2023 IEEE International Conference on Robotics and Automation (ICRA 2023)
May 29 - June 2, 2023. London, UK

Hazard Analysis of Collaborative Automation Systems: A Two-layer Approach based on Supervisory Control and Simulation

Tom P. Huck[1], Yuvaraj Selvaraj[2], Constantin Cronrath[2], Christoph Ledermann[1], Martin Fabian[2] *Senior Member, IEEE*, Bengt Lennartson[2] *Fellow, IEEE*, and Torsten Kröger *Senior Member, IEEE*

***Abstract*— Safety critical systems are typically subjected to hazard analysis before commissioning to identify and analyse potentially hazardous system states that may arise during operation. Currently, hazard analysis is mainly based on human reasoning, past experiences, and simple tools such as checklists and spreadsheets. Increasing system complexity makes such approaches decreasingly suitable. Furthermore, testing-based hazard analysis is often not suitable due to high costs or dangers of physical faults. A remedy for this are model-based hazard analysis methods, which either rely on formal models or on simulation models, each with their own benefits and drawbacks. This paper proposes a two-layer approach that combines the benefits of exhaustive analysis using formal methods with detailed analysis using simulation. Unsafe behaviours that lead to unsafe states are first synthesised from a formal model of the system using Supervisory Control Theory. The result is then input to the simulation where detailed analyses using domain-specific risk metrics are performed. Though the presented approach is generally applicable, this paper demonstrates the benefits of the approach on an industrial human-robot collaboration system.**

I. INTRODUCTION

An increasing number of automation systems interact with humans in safety-critical domains. International standards require that such safety critical systems must be subjected to a *hazard analysis* before commissioning [1], [2]. This is a design-time procedure to identify potentially unsafe system states before the system goes into operation. Hazard analyses are still often based on human reasoning and simple tools like checklists [3]. However, this does not scale well with increasing system complexity [4]. Testing under real-world conditions is also often infeasible (esp. in early development stages where prototypes are not available). Thus, the development of *model-based hazard analysis methods* is an active area of research. Model-based hazard analyses typically rely on *formal models* [5], [6], [7] or *simulation models* [8], [9], [10]. The differences are in the level of abstraction and the achievable degree of coverage [11]. Formal models are based on abstract mathematical-logical representations (e.g., automata) which often allow for exhaustive checks of the model's entire state space, resulting in guaranteed safety properties. However, physical safety aspects (e.g., collision geometries or forces) are frequently abstracted away to keep the analysis tractable. Simulation models also require some degree of abstraction, but are generally much more detailed than formal models. However, the higher degree of detail is commonly associated with larger state-spaces and more computational cost, which makes exhaustive analyses more difficult. In other words, there is a trade-off between *completeness* (i.e., finding all unsafe states in a model) and *accuracy* (i.e., achieving a good correspondence between model and real-world system).

We address this trade-off by proposing a two-layer approach that analyses the system on two abstraction levels. We use formal models to describe the system on an abstract level and infer from that a set of potentially unsafe system behaviours, which are then examined closer in simulation. We thereby combine the benefits from *exhaustive* analysis using formal methods and a *detailed* analysis using simulation. Behaviours that lead to potentially unsafe states are synthesised from the formal model using *Supervisory Control Theory* (SCT) [12]. Although SCT is a formal approach to synthesise feedback controllers for *discrete event systems* (DES), in this paper we re-purpose synthesis algorithms from SCT to identify system behaviours that lead to unsafe states. We aim to compensate limited detail of the formal model by modelling the system and safety specification conservatively, so that the synthesis yields an over-approximate set of unsafe behaviours. The potentially unsafe behaviours thus synthesised are then used as input for the simulation where they are analysed in more detail. In this second analysis step, it is determined whether the synthesised behaviours are indeed hazardous with respect to the system's original safety specification (i.e., the over-approximate set of unsafe behaviour is reduced to the set of actually unsafe behaviours).

While the approach is generally applicable, here it is demonstrated with examples from Human-Robot Collaboration (HRC). Our experiments show that even with relatively simple formal models, the two-layer approach finds significantly more unsafe behaviours than simulation-only methods. Our experiments also highlight the trade-off between modelling abstractions and accuracy. While this paper focuses on HRC systems, our approach is transferable, as its

*This work was supported by the German Federal Ministry for Economic Affairs and Climate Action under the project "SDM4FZI" (www.sdm4fzi.de), by FFI, VINNOVA under grant number 2017-05519 and under the ITEA3 "AITOC" project, and by the Wallenberg AI, Autonomous Systems and Software program (WASP) funded by the Knut and Alice Wallenberg Foundation. The support is gratefully acknowledged.

[1]Intelligent Process Automation and Robotics Lab, Institute of Anthropomatics and Robotics (IAR-IPR), Karlsruhe Institute of Technology, Germany. tom.huck@kit.edu

[2]Division of Systems and Control, Department of Electrical Engineering, Chalmers University of Technology, Gothenburg, Sweden.

foundational framework, SCT, is applicable to any system that can be modelled as a DES.

II. RELATED WORK

Many hazard analyses are still based on human reasoning and expert knowledge, supported by simple tools (e.g., checklists) [3], [1]. However, additional tools and methods have been developed. *Hazard and Operability Analysis* (HAZOP) [13], *HAZOP-UML* [14] and *Systems-Theoretic Process Analysis* (STPA) [4] are analysis methods which combine human reasoning with system diagrams (e.g. flow diagrams, control structure diagrams, UML-diagrams) and guide words for structured identification of hazards. *Expert systems* encode domain-specific knowledge in rule-bases that map sets of system properties to corresponding hazards and suggested safety measures [15]. *Formal methods* such as model checking express systems and safety specifications in strictly formalised representations (e.g., automata and linear temporal logic) so that systems can be automatically checked against their specifications [16], [17]. While model checking has traditionally been applied to check software and hardware designs, it can also be applied to identify hazards in cyber-physical systems like robots [5], [18], [19], [7].

For complex systems and systems with black-box components, formal methods may not be tractable. In these cases, simulation-based methods are an alternative. Simulation is frequently used for testing of safety-critical systems [20], especially in the domain of autonomous vehicles [21], [22], [23], but also in robotics [8], [9], [10] and aerospace [24].

There are also *hybrid approaches*, for instance simulation-based expert systems which rely partially on pre-defined rules (e.g. to calculate safety distances or safe velocities) and partially on 3D simulation (e.g. to visualise/compute spatial or physical aspects like hazardous zones or collision forces). Examples are the tools *CobotPlaner* [25] and *DynaRisk* [26].

Approaches combining formal methods and simulation have also been proposed in previous literature [27], [28]. The present paper differentiates itself from these prior works by two aspects: First, the formal methods are used as a light-weight pre-processing layer for the more detailed simulations rather than for a fully-fledged analysis. Second, this work uses SCT instead of conventional model checking approaches. The reasoning for this approach, as well as the associated advantages and limitations, will be discussed in more detail in Sec. VI.

III. TWO-LAYER HAZARD ANALYSIS

In this section we introduce our hazard analysis approach. Sec. III-A addresses the layer of DES. Sec. III-B discusses the simulation layer. For brevity, DES and SCT concepts are explained in a simplified, but intuitive manner. For detailed explanations, we refer to [12], [29], [30].

A. First Layer: Synthesis of Unsafe Behaviours

We model the collaborative system as a DES, which is a discrete-state, event-driven system that occupies at each time instance a single *state* out of its many possible ones, and transits to another state on the occurrence of an *event*. Thus, a characteristic feature of DES is the notion of instantaneous events that may be associated with common phenomena in a collaborative system, such as a collision or a human pressing a button. DES can be modelled and analysed in varying levels of detail [29].

As a DES modelling formalism, we use Extended Finite Automata (EFAs) [30]. An EFA extends finite automata with bounded discrete variables, guards (logical expressions) over the variables, and actions that assign values to the variables on the transitions.

Definition 1: An EFA is a tuple

$$E = \langle \Sigma, V, L, \rightarrow, L^i, L^m \rangle$$

where Σ is a finite set of events, $V = \langle v_1, v_2, \ldots, v_n \rangle$ is a finite set of bounded discrete variables, L is a finite set of locations, $\rightarrow \subseteq L \times \Sigma \times G \times A \times L$ is the conditional transition relation, where G and A are the respective sets of guards and actions, $L^i \subseteq L$ is the set of initial locations, and $L^m \subseteq L$ is the set of marked locations. Marked locations indicate desired states which must be reachable (more details about the significance of the marked states will follow later).

The current state of an EFA is given by its current location together with the current values of the variables. Let each variable $v_i \in V$ be associated with a bounded discrete domain $\hat{v}_i$ and $\hat{V} = \hat{v}_1 \times \hat{v}_2 \times \cdots \times \hat{v}_n$ be the domain of V. The set of states of an EFA is given by $Q = L \times \hat{V}$. The expression $l_0 \xrightarrow{\sigma:[g]a} l_1$ denotes a transition from location l_0 to l_1 labelled by event $\sigma \in \Sigma$, and with guard $g \in G$ and action $a \in A$. The transition is enabled when g evaluates to *true*. On occurrence of σ, a updates some of the values of the variables $v \in V$, thereby causing the EFA to change location from l_0 to l_1. Note that EFAs possess the same expressive power as finite automata (FA) and can be transformed into equivalent FA [30]. However, the richer structure of EFAs provide more compact models compared to FAs.

EFAs naturally interact through shared variables, but can also interact through shared events, which is modelled by *synchronous composition*. Common events occur simultaneously in all interacting EFAs, while local, non-shared events occur independently. $E_1 \parallel E_2$ denotes the synchronous composition of the EFAs E_1 and E_2. As defined by [30], transitions that are labelled by shared events but have mutually exclusive guards, or transitions that have conflicting actions, can never occur. This interaction mechanism provides an efficient way to model complex systems as a set of interacting EFAs in a modular way and to compositionally reason about their overall behaviour.

A DES model of the collaborative system is an abstraction of the feasible interactions between the subsystems, including those that lead to unsafe states. Ideally, the hazard analysis on this model should result in finding all behaviours that lead to unsafe states. In this paper, we use synthesis algorithms from SCT to identify such unsafe behaviours. SCT provides a framework for modelling, synthesis, and verification of reactive control functions for DES [12]. Given

a DES model of a system to control, the *plant* G, and a *specification* K of the desired controlled behaviour, SCT provides means to synthesise a *supervisor* that, interacting with the plant in a *closed-loop*, dynamically restricts the event generation of the plant such that the specification is satisfied.

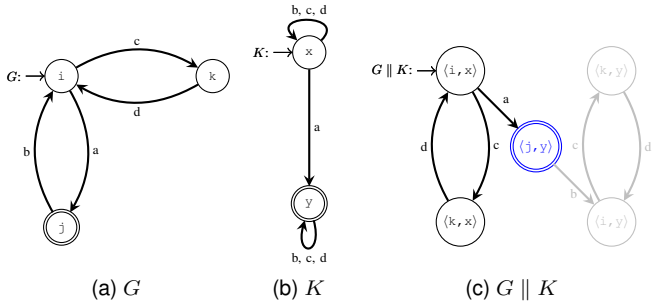
Supervisor synthesis is an iterative fixpoint algorithm that removes some behaviour from the synchronous composition $G \parallel K$ such that the resulting supervisor, and hence the closed-loop system, is guaranteed to fulfil certain properties, in addition to the specification [12], [29]. Two such properties relevant in our context are *non-blocking* and *minimally-restrictive*. The non-blocking property requires the supervisor to guarantee that the system can reach some marked state from any reachable state; and the minimally restrictive property requires the supervisor to minimise the behaviour that is removed. An illustration of the synthesis algorithm is given in Fig. 1, where the plant G and specification K are shown in Fig. 1a and Fig. 1b, respectively. The minimally-restrictive supervisor synthesised from $G \parallel K$ only removes those states and transitions that break the non-blocking property as shown in Fig. 1c. The software tool SUPREMICA [31] implements such synthesis algorithms and other techniques for modelling and analysis of DES.

For a supervisor synthesis problem, the specification typically describes *desired* behaviour through marked states in the DES model. Although it may seem counter-intuitive, we use marked states to specify *undesired*, i.e. hazardous, states of the collaborative system. Through synthesising a minimally-restrictive non-blocking supervisor, a subset of system behaviours is obtained. In this context, the non-blocking property means that the behaviours included in this subset reach a marked (and hence, unsafe) state. Minimal restrictiveness means that the supervisor yields an exhaustive set of *all* unsafe behaviours, and not only some. In this approach, marked states, such as human-robot collision states, may be specified based on human reasoning and expert knowledge, complemented by established hazard analysis checklists. Accordingly, such specification of marked states may inform what a suitable level of abstraction in the DES modelling of the collaborative system would be. The algorithmically generated set of corresponding hazardous behaviours can then be further analysed in simulation.

B. Second Layer: Simulation of Unsafe Behaviours

While the previous step already gives insights into potential hazards, certain safety-critical aspects are abstracted away in the DES model. Often, a definitive judgement whether a sequence is safe or unsafe is only possible from a more detailed analysis. We therefore perform a second analysis step in simulation. Note that this step is naturally domain-specific as each domain uses their own simulators. Thus, we only outline the general idea.

We create a simulation model where each system component, previously represented by an EFA, is now represented by a corresponding component in the simulation and each event from an EFA is associated with a certain activity



(a) G (b) K (c) $G \parallel K$

Fig. 1. Illustration of supervisor synthesis. The marked states are indicated by double circles. The grey states and transitions in $G \parallel K$ are disabled by the minimally-restrictive non-blocking supervisor.

of the respective component (e.g. an event from an EFA model of a human may correspond to a certain activity of a digital human model in the simulation). This correspondence allows for recreating the previously synthesised behaviours in simulation. However, directly recreating event sequences in a strictly sequential manner only adds limited value, since it does not consider the temporal nature of interactions between components in sufficient detail (e.g., whether a safety function responds quickly enough to avoid an unsafe state). The following example illustrates the necessity of the second simulation layer. Consider the following sequence:

$$(activateRobot, approachRobot, robotStops, enterWorkspace)$$

If we simulate this strictly sequentially, event by event, it will seem like the robot stops before the human enters the workspace. However, this may actually not be the case considering the robot's stopping time. Thus, we divide the events into a set of *proactive* and *reactive* events, where proactive events can occur without external stimuli, while reactive events are consequences of proactive events. We then extract from the synthesised behaviours only those events that are proactive and use them as input to control the simulation, while the reactive events are emergent reactions arising within the simulation. In the example above, the proactive events are $(activateRobot, approachRobot, enterWorkspace)$. These are used as inputs to drive the simulation. The reactive event $(robotStops)$ is not prescribed as a simulation input, but emerges internally in the simulation as a consequence of the behaviour encoded in the robot's simulation model. Assuming that this model is accurate, we can thus determine whether the robot responds safely considering its actual timing behaviour, and not the abstract behaviour.

IV. APPLICATION EXAMPLE

In the following, we present an example to illustrate our approach. Models and code are available on GitHub [32]. Consider a HRC workstation with formal models corresponding to Fig. 2 and a simulation model as depicted in Fig. 3. The collaborative task in this example is as follows: the human worker starts in the centre area (Fig. 3, A), retrieves a

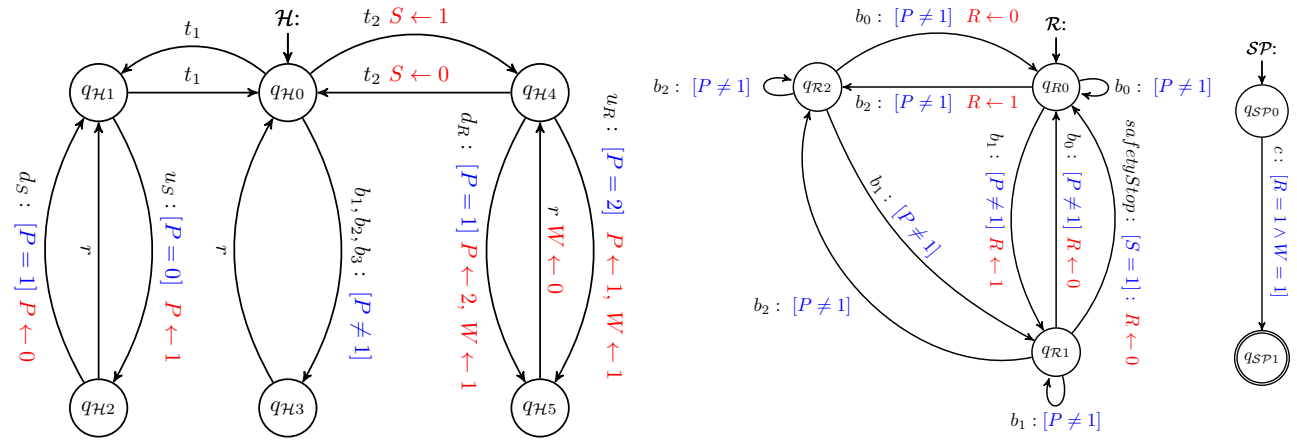


Fig. 2. EFA models of human worker $\mathcal{H}$ (left), robot $\mathcal{R}$ (middle), and safety specification $\mathcal{SP}$ (right). Guards are denoted in blue, actions in red. Marked locations are denoted by double circles. A transition can only be taken if the guard expression evaluates to *true*. When the transition is taken, variables are updated according to the action.

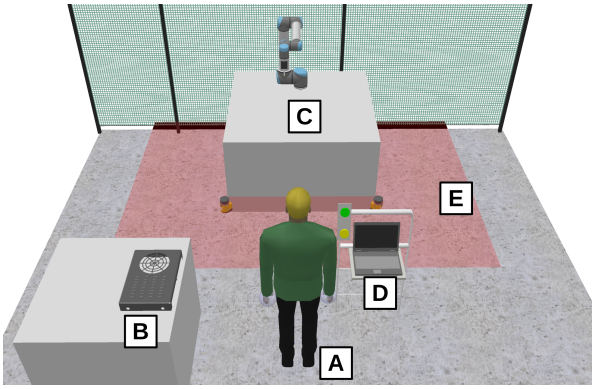


Fig. 3. HRC system from the application example (A: centre area, B: parts storage, C: robot station, D: control panel, E: laser scanner zone)

part from the table (B), places it in front of the robot (C), then walks back to the centre area (A), and activates the robot at the control panel (D). The robot performs a procedure on the part until the worker stops it through the control panel. The worker then retrieves the part and places it back on the table. As a safety measure, the area around the robot is monitored by a laser scanner (E, red area). A safety stop of the robot is triggered when the worker enters the detection area. To inspect the ongoing procedure from close distance without triggering a safety stop, the worker can override this safety function through the control panel. We now want to identify possible hazards in this HRC system. To that end, we model the system through EFAs, define a safety specification, and perform a supervisor synthesis to find unsafe behaviours.

The models are shown in Fig 2, with their locations (circles), events (arrows with black labels), guards (blue), and actions (red). For the the human $\mathcal{H}$, we introduce the following events: transiting between area A and B (t_1) and between A and C (t_2), as well as picking up/putting down the part at the storage table (up: u_S, down: d_S) and at the robot station (up: u_R down: d_R), respectively. Also, the human can press several buttons to stop or activate the robot, which are modelled as events b_0 (stop), b_1 (start in normal mode), and b_2 (start in safety override mode). The event r represents the human retracting the hand after a reaching motion. We introduce the variable $P \in \{0, 1, 2\}$ to track the part (0: at storage, 1: in worker's hands, 2: at robot station) and the variables $W \in \{0, 1\}$ and $S \in \{0, 1\}$ to track if the human currently occupies the shared human-robot workspace (W) and the laser scanner zone (S), respectively (0: not occupied, 1: occupied). Observe that some events require guard statements to be fulfilled before the transition is enabled. (e.g., to put down a workpiece at the robot station d_R, the guard is $P = 1$, because the worker must first be in possession of the part to put it down). For the robot $\mathcal{R}$, a location $q_{\mathcal{R}}$ is introduced for each operation mode ($q_{\mathcal{R}0}$: idle, $q_{\mathcal{R}1}$: working, $q_{\mathcal{R}2}$: working in safety override mode). Pressing the buttons (b_0, b_1, b_2) causes the robot to change its operation mode (note that b_0, b_1, b_2 are shared events appearing both in $\mathcal{H}$ and $\mathcal{R}$). Additionally, the robot has an event $safety\ stop$ which is enabled if the robot is running in normal operation mode and the laser scanner is occupied (i.e., $S = 1$). Note that $safety\ stop$ is not available in safety override mode (i.e., in $q_{\mathcal{R}2}$). The variable $R \in \{0, 1\}$ tracks if the robot is currently idle (0) or active (1). Note that in our models, the EFAs omit any information about the *duration* of events. For instance, in $\mathcal{R}$, it is not determined whether the safety stop is executed immediately as the human enters the detection zone, or if the robot requires some stopping time. It is only stated that $S = 1$ is a guard (i.e., a *precondition*) for the safety stop. This is a deliberate measure to compensate for loss of accuracy due to modelling abstractions: by leaving the timing open, we force the synthesis to consider all possible interleavings of events when searching for unsafe sequences. Whether these sequences are indeed possible under the actual timing behaviour of the system is determined in simulation, the second analysis step (cf. our discussion in Sec. III-B).

The safety specification $\mathcal{SP}$ simply has two locations $q_{\mathcal{SP}0}$ (safe) and $q_{\mathcal{SP}1}$ (unsafe). The unsafe location represents a

collision. Observe that the unsafe location is *marked*, but only reachable through event c (collision), which has the guard statement $R = 1 \wedge W = 1$. Thus, we consider a state to be a collision if the human is in the collaborative workspace and the robot is running at the same time (again, this is a conservative over-approximation to compensate for abstraction). We then perform a supervisor synthesis with respect to the plant $\mathcal{S} \parallel \mathcal{R}$ and the specification $\mathcal{SP}$. The resulting supervisor contains 90 states and 182 transitions.

In the simulation step of our analysis, we consider the events related to the human as proactive, and those related to the robot as reactive (compare Sec. III-B). We thus extract from the supervisor all human behaviours that lead to an unsafe state within a fixed time horizon of ten events, yielding 22 sequences (note that sequences in the supervisor can be arbitrarily long, as they may contain loops. To limit simulation time, we only extract sequences up to a fixed maximum length). One example for such an unsafe behaviour is the sequence $(b_2, r, t_1, u_s, r, t_1, t_2, d_R)$, where the human activates the robot in safety override mode, moves to the storage table, retrieves a part, moves back, then moves to the robot and places the part on the table, with the robot already running. With the robot in safety override mode, the event $safety\ stop$ is not enabled, thus leading to an unsafe state. We then simulate the extracted unsafe human behaviours to assess how hazardous they actually are when performed in conjunction with a more detailed and accurate model of the robot system. To quantify the level of danger without requiring a human user to inspect each simulation run, we compute a domain-specific *risk metric* r:

$$risk = \begin{cases} 0 & \text{case (a): } v_R < v_{crit} \\ \mathrm{e}^{-d_{HR}} & \text{case (b): } v_R \geq v_{crit};\ d_{HR} > 0 \\ \frac{F_c}{F_{\max}} + 1 & \text{case (c): } v_R \geq v_{crit};\ d_{HR} = 0\ . \end{cases} \quad (1)$$

where d_{HR} is the human-robot distance, v_R the robot speed, and v_{crit} a speed threshold (here: 250 mm/s). F_c is the estimated human-robot collision force, and F_{max} is a collision force limit according to [33]. The value of r is calculated in each simulation time step, and the maximum value is recorded for each simulated sequence.

V. EXPERIMENTS

A. Experimental Setup

We validate our approach by conducting experiments in two test scenarios, including the example presented above (referred to as scenario A), as well as two further scenarios B and C (see Fig. 4. For brevity scenarios B and C are not discussed here, but explanations can be found on GitHub [32]. For testing purposes, each of the HRC systems is deliberately designed to contain some safety-critical flaws. For instance, in scenario A (cf Sec. IV), there are two flaws: a delay in the robot's safety stop of the robot leads to a possible collision hazard and second, the safety override button allows the human to deactivate the safeguard, also leading to a collision hazard (although such an override button is unlikely to be found in a real robot system, we use it to introduce hazards for test purposes). We deploy our analysis described in Section III to find the sequences of events which lead to these unsafe states, using the tools SUPREMICA [31] for EFA modelling and supervisor synthesis, and *CoppeliaSim* [34] for simulation.

Performance criteria are as follows:

- N: The number of sequences found that lead to a collision state.
- r_{mean}: The mean risk value of the unsafe sequences.

For comparison, we also deploy two further approaches. In contrast to the approach above, these approaches search for unsafe event sequences *directly in simulation*, that is, without relying on a formal model as a first analysis step. In particular, we deploy random sampling, where events are sampled from a uniform distribution, and Monte Carlo Tree Search (MCTS). MCTS iteratively samples events, executes them in the simulator, and receives the resulting risk value (Eq. (1)) as a reward. By attempting to maximise the reward (and thus, the risk that is associated with a sequence), MCTS finds sequences leading to unsafe states. This approach has been introduced in our earlier work, and we refer to [10] for detailed explanations.

B. Test Runs and Results

Test runs for each of the three approaches (two-layer, MCTS, random) are executed with a maximum computational budget of 500 simulation runs. Each simulation run is limited to 12 (10) events in scenario A (B, C). Since the computation time for synthesising the supervisor (≤ 1 s) is negligible compared to the simulation time (approx. 30 min for 500 sequences), it is negligible in the budget. To limit the influence of statistical outliers due to randomised features (e.g., randomly sampled motion parameters), each test run is repeated ten times with different random seeds. Fig. 5 shows results averaged over the test runs for each approach.

In all scenarios, the two-layer approach found significantly more unsafe sequences than the simulation-only approaches. In terms of risk, results are relatively similar, with no

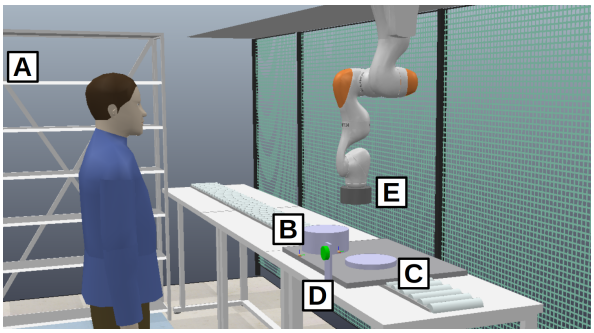


Fig. 4. Scenario B from the experiments. Here, the workflow is as follows: the worker retrieves parts from a shelf (A), inserts them into a housing (B) and activates the robot with a button (D) which then inserts a gearwheel (E) into the housing. Meanwhile, the worker inserts a part into the cover (C) and finally mounts the cover onto the housing. Potential hazards consist in the hand being crushed between gearwheel and housing, and the head colliding with the robot's elbow joint. Scenario C corresponds to scenario B, but features an additional light curtain between human and robot.

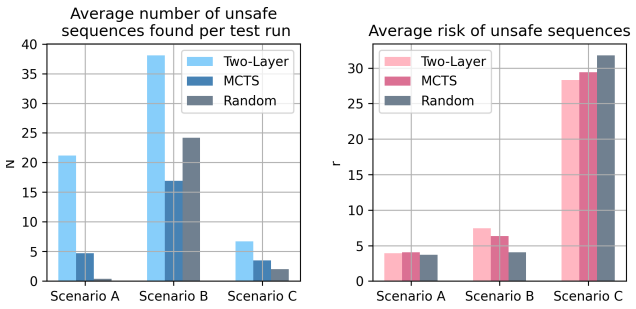


Fig. 5. Results from the test runs. Left: number of unsafe sequences found, right: average risk of unsafe sequences.

clear tendency discernible (probably because the achievable maximum risk is primarily limited by the maximum collision force, which depends on robot speed and mass).

Since safety specification and modelling on the first layer are more conservative than on the second, the first layer may raise ”false alarms”. A false alarm is an event sequences which appear to be hazardous in the formal model, but is not confirmed to be hazardous on the simulation level. Over all test runs, there were one false alarm in scenario A and 39 and 71 false alarms in scenarios B and C, respectively. It should be emphasized that false alarms, although undesired, are not critical from the users's perspective because they are filtered out in the second layer. The results in Figure 5 only contain ”true alarms”, that is, sequences which are confirmed to be hazardous in the second layer.

There is also the possibility of ”missed alarms”: A missed alarm occurs if the formal model erroneously indicates a sequence as safe, despite it being unsafe. The sequence is then not transferred to the second layer, although it would have turned out to be unsafe when simulated. Missed alarms are critical, because existing hazards are overlooked. In our experiments, a complete determination of the number of missed alarms is not feasible as this would require exhaustive simulation of all possible sequences, which is computationally too expensive. However, we can compare the unsafe sequences found by the simulation-only methods (i.e., MCTS and random sampling) with those found by the two-layer method. If a hazard is found by simulation-only methods but not by the two-layer method, this indicates missed alarms. In scenarios A and C, no missed alarms were found. In scenario B, however, eleven missed alarms were found.

VI. DISCUSSION AND FUTURE WORK

We demonstrated a novel hazard analysis approach with analyses on two distinct modelling layers. Contrary to conventional model checking approaches, we do not use the formal model as a stand-alone analysis tool, but as a preprocessing layer to find potentially critical event sequences for the second, simulation-based layer. The value of this approach is that it increases the efficiency of simulation-based testing compared to approaches that sample simulation sequences directly. With a detailed simulation as a second layer, the formal model can be abstract, simple and lightweight.

The main limitation of this approach lies in the dangers of modelling the system on a high abstraction level. Abstraction generally leads to a loss of accuracy that may lead to safe event sequences being misattributed as unsafe (”false alarms”) or unsafe event sequences being misattributed as safe (”missed alarms”). False alarms are undesired, but unproblematic, as they are filtered out in the second analysis layer. Missed alarms, however, are safety critical. We propose to mitigate the problem of missed alarms by adopting a conservative modelling approach on the first layer, so that the formal model is biased towards false alarms and over-approximates the set of unsafe behaviours. Yet, despite these precautions, missed alarms are not always avoidable as our experiments show. Furthermore, while our synthesis yields an exhaustive set of unsafe behaviours that are possible *within the model*, there is no guarantee that the model itself covers all behaviours that are critical in the real world.

On the other hand, one should keep in perspective that the aforementioned problems are not exclusive to the two-layer method. Simulation-only methods may also overlook hazards, either due to modelling omissions, or because limited computational budgets do not allow for exhaustive simulation of all possible sequences. After all, the two-layer method in our experiments still found more unsafe sequences than the simulation-only methods, despite the missed alarms.

Another point that merits discussion is why one should use SCT instead of established model checkers (e.g., *SPIN* [35]). In principle, the two-level approach would work with any formal verification tool. However, when unsafe states are found, conventional model checkers only return a single *error trace* (i.e., an example sequence leading to the unsafe state). The user then inspects the trace, fixes the underlying safety flaw, adapts the model, and re-runs the analysis. In our approach where false alarms are expected to occur frequently, this approach is not suitable: the user would need to inspect each error trace (including false alarms) and manually adapt the model, or the model checker might re-discover the same trace again. SCT, on the other hand, provides not a single error trace, but a full *set* of unsafe behaviours which can be transferred to the second level, where the false alarms are filtered out automatically without requiring user input.

In future, it may be worthwhile to explore splitting of the computational budget between the two-layer approach and simulation-only search. We will also consider introducing post-processing methods (e.g. clustering unsafe sequences), since multiple sequences may just be different manifestations of the same hazard. Another area for improvement is the modelling effort. While the analysis itself is automated, modelling is done by hand. This is time-consuming and error-prone and may outweigh the method's benefits. Finding ways to automatically create models or at least support the modelling procedure are therefore needed. In the long term, it may be also be interesting to consider the possibility of learning automata models from simulation [36], [37].

ACKNOWLEDGMENT

The authors thank Tamim Asfour for his support.

REFERENCES

[1] ISO, "ISO 12100:2011: Safety of machinery - General principles for design - Risk assessment and risk reduction," 2011.

[2] IEC, "IEC 61508-1:2010-1 Functional safety of electrical/electronic/programmable electronic safety-related systems - Part 1: General requirements," International Electrotechnical Commission, 2006.

[3] L. Hornung and C. Wurll, "Human-robot collaboration: a survey on the state of the art focusing on risk assessment," in *Berichte aus der Robotik - Robotix-Academy Conference for Industrial Robotics (RACIR) 2021*, Sep. 2021, pp. 10–17.

[4] N. Leveson, *Engineering a safer world: Systems thinking applied to safety*. MIT Press, 2011.

[5] M. Askarpour, D. Mandrioli, M. Rossi, and F. Vicentini, "SAFER-HRC: Safety analysis through formal verification in human-robot collaboration," in *International Conference on Computer Safety, Reliability, and Security*. Springer, 2016, pp. 283–295.

[6] M. Askarpour, L. Lestingi, S. Longoni, N. Iannacci, M. Rossi, and F. Vicentini, "Formally-based model-driven development of collaborative robotic applications," *Journal of Intelligent & Robotic Systems*, vol. 102, no. 3, 2021.

[7] M. Rathmair, C. Luckeneder, T. Haspl, B. Reiterer, R. Hoch, M. Hofbaur, and H. Kaindl, "Formal verification of safety properties of collaborative robotic applications including variability," in *2021 30th IEEE International Conference on Robot & Human Interactive Communication (RO-MAN)*. IEEE, 2021, pp. 1283–1288.

[8] D. Araiza-Illan, A. G. Pipe, and K. Eder, "Intelligent agent-based stimulation for testing robotic software in human-robot interactions," in *Proceedings of the 3rd Workshop on Model-Driven Robot Software Engineering*, 2016, pp. 9–16.

[9] P. Bobka, T. Germann, J. K. Heyn, R. Gerbers, F. Dietrich, and K. Dröder, "Simulation platform to investigate safe operation of human-robot collaboration systems," in *6th CIRP Conference on Assembly Technologies and Systems (CATS)*, vol. 44, 2016, pp. 187 – 192.

[10] T. Huck, C. Ledermann, and T. Kröger, "Virtual adversarial humans finding hazards in robot workplaces," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021.

[11] D. L. Dill, "What's between simulation and formal verification?" in *Proceedings 1998 Design and Automation Conference. 35th DAC.(Cat. No. 98CH36175)*. IEEE, 1998, pp. 328–329.

[12] P. J. Ramadge and W. M. Wonham, "The control of discrete event systems," *Proceedings of the IEEE*, vol. 77, no. 1, pp. 81–98, 1989.

[13] "IEC 61882:2016: Hazard and operability studies (HAZOP studies) - application guide," International Electrotechnical Commission, 2016.

[14] J. Guiochet, "Hazard analysis of human–robot interactions with hazop–uml," *Safety science*, vol. 84, pp. 225–237, 2016.

[15] R. Awad, M. Fechter, and J. van Heerden, "Integrated risk assessment and safety consideration during design of HRC workplaces," in *22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, Sep. 2017.

[16] C. Baier and J.-P. Katoen, *Principles of Model Checking*. MIT Press, 2008.

[17] E. M. Clarke, T. A. Henzinger, H. Veith, R. Bloem *et al.*, *Handbook of model checking*. Springer, 2018, vol. 10.

[18] M. Askarpour, D. Mandrioli, M. Rossi, and F. Vicentini, "Modeling operator behavior in the safety analysis of collaborative robotic applications," in *International Conference on Computer Safety, Reliability, and Security*. Springer, 2017, pp. 89–104.

[19] M. Askarpour, M. Rossi, and O. Tiryakiler, "Co-simulation of human-robot collaboration: From temporal logic to 3D simulation," in *1st Workshop on Agents and Robots for Reliable Engineered Autonomy, AREA 2020*, vol. 319. Open Publishing Association, 2020, pp. 1–8.

[20] A. Corso, R. Moss, M. Koren, R. Lee, and M. Kochenderfer, "A survey of algorithms for black-box safety validation of cyber-physical systems," *Journal of Artificial Intelligence Research*, vol. 72, 2021.

[21] J. Norden, M. O'Kelly, and A. Sinha, "Efficient black-box assessment of autonomous vehicle safety," *arXiv preprint arXiv:1912.03618*, 2019.

[22] W. Ding, B. Chen, M. Xu, and D. Zhao, "Learning to collide: An adaptive safety-critical scenarios generating method," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020.

[23] G. Chance, A. Ghobrial, S. Lemaignan, T. Pipe, and K. Eder, "An agency-directed approach to test generation for simulation-based autonomous vehicle verification," *arXiv preprint arXiv:1912.05434*, 2019.

[24] R. Lee, M. J. Kochenderfer, O. J. Mengshoel, G. P. Brat, and M. P. Owen, "Adaptive stress testing of airborne collision avoidance systems," in *2015 IEEE/AIAA 34th Digital Avionics Systems Conference (DASC)*. IEEE, 2015.

[25] F. IFF. (2023) Cobot Planner - Design Safe HRC Applciations Quickly and Easily (online tool). [Online]. Available: https://www.cobotplaner.de/preambel

[26] M. Bdiwi, "Intuitive Roboterprogrammierung und intelligente Werkzeuge," *JOT Journal für Oberflächentechnik*, vol. 62, no. 8, pp. 18–19, 2022.

[27] M. Webster, D. Western, D. Araiza-Illan, C. Dixon, K. Eder, M. Fisher, and A. G. Pipe, "A corroborative approach to verification and validation of human–robot teams," *The International Journal of Robotics Research*, vol. 39, no. 1, pp. 73–99, 2020.

[28] M. Askarpour, M. Rossi, and O. Tiryakiler, "Co-simulation of human-robot collaboration: From temporal logic to 3D simulation," in *1st Workshop on Agents and Robots for Reliable Engineered Autonomy, AREA 2020*, vol. 319. Open Publishing Association, 2020, pp. 1–8.

[29] C. G. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems*, 2nd ed. New York, NY, USA: Springer Science & Business Media, 2008.

[30] M. Skoldstam, K. Akesson, and M. Fabian, "Modeling of discrete event systems using finite automata with variables," in *2007 46th IEEE Conference on Decision and Control*. IEEE, 2007, pp. 3387–3392.

[31] R. Malik, K. Akesson, H. Flordal, and M. Fabian, "Supremica-An efficient tool for large-scale discrete event systems," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 5794 – 5799, 2017, 20th IFAC World Congress.

[32] T. Huck. (2023) Github repository. [Online]. Available: https://github.com/Huck-KIT/ICRA2023

[33] "ISO TS 15066:2016 Robots and robotic devices - Collaborative robots," International Organization for Standardization, 2016.

[34] E. Rohmer, S. P. N. Singh, and M. Freese, "Coppeliasim (formerly v-rep): a versatile and scalable robot simulation framework," in *Proc. of The International Conference on Intelligent Robots and Systems (IROS)*, 2013, www.coppeliarobotics.com.

[35] G. J. Holzmann, "The model checker SPIN," *IEEE Transactions on Software Engineering*, vol. 23, no. 5, pp. 279–295.

[36] Y. Selvaraj, A. Farooqui, G. Panahandeh, W. Ahrendt, and M. Fabian, "Automatically learning formal models from autonomous driving software," *Electronics*, vol. 11, no. 4, p. 643, 2022.

[37] A. Farooqui, P. Falkman, and M. Fabian, "Towards automatic learning of discrete-event models from simulations," in *2018 IEEE 14th International Conference on Automation Science and Engineering (CASE)*. IEEE, 2018, pp. 857–862.