *violation == false* **then**
  run **Algorithm 2**
end
return $violation$

go through authentication. As a result, the by-pass of requests coming from the stop function must be implemented in the authentication component as shown in Figure 3. We check if authentication components have the attribute `ByPassSafety` set to `true`, and run the safety algorithm. Our approach supports reusability after covering IEC 62443-3-3[1] and IEC 61508[2].

Artifact Generation Having validated the models, our approach provides a visualization of the graph of the system alongside information regarding the violation of requirements. This step is necessary for the system architects to double-check if the converted graph accurately represents the system model to make sure the given information is accurate. Our approach informs about the exact system part or parts that violate the safety and security requirements. In case the graph is not accurate, system architects can tag their system components and rerun the approach. It is possible to extend the metadata definitions to accurately reflect the needs of different systems.

V. Discussion

We present a sample case, and analyse the results.

Sample Case Figure 4 presents a *collaborative robot (cobot)* that unloads different components of a car production. The sample case was implemented in the Pilot Factory of TU Wien[6]. We only show the system parts for space reasons. The cobot has two modes, i.e., *cooperative* and *automatic*. In the cooperative mode, a human operator performs quality checks of the unloaded components. In the automatic mode, the cobot performs the tasks fully automatic without the intervention of humans. This mode happens when workforce are not available, e.g., at night shifts. In the cooperative mode, the users can stop the cooperative using *stop buttons* for safety reasons. When the cobot is in the automatic mode, *light barriers* stop the operation in case a human is around the cobot.

Human operators can control the cobot using a *cobot controller* that performs a role-based authentication. However, the cobot controller by-passes authentication in case a stop button is pressed to give priority to safety. As mentioned before, this safety function is according to ISO 13850:2015[3] to prevent damage to human lives. The attribute `ByPassSafety` is set to `true` in the cobot controller to implement this behavior. A *Manufacturing Execution System (MES)* optimizes the cobot's operation. Moreover, the MES gives access to different users, such as the maintainer, to control the industrial system.

The MES connects to the cobot controller and stop functions via a *switch* that implements an enterprise-wide *firewall*. Another switch provides a secure connection to the outside world, e.g., to integrate into a cloud platform. An attacker can insert *malicious software* into the system with the intention of harming human lives. In this case, the malicious software gains access to the cobot controller and overrides the `ByPassSafety` to `false`. This change will block the stop buttons to perform the safety functions in a hazardous situation.

Listing 2 presents the SysML 2.0 model of the illustrative sample case, tagged with our metadata definitions. Figure 5

[6]https://www.pilotfabrik.at/language/en/

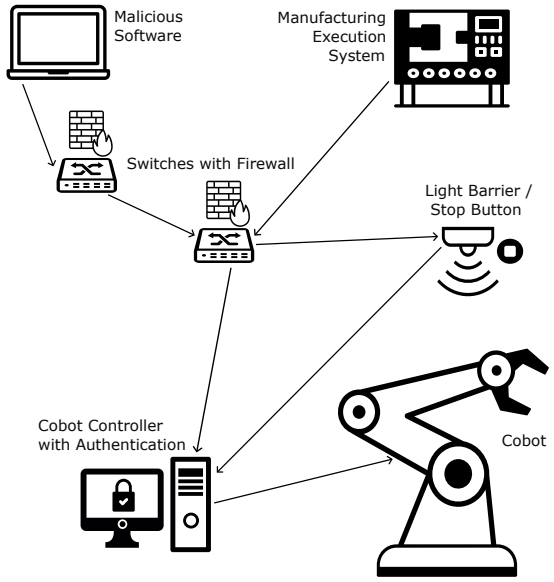

Fig. 4: Illustrative Sample Case: Collaborative Robot

shows the automatically created graph of the illustrative sample case. We present the graph of the system model so that the architect can verify the correctness of the tagging.

```
package CobotSample {
    import ReqCheckerMeta::*;

    part Cobot
          {@Asset {RequiresSafetyFunction = true;}}
    part MES
          {@Asset {RequiresSafetyFunction = false;}}
    part MaliciousSoftware
          {@Asset {RequiresSafetyFunction = false;}}

    part LightBarrier {@StopFunction;}
    part StopButton {@StopFunction;}

    part ExternalSwitch
          {@Authentication {ByPassSafety = true;}}
    part EnterpriseSwitch
          {@Authentication {ByPassSafety = true;}}
    part CobotController
          {@Authentication {ByPassSafety = true;}}

    connect MES to EnterpriseSwitch;
    connect EnterpriseSwitch to LightBarrier;
    connect EnterpriseSwitch to StopButton;
    connect EnterpriseSwitch to CobotController;
    connect LightBarrier to CobotController;
    connect StopButton to CobotController;
    connect CobotController to Cobot;
    connect MaliciousSoftware to ExternalSwitch;
    connect ExternalSwitch to EnterpriseSwitch;
}
```

Listing 2: SysML 2.0 Tagged Model of the Sample Case

Results Analysis In this section, we discuss multiple scenarios of requirements validation. Our sample case fulfills all requirements. In this case, our approach provides the results of the requirements validation as shown by Figure 6.

We model the case, where the *ExternalSwitch* does not provide authentication and is tagged as an *asset*. In this scenario, our approach informs about all paths, where the requests are not authenticated, i.e., from the *MaliciousSoftware* to the *ExternalSwitch*. We give information on the exact

Fig. 5: The Converted Graph of the Illustrative Sample Case

Passed the Requirements of the Standard IEC 62443-3-3.

Passed the Requirements of the Standard IEC 61508.

Passed the Requirements of the Standard ISO 13850:2015.

Fig. 6: Results of our Approach: Passed All Requirements

Failed the Requirements of the Standard IEC 62443-3-3.
The following paths have no authentication component:
MaliciousSoftware -> ExternalSwitch

Passed the Requirements of the Standard IEC 61508.

Passed the Requirements of the Standard ISO 13850:2015.

Fig. 7: Results of our Approach: No *Authentication*

system parts that violated the requirements, presented by Figure 7. Moreover, we study the following scenario. An attacker falsifies the attribute *ByPassSafety* of the *CobotController*. In this case, the stop functions will not work properly to stop the *Cobot* in a hazardous situation, violating the standard ISO 13850:2015 as shown by Figure 8.

VI. Conclusion

In this paper, we proposed an approach to automatically validate safety and security requirements and generate artifacts informing about the violation report. The integrated safety and security plays a vital role in the IT/OT convergence of

Passed the Requirements of the Standard IEC 62443-3-3.

Passed the Requirements of the Standard IEC 61508.

Failed the Requirements of the Standard ISO 13850:2015.
CobotController has ByPassSafety set to false.

Fig. 8: Results of our Approach: No *ByPassSafety*

industrial systems. On the one hand, security standards, such as IEC 62443, require all requests to be authenticated. On the other hand, safety standards, e.g., ISO 13850:2015, require that safety measures must be available at all times to stop the operation if a human life is in danger. We studied the research question how we can automatically validate safety and security requirements according to related standards, e.g., IEC 62443[1], IEC 61508[2], and ISO 13850:2015[3], for industrial systems.

To this end, we used an MBSE approach [16] using the SysML 2.0 textual representation. We proposed metadata in the domain of safety and security, which are used as tags in the system model. The tagged elements are automatically converted to graphs for requirements validation. We had reusability in mind, when designing our method. It is possible to extend our models covering more aspects of the safety and security iteratively. After validating only two standards, i.e., IEC 62443 and IEC 61508, we reused our approach to validate the standard ISO 13850:2015. In this paper, we provided support for these three standards as a proof-of-concept. For future work, we aim to provide a comprehensive framework.

REFERENCES

[1] A. Ahlbrecht and O. Bertram. Evaluating System Architecture Safety in Early Phases of Development with MBSE and STPA. In *2021 IEEE International Symposium on Systems Engineering (ISSE)*, pages 1–8. IEEE, 2021.

[2] M. M. Alani and M. Alloghani. *Security Challenges in the Industry 4.0 Era*. Springer International Publishing, Cham, 2019.

[3] V. Bertram, S. Maoz, J. O. Ringert, B. Rumpe, and M. von Wenckstern. Component and connector views in practice: An experience report. In *2017 ACM/IEEE 20th International Conference on Model Driven Engineering Languages and Systems (MODELS)*, pages 167–177, 2017.

[4] P. De Saqui-Sannes, R. A. Vingerhoeds, C. Garion, and X. Thirioux. A Taxonomy of MBSE Approaches by Languages, Tools and Methods. *IEEE Access*, 10:120936–120950, 2022.

[5] B. C. Ervural and B. Ervural. *Overview of Cyber Security in the Industry 4.0 Era*. Springer International Publishing, Cham, 2018.

[6] S. Hollerer, W. Kastner, and T. Sauter. Safety and security: A field of tension in industrial practice. In *2023 IEEE 21st International Conference on Industrial Informatics (INDIN)*, pages 1–7, 2023.

[7] S. Hollerer, W. Kastner, and T. Sauter. Towards a comprehensive ontology considering safety, security, and operation requirements in ot. In *2023 IEEE 28th International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1–4, 2023.

[8] S. Japs. Security & Safety by Model-based Requirements Engineering. In *2020 IEEE 28th International Requirements Engineering Conference (RE)*, pages 422–427. IEEE, 2020.

[9] S. Japs and H. Anacker. Resolution of safety relevant security threats in the system architecture design phase on the example of automotive industry. *Proceedings of the Design Society*, 1:2561–2570, 2021.

[10] A. Kharatyan, J. Tekaat, S. Japs, H. Anacker, and R. Dumitrescu. Metamodel for safety and security integrated system architecture modeling. *Proceedings of the Design Society*, 1:2027–2036, 2021.

[11] S. Kropatschek, S. Hollerer, D. Hoffman, D. Winkler, A. Lüder, T. Sauter, W. Kastner, and S. Biffl. Combining models for safety and security concerns in automating digital production. In *2023 IEEE 21st International Conference on Industrial Informatics (INDIN)*, pages 1–8, 2023.

[12] K. Kurniawan, A. Ekelhart, E. Kiesling, G. Quirchmayr, and A. M. Tjoa. Krystal: Knowledge graph-based framework for tactical attack discovery in audit data. *Computers & Security*, 121:102828, 2022.

[13] J.-w. Myung and S. Hong. Ics malware triton attack and countermeasures. *Int. J. Emerg. Multidiscip. Res*, 3:13–17, 2019.

[14] S. Pirbhulal, V. Gkioulos, and S. Katsikas. Towards integration of security and safety measures for critical infrastructures based on bayesian networks and graph theory: A systematic literature review. *Signals*, 2(4):771–802, 2021.

[15] M. Quamara, G. Pedroza, and B. Hamid. Multi-layered Model-based Design Approach towards System Safety and Security Co-engineering. In *2021 ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)*, pages 274–283. IEEE, 2021.

[16] A. L. Ramos, J. V. Ferreira, and J. Barceló. Model-based systems engineering: An emerging approach for modern systems. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(1):101–111, 2012.

[17] N. Subramanian and J. Zalewski. Quantitative Assessment of Safety and Security of System Architectures for Cyberphysical Systems Using the NFR Approach. *IEEE Systems Journal*, 10(2):397–409, 2016.

[18] B. Vogel-Heuser, D. Schütz, T. Frank, and C. Legat. Model-driven engineering of Manufacturing Automation Software Projects – A SysML-based approach. *Mechatronics*, 24(7):883–897, 2014.