A Safety and Security Requirements Management Methodology in Reconfigurable Collaborative Human-Robot Application

Ali M.Hosseini*, Clara Fischer‡, Mukund Bhole§, Wolfgang Kastner§, Thilo Sauter†¶, Sebastian Schlund‡
Institute of Computer Technology, TU Wien, Vienna, Austria*†
Institute of Management Science, TU Wien, Vienna, Austria‡
Institute of Computer Engineering, TU Wien, Vienna, Austria§
Dep. of Integrated Sensor Systems Danube Univ. Krems, Wiener Neustadt, Austria¶
Email: [First Name].[Last Name]@tuwien.ac.at

***Abstract*—The current industry has to adapt to rapidly changing customers' needs. Reconfigurable manufacturing, therefore, provides capacity and functionality on demand which is essential for competitiveness in fast-changing markets. Furthermore, Industry 4.0 or even more so, Industry 5.0 emphasizes human-centred production with collaborative robots, Cobots, to create human-robot interactions. In such scenarios, safety and security are difficult to address due to the intrinsic features of reconfigurable manufacturing, like exposure to numerous requirements changes in a short period. As safety and security can conflict in different phases of the system life-cycle, one of the earliest activities to avoid conflicts is requirements engineering which can significantly diminish the cost and time of fixing issues compared to later phases like operation. This paper proposes a methodology for safety and security requirements interaction management, including conflict detection and resolution, and shows its applicability through a reconfigurable collaborative human-robot use case. Based on the proposed methodology, we detected and resolved two safety and security requirement conflicts.**

***Index Terms*—Safety, Security, Reconfigurable Manufacturing, Cobot System, Requirements Engineering**

I. Introduction

Reconfigurable manufacturing systems [1] allow a fast and effective reaction to sudden and frequent changes in demand. Convertibility and scalability are two core features of reconfigurable systems. Scalability is the ability to achieve fast and efficient incremental changes in capacity, and convertibility is the ability to conduct quick transformation between existing products and adapt to future ones.

Safety and security as non-functional requirements constrain and characterize functional requirements in manufacturing systems. There are several standards for security and safety, like IEC 62443 for security and IEC 61508 for functional safety. In practice, safety and security activities are mostly distinct and handled separately. However, safety and security affect each other, and their mutual interaction can lead to conflicts where enhancing safety puts security at risk or vice-versa [2]. Early understanding of requirements and choice of architectural elements is essential to manage manufacturing systems in different life-cycle phases [3]. The architectural elements include users, hardware components, software codes, facilities, policies, and documents required to produce system-level results. The results include system-level qualities, properties, characteristics, functions, behaviour, and performance [4]. Since requirements engineering is one of the earliest activities for system design and integration, identifying potential conflicts and risks is essential in reducing the costs and efforts compared to later phases like operation [5]. Requirements interaction management consists of activities for identifying, managing, and disposing of relationships among requirements sets as part of requirements engineering [6]. It is more efficient to perform incremental requirements interaction management during reconfiguration. The term incremental here means that the requirements engineer only will focus on affected parts during reconfiguration to inspect possible consequences instead of doing it again for the whole process [7].

However, using requirements management to identify safety and security conflicts of reconfigurable systems in the early integration phase faces hurdles caused by the current laws. The machinery directive, which harmonizes health and safety requirements for machinery in the European member states, represents regulations for complete and incomplete machines and changes in machines [8]. An industrial robot, for example, represents an incomplete machine. Without a tool, safety equipment, and additional components, the robot is not considered as a complete machine. Therefore, the respective stakeholders, usually system integrators, are responsible for the risk assessment of the entire application. Especially for collaborative applications, which present a fence-less interaction between a human and a robot, this assessment can become very elaborate and usually requires the help of external testing and certification institutes [9]. Furthermore, according to the machinery directive, any change in the machine requires a new assessment. In the case of a collaborative application, even adapting the robotic path is considered a change, which makes the reconfiguration of such systems very difficult. In addition, such risk assessments are usually performed in the integration phase. However, earlier identification of safety and security risks and their mutual impacts could save significant efforts and time.

This paper presents a methodology for safety and security requirements conflict management in re-configurable systems, with the example of a human-robot application. Here, we will answer the following research questions:

- What are the interconnections of safety and security in terms of defined system requirements and how can they be modelled?
- How can reconfiguration affect safety and security requirements and their interconnections?
- How can conflicts between safety and security requirements be identified and resolved?

This paper is organized as follows: Section II presents the state of the art in the field of safety and security dependencies and conflicts, as well as specific requirements for collaborative human-robot applications. In Section III, we present our methodology for conflict management, based on requirements which we analyze and discuss on a configurable human-robot use case, described in Section IV. Finally, we conclude our paper and give an outlook on future work in Section VI.

II. State of the Art

A. Requirement Traceability

Requirement traceability is considered an effective method to trace the requirement changes in the complete life-cycle of the process. Requirement traceability not only enhances the change management approach but also assures the correctness and grade of the entire life-cycle of the system. These requirements can be traced in different directions, as shown in Fig. 1. The needs of customers are traced forward to requirements. This link helps one to find affected requirements when a customer's need changes. Reversely, one can trace back from requirements to customer needs to detect the source of requirements. Forward tracing from requirements to one specific product or system architecture element makes it possible to ensure that all the requirements are addressed. Tracing specific product elements backwards to requirements justifies why each element was created. Requirement traceability has several benefits; with the lack of trace information, there is a high chance that one misses a system element that would be affected when adding, deleting, or modifying a particular requirement. Besides, using trace information, it is possible to identify how changes propagate from the system architecture element through the requirements and vice-versa [10].

System Modelling Language (SysML) is a powerful means for having traceability not only at the requirement level but at the system architecture level or a combination of both. SysML supports different diagrams (requirement diagrams, behavioural diagrams and structural diagrams) as system views that can be synchronized and, therefore, track changes for incoming inputs from the reconfiguration process [11].

SysML defines six types including ”Trace”, ”Refine”, ”Derive”, ”Copy”, ”Satisfy” and ”Verify”. These types of relationships allow for cross-connecting architectural elements and requirements. Table I depicts the types of relationships by SysML. SysML serves as a standardized notion allowing all stakeholders in system development to have the same understanding and effective communication. Different relationships for cross-model interconnection in SysML can be identified on the basis of relationship types introduced in Table I [12].

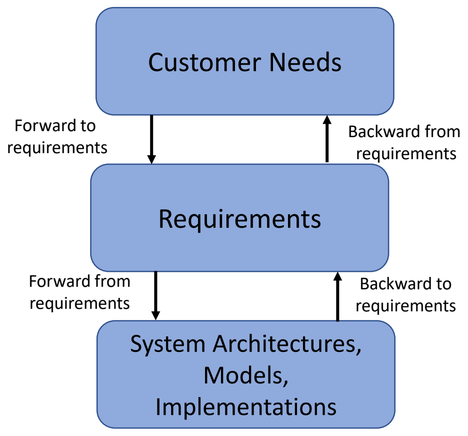
Customer Needs
Forward to requirements
Backward from requirements
Requirements
Forward from requirements
Backward to requirements
System Architectures, Models, Implementations

Fig. 1. Traceability from customer's needs to final product [10]

TABLE I
Relationships for cross model interconnection in SysML.

Type	Description
Trace	General trace relationship between requirements and any other architectural elements
Refine	Shows an architectural element that clarifies a requirement
Derive	A requirement is derived from another requirement
Copy	Reused requirement by copying a requirement
Satisfy	Shows a design or an architectural element that satisfies the requirement
Verify	Used to show a test case that has been used to verify a requirement

B. Safety and Security Requirements Dependencies

While considering safety and security dependencies, we encounter a couple of different approaches. However, based on [13], they can be categorized into two main classes: In a *Unified Approach*, safety and security with their dependencies are considered in a single term from the beginning. An *Integrated Approach* deals with safety and security separately and then combines those two at the end. Further, there can be four different dependencies between safety and security as follows [14]:

- *Conditional Dependency*: Satisfaction of security requirements conditions safety or vice-versa. For instance, attackers can manipulate sensor data and create a dangerous safety situation.
- *Mutual Reinforcement*: Fulfilling safety requirements reinforces security and vice-versa.
- *Conflicting dependency*: Addressing a security requirement can have a negative impact on a safety requirement or vice-versa. In deterministic timing, it helps to improve safety since it allows one to decide how to deal with

worst-case delays. However, deterministic timing is vulnerable to side-channel attacks in which the attacker tries to compromise encryption by analyzing the time taken to perform cryptographic algorithms [2].
- *Independence*: No direct dependency between safety and security.

C. Safety and Security Requirements Conflicts Detection and Resolution

In this paper, the meaning of conflict in the context of safety and security is a situation in which safety negatively impacts security or vice-versa. There are different approaches to defining, identifying, and resolving conflicts. Conflict definition and detection can be done using rule-based, taxonomy-based, or ontology-based approaches [15]. In a rule-based approach, conflicts happen if the rules or policies are conflicting since they perform a contrasting or unwanted action on the same part of the system. In taxonomy-based approaches, requirements are grouped into classes based on their properties, which are used to identify conflicts between requirements. An ontology-based approach is a specification of the conceptualization of a domain that allows the representation of knowledge via well-defined semantic and syntactic rules leading to finding conflicts [16]. Identified conflicts can be resolved dynamically or statically. In dynamic methods, the conflicts should be discovered and solved dynamically in run-time, while in static methods, the focus is on conflict avoidance by resolving them at design time [16]. For both methods, one can define different policies to be applied in a conflicting situation, as shown in Table II [6].

D. Safety and Security of Human-Robot Interaction (HRI)

In this paper, we will focus on a collaborative application, meaning the operator interacts fence-less with a collaborative robot (Cobot). Cobots are compliant robots equipped with specific safety sensors, for example, torque sensors on joints, designed to be used for collaboration tasks without fences. It is important to mention that only tasks and applications can be considered collaborative, independent of the robot used. The term collaborative robot is explicitly no longer mentioned in the new standards [17]. A risk assessment of the entire application is required, including the Cobot, end-effector, robotic path, manipulated tools, external sensors, and further components of the workstation. Besides the general machinery safety requirements in [18], for collaborative applications ISO/DIS 10218-2:2020, the latest draft of the standard for safety requirements of industrial robot systems, applications and integration [17] has to be considered. In this document, the technical specification for collaborative robot systems ISO/TS 15066:2016 [19], is being integrated.

TABLE II
CONFLICT RESOLUTION POLICIES.

Type	Description
Elimination	One of the conflicting requirements shall be eliminated.
Prioritization	The conflicting requirements shall be prioritized.
Refinement	Changing conflicting requirements in a way that resolves conflict.
Postponement	Postponing conflict resolution (resolving other conflicts may solve this one).

Depending on the level of interaction, the following HRI types are considered in literature [20]: *Coexistence* means that the human and the robot work next to each other, without protection fences, but in separate workspaces. In *Synchronization*, the interaction agents have a shared workspace, but they only enter it separately, meaning they are working time-shifted. In *Cooperation* the operator and the robot can enter their shared workspace at the same time, but they are working on different workpieces. And the last, *Collaboration*, describes the closest level of human-robot interaction, where both interaction partners are working together in a shared workspace, at the same time, on the same product.

Fenceless HRI is generally incorporated into the following safety modes of human-robot interaction [17]. For more general requirements, see [17], [19], [21]. *Monitored-standstill* presents a category two stop, a controlled stop with an energy supply remaining at standstill, and can be used in cases where the robot may only move if the human is not in the shared workspace. As the human enters this space, the robot has to go into a monitored-standstill and only unlocks as soon as the worker has left this area. In *Hand-guided controls (HGC)* the robot is moved manually via a hand-guiding device. As soon as the human stops the manual guidance, the robot must enter a monitored-standstill unless at least one of the two following modes provides risk reduction. *Speed and Separation Monitoring (SSM)* limits the robot's speed, depending on its distance from the human. If the human is beyond a predefined safety distance, the robot can drive fast but must slow down when the worker enters the safety distance and stop at a minimum distance. *Power and Force Limiting (PFL)* restricts the application, to comply with the biomechanical limits specified in ISO/TS 15066:2016. These thresholds are derived from studies for pain sensibility and represent maximum values for pressure and force for 29 body regions, that can be tolerated without causing any pain in the event of a collision. The evaluation of this safety mode requires physical tests and biomechanical measurements, see [22].

In [23], the effects of cyber attacks on Cobot safety have been investigated. IEC 62443 is an international series of standards that handles cybersecurity in industrial control systems and is also applicable to the security of a collaborative application. IEC TR 63074 and IEC TR 63069 bridge functional safety and cyber security [24]. They provide recommendations on the common application of IEC 61508 (all parts) and IEC 62443 (all parts) in the field of industrial process measurement, control and automation. Although the aforementioned standards for collaborative applications are more focused on safety either, it is not feasible to consider these components separately, because of their interconnected nature. Adding a safety sensor can provide an external access route that could be used for an attack. Depending on the attack, it could lead to a change in speed or movement if the controller gets attacked, which in turn poses a further threat to safety. On the other hand, adding a firewall improves security but requires additional physical components, such as a control box. However, this control box poses an additional safety risk, especially if it is mounted overhead, there the risk of hitting one's head. In addition, there is a risk of electrical hazards.

III. Methodology for Requirements Interaction Management

The proposed requirements interaction management methodology is shown in Fig. 2, which demonstrates the required steps for managing requirements in the example of a reconfigurable human-robot application. The objective of this methodology is to speed up the reconfiguration process and prevent safety and security conflicts in requirements space. This methodology introduces required activities for incremental analysis of a reconfigurable system to handle or prevent the conflicting situations that can happen between safety and security requirements.

In the first step, the activity starts with the existing requirements and system architecture or design specifications of the first configuration. Due to a large number of requirements and architectural elements, it is inefficient to perform analysis for the whole system during each reconfiguration. Partitioning focuses on dividing requirements and architectural elements into smaller and more manageable subsets. It allows focusing on the changed or affected parts rather than all parts. This methodology makes requirements conflict management more efficient, especially in large-scale systems. The partitioning can be conducted based on the system's goals, functions, requirements' attributes, scenarios, etc [6]. Different relationships among requirements and system architectural elements should be identified for each partition. Table .I can be used for this purpose. After identifying and modelling the relationships, the changes in requirements and architectural elements caused by reconfiguration are applied to already defined partitions, requirements, architectural elements, and relationships. The change management process becomes faster because of already defined partitions and SysML. SysML supports different diagrams (requirement diagrams, behavioural diagrams and structural diagrams) as system views that can be synchronized and, therefore, track changes for incoming inputs from the reconfiguration process.

Updated requirements and system architectures enable finding conflicts among safety and security requirements. As discussed in Section II-C, different approaches can be employed to define, detect, and resolve conflicts. We will discuss this in more detail in the following sections, where we apply the methodology to find conflicts during the reconfiguration of a collaborative human-robot application. Eventually, the results would be new sets of requirements and system architectures that are ready to undergo another reconfiguration process.

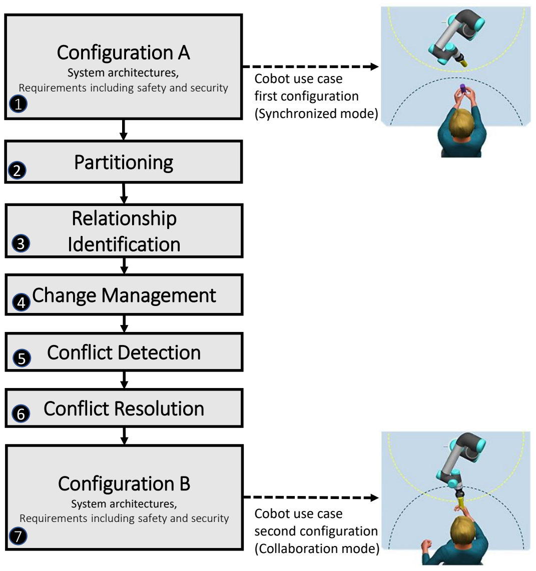

Fig. 2. Methodology for the Requirement Interaction Management

IV. Configurable Human-Robot Use Case

A. Description of the Collaborative Use Case

The use case presents the assembly of a ski binding, the workspace of which can be seen in Fig 3. A collaborative robot picks the parts of the binding from the robot workspace and places them accordingly on the ski in the collaborative workspace. The human tasks include the mounting and screwing of the binding.

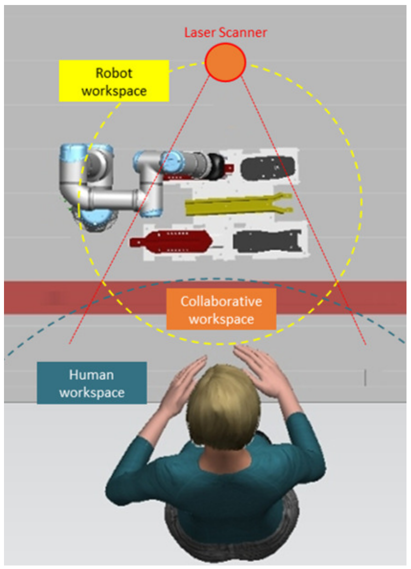

Fig. 3. Human-Robot Interaction Workspace Layout

1) Configuration 1: Synchronization (SSM): In the first configuration, the operator and the robot interact in synchronization. They are placing and fixing one part at a time and entering the shared, collaborative workspace in a time-shifted manner. Via a sequence button, the human is communicating with the robot, to pause and start its movement. To ensure safety, a laser scanner monitors the application from above and is used to guarantee speed and separation monitoring (SSM).

In order to increase the efficiency of the process, it is being reconfigured for closer interaction and shorter cycle times.

2) Configuration 2: Collaboration (PFL): The second configuration describes the same task, with the difference that the robot can already pick up and place the next part while the human is still attaching the previous one. Both interaction partners can work at the same time in the collaborative workspace. If the human needs extended time for screwing, he or she can use the cycle button to pause the robot and let it continue. The safety shall be ensured by power and force limiting (PFL), therefore, the laser scanner is no longer needed. However, a Real-Time Data Exchange (RTDE) protocol is used to communicate, and a script shall generate the robot's motion data in real time and upload it to the cloud Cobot service platform if a collision occurs. This additional function aims to check the location and frequency of the impact by the robot's position.

Fig. 4 shows the system architectures of the two configurations. The connection between the components is illustrated by blue arrows, and it is also indicated how they communicate. In addition, green dashed lines represent the interaction between the operator to the system. In both configurations, the operator starts the process with the robot's teach pendant, which is equipped with an emergency stop button. However, to meet the requirement of having a universally accessible emergency stop button, an additional one was installed.

B. Conflict Detection Based on Proposed Methodology

In the first configuration, synchronization mode, we have to prepare all the system architectures, design specifications and requirements specification documents (see Fig. 4 and Table III). To this end, brainstorming is used for requirement elicitation. Afterwards, we use a scenario-based approach to build partitions. Scenarios represent paths of possible behaviour through a use case, and these are investigated to elaborate requirements [25]. Scenarios help to determine groups of requirements and architectural elements that are related. For the first configuration, we consider three scenarios:
a) Production Mode
b) Robot Failure
c) Velocity-Distance Violation

Table IV shows the *Production Mode* scenario consisting of a few steps. By analyzing this scenario, we can see which requirements and system architecture elements are connected and have physical or logical interactions. For instance, step 2 shows the authentication requirement, while the emergency stop button or laser sensor is not involved in this scenario. We index all the requirements and system architectures into partitions using SysML by writing a detailed scenario. The other two scenarios, robot failure and velocity-distance violation are not discussed in detail since they are similar to production mode.

The next step is identifying and modelling the relationships in partitions resulting from the previous step. This is done by SysML which serves as a standardized notion allowing all stakeholders in system development to have the same understanding and effective communication.

Now, we have all the requirements and system architectures for the synchronized configuration in the form of partitions with identified and modelled relationships. To start reconfiguration, we find all the changes required by the second configuration to requirements and system architectures and apply them to the existing models from the first configuration. This strategy helps to reconfigure the system faster, and there is no need to go through all the safety and security requirements, which is costly and very time-consuming. Requirements and system architecture change management is a wide research area and there are numerous works addressing impact analysis, change prioritization, user involvement, etc [26]. We have used requirements and architectural elements traceability to manage changes in SysML diagrams.

Most of the requirements can be described as actions that transfer system states from initial to final states. This action gets started by a trigger and consumes a resource(s) to complete the task. We categorize conflicting situations based on the consequence into three main rules, *Negative Impact*, *Non-determinism*, and *Block*. Safety and security requirements can impact each other negatively when one requirement's action is against the other and makes it difficult to be fulfilled. For example, using the same element for two safety and security requirements simultaneously can decrease its performance. A non-deterministic situation can occur to dynamic system behaviour when safety and security actions with the same priority act simultaneously and tend to change the system's states differently. In this situation, the system's states are not deterministic. A block type is when a safety requirement blocks the security requirement or vice versa like when safety and security requirements depend on the same resource (e.g., two requirements are satisfied by the same architectural element such as an actuator) that can serve only one requirement at a time.

According to the defined rules (*Negative Impact*, *Non-determinism*, and *Block*), we can manually and visually inspect the SysML diagrams to find conflicts. In SysML diagrams, we can see different relationships among requirements and architectural elements. They specify requirement resources, traces, and architectural elements which satisfy the requirements. According to the rules for finding conflicts, one can use the information provided by SysML to find a matching situation. For instance, SysML diagrams can highlight the requirements resources. If more than one requirement relies on the same resource (for example, two requirements use a sensor at the same time), they can be a candidate for a block or negative impact conflict.

Here are two found conflicts using the suggested methodology in the reconfiguration of a collaborative human-robot application:

Conflict 1: Req 8.0 (safety) partially blocks Req 26.0

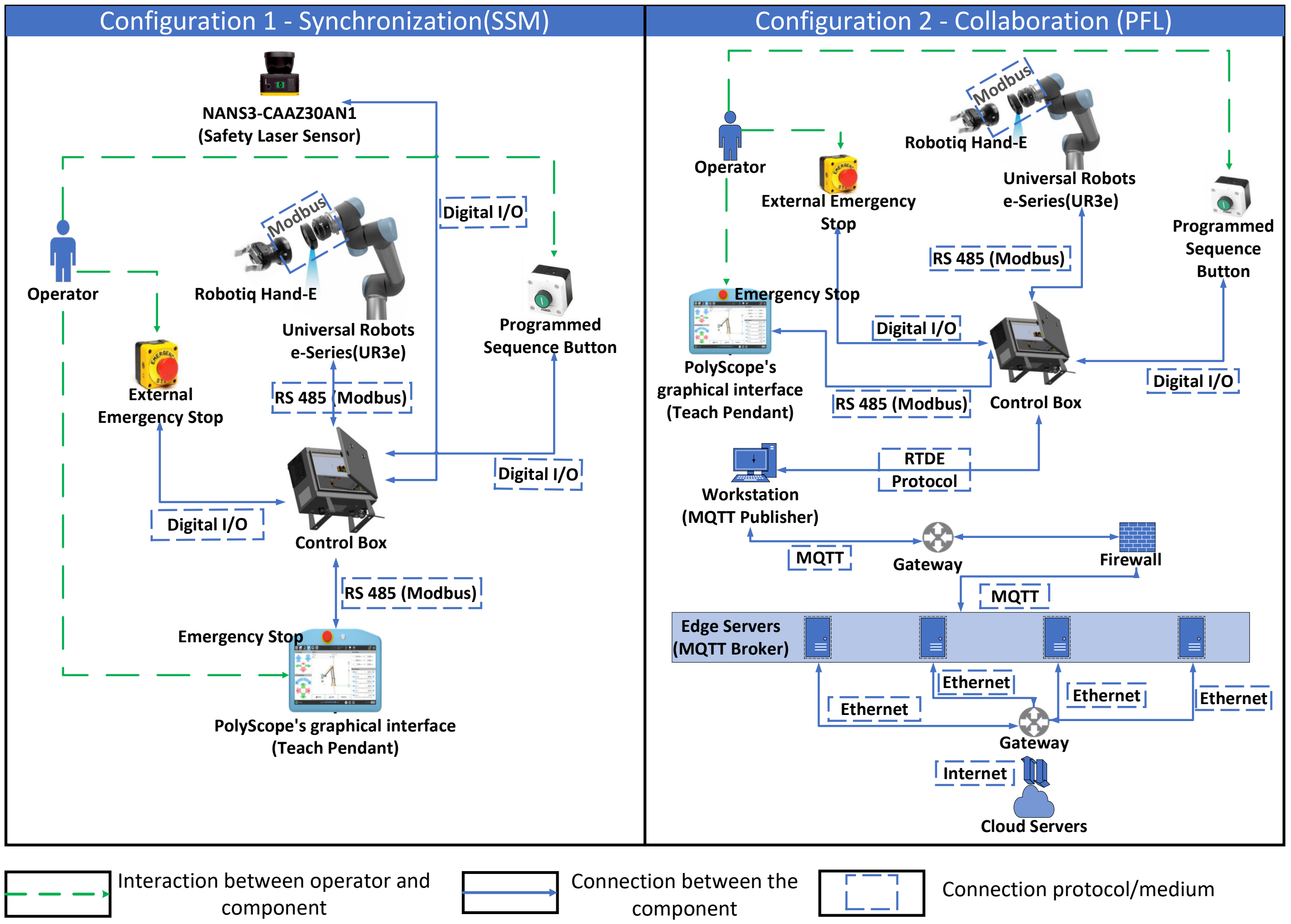

Fig. 4. System architecture of a human-robot interaction use case, in two configurations. Configuration 1, Synchronization in Speed and Separation Monitoring (SSM), configuration 2, Collaboration in Power and Force Limiting (PFL)

(security). From a safety viewpoint, safety settings shall be stable while for security reasons, the system admin must perform patch management or security updates in case of any security issues even remotely. Fig. 5 illustrates a simplified version of a SysML diagram showing different relationships in this conflicting situation. It is possible to see how two safety and security requirements (Req 8.0 as a safety requirement, and Req 26.0 as a security requirement) trace forward to the controller as an architectural element and relate its attributes such as *RobotSafetyConfiguration*.

Table II shows four policies for conflict resolution. These policies can come together with various conditional statements specified by domain experts, requirements engineers, system architecture, and other stakeholders. For resolving *conflict 1*, we assumed safety has the highest priority in our case and therefore, we refined the security requirements not to conflict with the safety requirement. Because we need to take suitable actions in case of security incidents remotely for timely response, we limit Req 26.0 to non-safety configurations.

Conflict 2: Req 17.0 (safety) impacts negatively Req 24.0 (security). Various types of information, such as configuration, run-time data, and technical specifications need to be transferred to the cloud. Since the cloud itself expands security attack vectors, this safety requirement can expose much information in case of an attack on the cloud or during data transmission. To avoid transferring various types of information to the cloud Cobot service platform for security reasons, we can eliminate the safety requirement or refine it, as failure detection shall be done on the workstation rather than in the cloud.

V. Discussion

Based on the results of our use case, here are two general recommendations. First, the possibility of remote access to the system's settings can help in the areas of maintenance and problem-solving, but it always represents a possible security gap, which can then affect safety. Therefore, we would advise against this option unless necessary.

Second, cloud applications have been widely growing recently due to the high amount of storage and computation power. However, it should be taken into account what kind

TABLE III
SELECTED REQUIREMENTS IN SYNCHRONIZED AND COLLABORATIVE CONFIGURATIONS

Id	Name	Description	Requirement Type	Configuration
Req 1.0	Path Planning	The robot path shall be programmed in order to let the robot pick and place the parts on the ski-binding, in a predefined sequence.	Functional	Common
Req 2.0	Manual Control	The robot shall be controllable manually by operator.	Functional	Common
Req 2.1	Manual Stop/Start/Restart	The operator shall be able to start, stop or reset the process via pendant (touch panel).	Functional	Common
Req 3.0	Synchronization	The robot shall work with the human to share the common workspace in a time-shifted manner.	Functional	Synchronized
Req 3.1	Sequence Control	The human shall be able to manually control the sequence of the process.	Functional	Synchronized
Req 4.0	Homing	The robot shall go to the predefined home position when the process stops.	Functional	Common
Req 5.0	Cloud Feature	The collaborative system shall be monitorable from cloud.	Functional	Collaborative
Req 5.1	Cloud Storage	The collaborative system run-time data shall be stored in the cloud.	Functional	Collaborative
Req 5.2	Sample Time	The sample time of cloud storage shall be less than 100 mili sec.	Functional	Collaborative
Req 8.0	Safety Limits	The safety limits of the robot may not be changed.	Safety	Common
Req 9.0	Emergency Stop	System shall support emergency stop mechanism.	Safety	Common
Req 9.2	Emergency Stop Button	The system shall shut down in case of emergency if the emergency stop is pressed.	Safety	Common
Req 11.0	Human-Robot Interaction	The tasks of the human and the robot have to be clearly defined, and may not be changed.	Safety	Common
Req 12.0	SSM	The speed of the robot has to be regulated depending on the separation distance to the human.	Safety	Synchronized
Req 13.0	Minimum Distances	The minimum distances between human worker and robot shall according to the formula of EN ISO 1387:2019 has to comply.	Safety	Synchronized
Req 14.0	Collision Avoidance	No contact between the human and the robot is allowed.	Safety	Synchronized
Req 14.1	Stop Collision	System shall be stopped automatically in case of collision.	Safety	Collaborative
Req 16.0	Power and Force Limiting	The power and the force of the application have to be limited in order to comply with the biomechanical limits according to ISO/DIS 10218-2:2020 in case of a collision.	Safety	Collaborative
Req 17.0	Failure Detection	The system shall be able to detect failures via a failure detection algorithm over the cloud.	Safety	Collaborative
Req 17.1	Failure Detecting Stop	The cloud shall be able to stop the process automatically if any failure is detected.	Safety	Collaborative
Req 20.0	Authentication	The system shall authenticate all users including humans, software processes, devices, etc.	Security	Common
Req 20.1	Human Authentication	The Pendant shall authenticate the human to prevent intentional or unintentional unauthenticated access.	Security	Common
Req 21.0	Untrusted Network	System shall prevent access from untrusted wireless or wired networks.	Security	Common
Req 22.0	Use Control	The emergency stop has to be clearly visible and universally accessible.	Security	Common
Req 24.0	Confidentiality	The Cobot-related information shall be protected against unauthenticated users.	Security	Common
Req 24.1	Information Confidentiality	The system shall protect the information confidentiality at rest or in transit via untrusted networks.	Security	Common
Req 25.0	Safety Configuration Protection	The safety settings of the robot shall to be password secured.	Security	Common
Req 26.0	Remote Configuration	System administrator shall be able to update system configuration remotely.	Security	Collaborative
Req 27.0	Communication Integrity	The system shall provide the capability to protect the integrity of transmitted information.	Security	Collaborative
Req 28.0	Boundary Protection	The system shall provide the capability to monitor and control communications at zone boundaries with suitable firewalls, routers, gateways, etc.	Security	Collaborative

of information is going to be sent to the cloud Cobot service platform. The complexity of the manufacturing systems might cover different relationships among requirements and architectural elements inside and outside of a factory. For instance, a data breach from the cloud can disclose safety critical information about the assets inside a factory. Using the methodology provided in this paper, one can build a rather clear view of the system to avoid conflicting situations between safety and security.

VI. CONCLUSION AND OUTLOOK

The paper presents a safety and security requirements interaction management methodology focusing on conflict resolution and targeting reconfigurable manufacturing. The proposed methodology takes advantage of combining system requirements (functional, safety and security) and system architectures to build a clear view of the system, which is responsive to changes stemming from reconfiguration. Moreover, proactive conflict resolution in the early phases of the system life-cycle, such as requirements engineering, can reduce time and cost in comparison to later phases, like operation.

In this paper, SysML has been used to identify and model the relationships among requirements and architectural elements since it provides synchronized views of the system ranging from requirement diagrams to structural diagrams to behavioural diagrams. The proposed methodology is used to detect and resolve potential safety and security conflicts in a human-robot use case during a reconfiguration process from synchronized (speed and separation monitoring) to collaborative (power and force limiting) mode with some new features like remote access. As a result, we found two conflicts between safety and security and proposed resolution actions.

The process of conflict detection and resolution in our

TABLE IV
PRODUCTION MODE-SYNCHRONIZED CONFIGURATION.

Step	Description
1	The operator enters the working area
2	The operator enters his/her user name and password in the pendant
3	The controller verifies the entered information
4	The operator receives permission to work
4.1	If the entered username and password are invalid, the pendant shows the wrong username/password and asks for trying again
5	The operator presses the start button on the teach pendant
6	The robot picks up part no. 1, manipulates, and enters the common working area
6.1	If the operator is still working, he/she must press the sequence button to stop the robot.
7	The robot places part 1 on the ski binding, leaves the collaborative workspace and the operator enters it and fixes the part
8	Steps 6-7 are repeated till all parts are fixed
9	The operator shuts down the process and the robot moves to the home position
10	The operator logs out of his account and leaves the working area

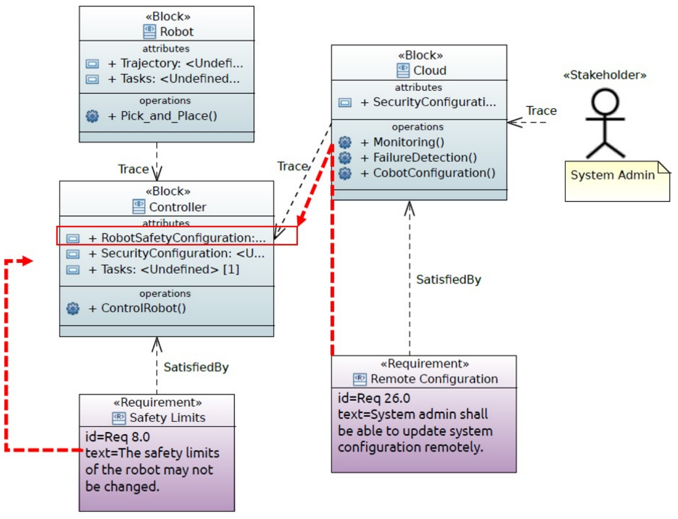

Fig. 5. A simplified view of SysML model combining safety and security requirements with architectural elements and their relationships.

methodology is manual, which can be difficult for large-scale systems with thousands of requirements. In future work, we will focus on automation of the methodology by declaring requirements, architectural elements, conflict rules, and conflict resolution policies formally.

ACKNOWLEDGMENT

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

REFERENCES

[1] A. Napoleone, A.-L. Andersen, A. Pozzetti, and M. Macchi, "Reconfigurable manufacturing: a classification of elements enabling convertibility and scalability," in *IFIP International Conference on Advances in Production Management Systems*. Springer, 2019, pp. 349–356.

[2] C. Menon and S. Vidalis, "Towards the resolution of safety and security conflicts," in *2021 International Carnahan Conference on Security Technology (ICCST)*. IEEE, 2021, pp. 1–6.

[3] B. Nuseibeh, "Weaving together requirements and architectures," *Computer*, vol. 34, no. 3, pp. 115–119, 2001.

[4] R. Cloutier, G. Muller, D. Verma, R. Nilchiani, E. Hole, and M. Bone, "The concept of reference architectures," *Systems Engineering*, vol. 13, no. 1, pp. 14–27, 2010.

[5] A. M. Hosseini, T. Sauter, and W. Kastner, "A safety and security reference architecture for asset administration shell design," in *2022 IEEE 18th International Conference on Factory Communication Systems (WFCS)*. IEEE, 2022, pp. 1–8.

[6] W. N. Robinson, S. D. Pawlowski, and V. Volkov, "Requirements interaction management," *ACM Computing Surveys (CSUR)*, vol. 35, no. 2, pp. 132–190, 2003.

[7] N. A. Ernst, A. Borgida, and I. Jureta, "Finding incremental solutions for evolving requirements," in *2011 IEEE 19th International Requirements Engineering Conference*. IEEE, 2011, pp. 15–24.

[8] DIRECTIVE 2006/42/EC, "of the european parliament and of the council of the european union: Machinery directive," 17 May 2006.

[9] M. Rathmair and M. Brandstötter, "Safety as bad cop of physical assistance systems?" in *Smart Technologies for Precision Assembly*, ser. IFIP Advances in Information and Communication Technology, S. Ratchev, Ed. Cham: Springer International Publishing, 2021, vol. 620, pp. 344–357.

[10] K. Wiegers and J. Beatty, *Software requirements*. Pearson Education, 2013.

[11] E. J. Vidal and E. R. Villota, "Sysml as a tool for requirements traceability in mechatronic design," in *Proceedings of the 2018 4th International Conference on Mechatronics and Robotics Engineering*, 2018, pp. 146–152.

[12] S. Friedenthal, A. Moore, and R. Steiner, *A practical guide to SysML: the systems modeling language*. Morgan Kaufmann, 2014.

[13] G. Kavallieratos, S. Katsikas, and V. Gkioulos, "Cybersecurity and safety co-engineering of cyberphysical systems—a comprehensive survey," *Future Internet*, vol. 12, no. 4, p. 65, 2020.

[14] S. Kriaa, L. Pietre-Cambacedes, M. Bouissou, and Y. Halgand, "A survey of approaches combining safety and security for industrial control systems," *Reliability engineering & system safety*, vol. 139, pp. 156–178, 2015.

[15] P. Pradeep and K. Kant, "Conflict detection and resolution in iot systems: A survey," *IoT*, vol. 3, no. 1, pp. 191–218, 2022.

[16] D. Chaki, A. Bouguettaya, and S. Mistry, "A conflict detection framework for iot services in multi-resident smart homes," in *2020 IEEE International Conference on Web Services (ICWS)*. IEEE, 2020, pp. 224–231.

[17] ISO/DIS 10218-2:2020, "Robotics – safety requirements for robot systems in an industrial environment – part 2: Robot systems, robot applications and robot cells integration," March 2021.

[18] DIN EN ISO 12100:2010, "Safety of machinery – general principles for design – risk assessment and risk reduction," March 2011.

[19] ISO/TS 15066:2016, "DIN SPEC 5306 robots and robotic devices: Collaborative robots," 2017.

[20] A. A. Malik and A. Bilberg, *International Journal on Interactive Design and Manufacturing (IJIDeM)*, vol. 13, no. 4, pp. 1541–1547, 2019.

[21] DIN EN SO 10218-2:2011, "Din en iso 10218-2:2011: Robots and robotic devices – safety requirements for industrial robots – part 2: Robot systems and integration," Juni 2012.

[22] C. Fischer, M. Steiner, M. Neuhold, M. Papa, A. Markis, and S. Schlund, "An investigation of the measurement of transient contacts in human-robot interaction," in *International Conference on Robotics in Alpe-Adria Danube Region*. Springer, 2022, pp. 547–555.

[23] S. Hollerer, C. Fischer, B. Brenner, M. Papa, S. Schlund, W. Kastner, J. Fabini, and T. Zseby, "Cobot attack: a security assessment exemplified by a specific collaborative robot," *Procedia Manufacturing*, vol. 54, pp. 191–196, 2021.

[24] H. Kanamaru, "Bridging functional safety and cyber security of SIS/SCS," in *2017 56th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE)*. IEEE, 2017, pp. 279–284.

[25] A. G. Sutcliffe, N. A. Maiden, S. Minocha, and D. Manuel, "Supporting scenario-based requirements engineering," *IEEE Transactions on software engineering*, vol. 24, no. 12, pp. 1072–1088, 1998.

[26] S. Anwer, L. Wen, and Z. Wang, "A systematic approach for identifying requirement change management challenges: Preliminary results," *Proceedings of the Evaluation and Assessment on Software Engineering*, pp. 230–235, 2019.