
Proceedings Paper:
Lesage, Benjamin Michael Jean-Rene and Alexander, Rob orcid.org/0000-0003-3818-0310 (2021) SASSI: Safety Analysis using Simulation-based Situation Coverage for Cobot Systems. In: Proceedings of SafeComp 2021. Lecture Notes in Computer Science . , pp. 195-209.

https://doi.org/10.1007/978-3-030-83903-1_13

SASSI: Safety Analysis using Simulation-based Situation Coverage for Cobot Systems

Benjamin Lesage[1] and Rob Alexander[1]

Department of Computer Science, University of York, York, UK
first.last@york.ac.uk

Abstract. Assessing the safety of collaborative robot (cobot) systems is a difficult task due to the myriad of possible interactions between robots and operators, and the potential for injury to the operators. Using a situation coverage approach we can define the individual components of such interactions, and thereby describe the problem space and the coverage achieved when testing it. In this paper, we propose a situation coverage approach for testing the safety of a cobot system. Our approach suggests using a combination of safety analysis techniques and simulation-based testing to define situations of interest and explore hazardous situations while only endangering virtual operators. We challenge our assumptions by applying our method to an example based on a real-world use-case. The proposed metrics, if they provide no advantage to guided test generation techniques over random ones, helped us trim the generated configuration landscape to identify safety gaps.

Keywords: Cobot · Situation Coverage · Simulation-based testing · Safety Analysis.

1 Introduction

Cooperative robots (cobots) [4] aim to allow human operators and robot workers to share the same work-space, and work jointly to achieve a common goal. Because of safety concerns, however, users tend to build physical barriers to isolate robots from the operators [19, 26]. These barriers tend to limit the level of cooperation between human and robot, and reduce the advantages of deploying cobots in the industrial space.

The CSI: Cobot project [7] aims to reduce the need for barriers in cobot systems. The project proposes novel sensing and control techniques to improve cobots' awareness of their environment, especially regarding interactions with human operators. A crucial requirement for the adoption of new techniques is achieving confidence in the overall safety of the system. We consider safety aspects in the context of the CSI: Cobot project, in particular investigating the impact of changes in the system setup.

Simulation-based techniques, e.g. the CARLA simulator [8], are a common approach to evaluate autonomous systems [5, 29, 21]. Simulations allow fast iterations over varied configurations, including hazardous ones, without endangering the system itself, or its environment. However, the use of simulations

for testing safety constraints raises important issues. First, the safety case must ensure the simulated environment is representative of the system under consideration [2, 16]. Second, the tools must decide on a set of configurations (from the effective-infinite configuration space) to drive testing and evaluate confidence in the system safety [18].

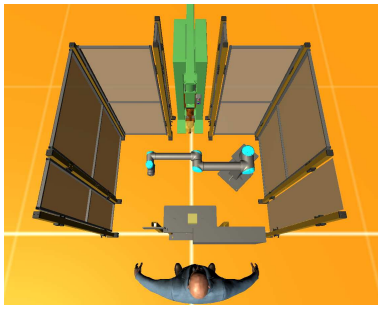
Our approach relies on simulation-based, situation-driven testing to evaluate the safety of a cobot system. It builds on the safety analysis of the cobot system under consideration, which identifies accidents and losses that arise from unsafe operation. The safety artefacts, generated as part of this analysis, capture undesirable situations which though not hazardous by themselves may lead to a loss. In the context of situation coverage [3], safety situations inform our exploration of the system configurations and provide for an evaluation of the confidence in the system safety.

We first introduce a manufacturing cobot use case to highlight our approach in Section 2. We then outline the general principles of our approach in Section 3. Section 4 discusses the identification of safety analysis artefacts, then monitored to identify hazards and guide the generation of simulation configuration, respectively in Section 5 and 6. Section 7 introduces our setup to assess the validity of our approach. We compare our approach to existing work (Section 8) before summarising our results in Section 9.

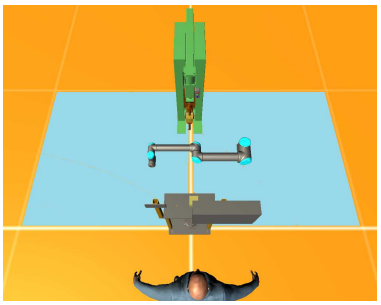
2 Case Study: Industrial manufacturing Cobot

We consider an industrial use case, defined in the context of the CSI: Cobot project [7], involving the cooperation of a human operator and a robotic arm to assemble small metal components ("assemblies"). Note that the general principles of our approach are not tied to the specific use case or tools we discuss in the following. The operator provides a non welded assembly at a designated work bench. The arm then retrieves the assembly and transports it to a spot welder for processing, before returning to the same work bench for a handover. The operator should keep out of the cell while the arm is active less he puts himself at risk of injury. All the processing currently occurs within a walled cage, with sensors to ensure no operator is present while the welder is active or the arm is moving. The cell is depicted in Figure 1a.

Our default setup is an abstraction of the industrial use-case, depicted in Figure 1b. All major components are in place, the welder, the cobot arm, and the shared bench. The highlighted space in the middle defines the cell region. The safety cage has been omitted. In lieu of a cage, a presence sensor stops the arm when the operator enters the cell. Due to constraints of our simulation environment, control of the arm and operator is limited with no exchange of assembly between them. The arm is programmed to loop between two waypoints from the bench at the bottom, to the welder at the top.



(a) Cell configuration with an operator (bottom), and an arm (centre) with a clear path to a welder (top).



(b) Setup under evaluation, with the cell region outlined (center) and the safety walls removed.

Fig. 1: Considered industrial use case configuration and evaluated setup.

3 Overview of the SASSI method

The SASSI method relies on the artefacts produced by the safety analysis. A key principle of our approach is to derive sufficient knowledge from the safety artefacts to guide the testing of the system. The objective is to understand if and how safety issues might arise in the system. Safety artefacts provide a safety-centric view of the entities involved in the system, their interactions, their relevant properties, and how those may lead to hazards; the safety analysis thus informs multiple aspects of the toolchain.

The system design is at the root of the process and it guides all further steps, defining the environment, its operating conditions, and safety requirements. Our method is building confidence into the system by testing it, searching for safety-relevant configurations, to provide feedback regarding safety aspects into the design. Figure 2 presents the overall workflow of the analysis method.

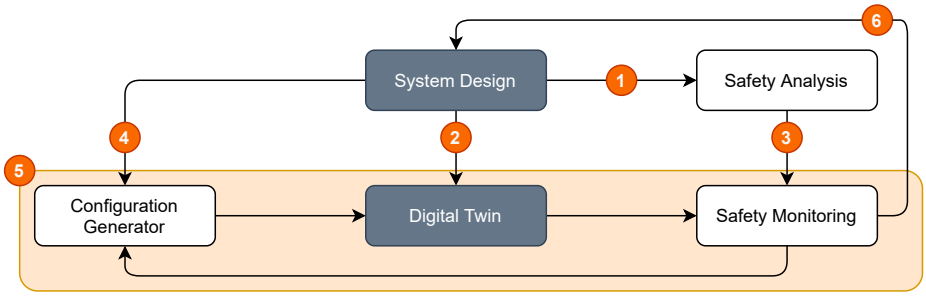


Fig. 2: Overview of the Analysis method

The first step is to perform a safety analysis of the system (1) to understand the occurrences of hazards in the system, and the conditions leading to such events (Section 4). The system design further informs the development of a simulation environment, the Digital Twin (2) which constitutes the baseline for our simulation-based approach. The Digital Twin also allows for the evaluation of system setups before their deployment in the actual cell. The artefacts

of the safety analysis provide information on safety-relevant situations in the system (3) and the conditions for their detection (Section 5). The system design further constrains acceptable configurations of the system. It defines elements open to variations and their degree of freedom, thus outlining the domain of our search (4). Safety artefact components observed running the simulation, from a generated configuration, provide feedback on situations of interest (5) during the search (Section 6). Finally, the coverage of generated and observed situations during analysis (6) provides some confidence in the safety of the system, or highlights shortcomings that need to be addressed.

4 Analysing the system safety

The Safety Analysis aims to understand the safety of the system, by identifying potential hazards, and the situations or causes leading to these events so they can be managed. The analysis needs to be aware of the components and entities interacting in the system as well as its operating environment. Without loss of generality, we present the Systems Theoretic Process Analysis (STPA) technique as the underlying safety analysis. The results of the application of STPA to our use case and our experience have been documented in [1].

STPA [20] originates from systems approaches to safety engineering. Accidents are assumed to arise from insufficient feedback or inadequate control in the system, as modelled by a control structure. The STPA analysis process is as follows:

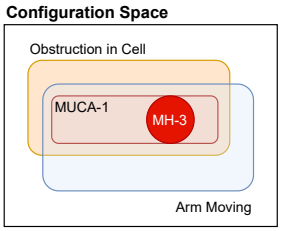
1. *Identify accident and loss scenarios*: these encompass a range of undesired events such as damage to property, injury to humans, or environmental pollution.
2. *Construct the system control structure*: the control structure captures the entities in the system and the flow of control and feedback between them.
3. *Identify unsafe control actions (UCAs)*: UCAs correspond to the execution of actions in undesirable configurations of the environment and the system.
4. *Identify causal factors and control flaws*: this step considers how unsafe actions arise as a result of inadequate control.

Our method relies on the artefacts produced by the STPA technique, notably hazards and unsafe control actions. Hazards identify events which by definition challenge the safety of the system. Unsafe control actions (UCAs), while not hazardous themselves, are undesirable. A UCA is an action the execution of which or lack thereof, in a given a configuration of the system, may give rise to a hazard. Each safety artefact thus captures a situation, a combination of components relevant to the safety of the system.

Our intuition is that monitoring for UCAs during testing can help identify hot spots for safety in the explored space of configurations, and focus the analysis effort on those regions more likely to result in hazards. The identification and monitoring of individual situation components can further guide the search towards UCA occurrences. Situation components finally outline a coverage target

for the search. Focus should be given to strategies which cover varied combinations of situations, as they provide more confidence in the safety of the system.

Example 1 *We consider the cell in Figure 1a and two situations captured by the safety analysis. Should the arm move while its path is obstructed (UCA **MUCA-1**), it may lead to a situation where a hazardous collision occurs involving the arm (Hazard **mH-3**). Configurations where there is an **Obstruction in cell** and the **Arm is moving** are more likely to trigger these safety-relevant situations.*



5 Monitoring safety artefacts

Monitoring aims to identify the set of safety-relevant situations that occurred during simulation. The simulation environment tracks the state of relevant components over time, producing a trace of events in the system. Runtime Verification methods [10] provide a vast array of tools and techniques to identify the violation of specific properties in a system from such a trace.

We use Fuzzy Linear Temporal Logical (LTL) [9, 12] to model safety artefacts. Our intuition is to measure how close an artefact is to occurring over time in a given configuration. An LTL formula captures a condition on the future of a path, combining predicates using logic, e.g. a or b, and temporal operators, e.g. d *eventually* occurs. The truth value of a predicate under fuzzy valuation [12] ranges between 0 (false) and 1 (true). As opposed to a crisp $true$ or $false$ valuation, fuzziness should provide search heuristics with a finer-grain metric to compare different configurations. We further extend value comparison operators with a tolerance. Once the compared values are in the tolerance range, the comparison valuation linearly tends towards 1 as they draw closer to each other.

Example 2 *Hazard MH-2 captures the situation where the arm exceeds its velocity restrictions in proximity of another entity. It is divided in two components.*

1. *the arm exceeds its velocity restriction:* $comp_s(t) = arm.velocity(t) \geq_{\mathcal{V}} PROX_V$
2. *the arm is in proximity of an entity:* $comp_d(t) = arm.distance(t) \leq_{\mathcal{D}} PROX_D$

$arm.velocity(t)$ and $arm.distance(t)$ respectively capture the velocity of the arm and its distance to the closest entity at instant t. Constant $PROX_V$ constrains the arm velocity in proximity of another entity, and $PROX_D$ defines the proximity distance threshold for the arm.

If the arm is in close proximity of an entity, $arm.distance(t) \leq PROX_D$, then the constraint is satisfied: $comp_d(t) = 1$. Conversely as per our fuzzy valuation rules, if the arm is far enough from the closest object, $\mathcal{D} + PROX_D < arm.distance(t)$, then constraint is not satisfied: $comp_d(t) = 0$. The valuation of $comp_d$ linearly increases as the difference between $arm.distance$ and $PROX_D$ decreases.

*MH-2 is satisfied for a given configuration if both conditions, $comp_s$ and $comp_d$, **eventually** occur at the same time. Under fuzzy logic, the **conjunction** and **always** temporal operator (respectively **disjunction** and **eventually**) are defined as the maxi-*

mum (respectively minimum) of the combined predicates. The valuation of MH-2 under configuration c can thus be computed as:

$$Occurs(\text{MH-2}, c) = \max_{t} \Big(min\big(comp_d(t), comp_s(t)\big) \Big)$$

The situations we monitor for are formalised from STPA artefacts, namely Hazards, UCAs, and their components. The textual description of each artefact needs to be translated into a formal monitor specification, as per Example 2, which might introduce errors in the process. We rely on a validation step to assess the suitability of our formal artefact specification. Each formalised artefact is tested against a number of crafted event traces, e.g. setting the value of $arm.velocity$ and $arm.distance$ at different instants t, and compared to expected truth values.

The simulation needs to expose the required information for monitoring. The safety analysis thus informs the design of the simulation not only in terms of the components and actions that should be modelled, but also regarding some of the events that need to be tracked. The safety monitors can be incorporated in the simulation tool itself, or applied to the output of the tool. We rely on a separate, offline monitoring approach. This allows the parallel development of the simulation environment, and the definition and validation of monitors. Artefacts are formalised into expressions in the Python language using the MTL library [25] for monitoring.

The safety monitors might rely on ground truth information that is unavailable or incorrect in the physical system due respectively to sensing gaps or faults. Conversely, some artefacts may need to be simplified for evaluation and monitoring, preferably subsuming the target condition to prevent false negatives at the cost of false positives.

Example 3 *In the absence of a physics model, a collision between the arm, the operator, or the assembly can be registered as soon as they connect, irrespective of their relative speed and mass.*

6 Generating and evaluating configurations

Automated testing techniques for autonomous robots need to address a number of challenges. The tools need to evaluate the confidence in the safety of the system to assess whether or not sufficient testing has been performed. They should also guide the testing strategy to produce situations that are interesting but also plausible and realistic.

Situation coverage [3] is a coverage criterion adapted to the testing of autonomous robots. The underlying principle is to identify components of the environment the system might encounter, identify how they can vary, and ensure that they, and combinations thereof, are evaluated during testing. High coverage speaks for the quality of the testing strategy and provides an indicator that the system has been considered in a variety of contexts. Testing of an autonomous vehicle would be required to navigate different types of road intersection with

various shapes, combined with the type of vehicles it might encounter at the intersection, their direction of travel, and any other reasonable factor.

Macro-level components define a requirement on the inputs and their combinations exercised during testing. Their identification is obviously informed by the system design which outlines reasonable operating configurations for the system. Micro situation components define the set of situations that should always, or never, be observed during the test campaign. The safety analysis provides such information, capturing the set of events the system should cope with, e.g. a human enters the cell during a welding operation.

We propose using the safety artefacts to guide testing. The occurrence of a safety artefact reflects negatively on the confidence in the safety of the system. We define a fitness metric to use as an objective function with state-of-the-art search heuristics. A configuration with a higher fitness triggers the occurrences of more artefacts and will be favoured by the search:

$$Fitness(c) = \sum_{h \in Hazards} W_H \times Occurs(h, c) + \sum_{u \in UCAs} W_U \times Occurs(u, c) \quad (1)$$

Where $Hazards$ and $UCAs$ are the set of hazards and UCAs identified by the safety analysis, $Occurs(s, c)$ is the fuzzy truth value capturing the occurrence of artefact s under configuration c (see Section 5), W_H and W_U weigh the occurrences of hazards and UCAs. We consider $W_H = 10$ and $W_U = 1$ in the following although weights can be adjusted per artefact, based on their severity.

To assess the coverage achieved by test campaign t, we consider the portion of safety artefacts triggered by the configurations encountered during testing:

$$ArtefactCoverage(t) = \frac{|\{s | s \in Artefacts \wedge \exists c \in t, Occurs(s, c) = 1\}|}{|Artefacts|} \quad (2)$$

Where $Artefacts = Hazards \cup UCAs$ is the set of safety artefacts identified by the safety analysis.

As the absence of observed safety artefacts is not a guarantee of safety, we further split artefacts into their individual components to compute coverage metrics. The components coverage is a measurement of the completeness of the testing with regards to safety components. A component or a combination thereof must be observed in all possible states to be considered as fully covered:

$$ComponentCoverage_N(t) = \frac{\left| \bigcup_{s \in \binom{N}{SC}} \bigcup_{c \in t} Occurs^N(s, c) \right|}{2^N \times |\binom{N}{SC}|} \quad (3)$$

Where $Occurs^N(s, c)$ captures the joint occurrences of components in s in configuration c. As an example consider components A and B. $Occurs^2(\{A, B\}, c)$ can contains at most 4 values, $A \wedge B$, $A \wedge \neg B$, $\neg A \wedge \neg B$, $\neg A \wedge B$. SC is the set of safety components derived from $Artefacts$, and $\binom{N}{SC}$ the set of N-length combinations of safety components. SC is built from the safety artefacts by automatically collecting the predicates and comparisons in their formulation, e.g. $comp_d$ and $comp_s$ in Example 2.

7 Evaluation

The SASSI method outlines general principles for testing a cobot system for safety. We rely in particular on safety artefacts to capture safety-relevant situation components. Those components guide the search through the configuration space for interesting scenarios, and provide an evaluation of the relevant coverage of the situation space. This section challenges our intuitions:

- *Research Question 1.* Are UCA necessary for hazards to occur?
- *Research Question 2.* Are situation-based heuristics a good guide for testing?

7.1 Problem space

We identified in Table 1a the artefacts monitored by the simulation. We simplified some conditions w.r.t. to the original artefacts to cope with limitations of the simulation environment, as suggested in Section 3. The configuration variables in Table 1b outline the configuration domain and search space.

Table 1: Configuration of the simulation for evaluation

(a) Safety artefacts monitored by the simulation.

Id	Event
MH-1	The arm exceeds its velocity restriction in either region.
MH-2	The arm exceeds its velocity restriction in proximity of another entity.
MH-3	The arm, assembly, or operator is compromised because of a collision.
MUCA-1	The arm moves while the cell is obstructed.
MUCA-2	The arm stops moving before it reaches its target position.

(b) Definition of the problem search space

Variable	Type	Semantic
Arm position	$(i, j, k) \in \mathbb{R}^3$	Where the arm is in the cell, outlined in Figure 1b
Operator position	$(x, y, z) \in \mathbb{R}^3$	Where the operator is in the cell or outside
Restrict Velocity	$(b \in \mathbb{B})$	Likely velocity threshold violations if unset

7.2 Simulation setup

We discuss the integration for our evaluation with the Digital Twin developed in the CSI: Cobot project [7]. The simulation captures a model of the system and its behaviour. It is considered as a black box in our approach, used to evaluate the system's response to varied situations without any risk to its actors. The simulation needs to satisfy the safety monitoring and configuration requirements.

The simulation scene is setup as described in Section 2. The position of the operator is fixed during each run, and the safety stop will be prevent the arm from moving if the operator is inside the cell. Collisions are recorded as soon as the arm and operator connects. Virtual sensors, with no pendent in the physical system, provide for ground truth information regarding the velocity of the arm and its proximity to other entities, namely the operator. The cell region is defined as a volume which monitors entities entering or leaving.

The processing framework for our analysis focuses on interactions with the Digital Twin (illustrated by Step 5 in Figure 2), ensuring valid configuration files are generated and observations can be processed to identify situations of interest. The Digital Twin provides hooks to configure pre-existing entities in the simulated environment. All entity properties can be controlled through a unified configuration file. Communication between independent actors within the twin occurs through message passing, similarly to the real platform. Listeners capture all cross-entity messages, functional or safety-related, issued during a run into in a unified record.

Example 4 *The cell region is defined as a safety region volume which monitors entities entering or leaving the volume. The arm safety stop discussed in Section 2 reacts to unexpected entrances upon receiving the corresponding message. The same message provides for the computation of the "obstruction in cell" component during safety monitoring.*

Our toolchain supports the generation of configuration files for the Digital Twin, exposes primitives to process the messages recorded during a run, and allows for the evaluation of properties on the resulting observations. We configure the process by defining the domain for the search, the set of safety artefacts under evaluation, and the conversion from messages to variables required for artefact evaluation such as *arm.distance* in Example 2. The tools automatically extract the required safety components, and the corresponding coverage and fitness metrics for a run. Combined with a search heuristic, they provide the automated evaluation of the system.

7.3 Search heuristics

We consider a number of heuristics to explore the configuration space. All heuristics were provided with the same budget of 1000 runs and the same input domain.

Ran randomly explores the configuration space, generating solutions within the provided constraints. It is used as a baseline for comparison against guided approaches.

GA-max (resp. **GA-min**) uses an elitist genetic algorithm [27, 24] to maximise (resp. minimise) the fitness metric introduced in Section 6. Genetic algorithms operate on a population of configurations by selecting, and mutating the best individuals. The two configurations respectively aim to maximise and minimise the *concurrent* occurrences of safety artefacts in the system.

Quality diversity algorithms (**QD**) [22, 6] use a similar evolutionary approach. An archive of solutions is maintained and the search aims for both diversity, i.e. illuminating the archive, and individual performance, i.e. best in each niche. We consider two variations of the **QD** configuration. **QD-NS**, removing the safety stop if an operator is in the cell, increases the likelihood of collisions. **QD-NF** does not rely on fuzzy valuation for safety artefacts, only 1 for *true* and 0 for $false$, disrupting the search by reducing the granularity of the fitness metric.

We extract two features to define the niche covered by a configuration: the highest-ranked observed hazard, and the highest-ranked observed UCA. Hazards

and UCAs are ranked by their identifier in the STPA analysis. This is a compromise over more accurate features, e.g. capturing the set of observed hazards, to keep the size of the QD archive reasonable. The heuristic is set to minimise the fitness metric in each niche. This configuration aims to observe a variety of safety artefacts and conditions leading to hazards in isolation of each other.

7.4 Results

We challenge our intuition on the feasibility of our approach and its underlying assumptions. Our work focused on building the Digital Twin, and the required tooling to allow for testing for safety using the proposed approach. The results in this section thus focus on state of the art heuristics for test generation[1].

Are UCAs necessary for hazards to occur? We first evaluate our hypothesis of a relation between the occurrences of UCAs and Hazards outlined in Example 1, that is the occurrence of a UCA, as captured by the safety analysis, is necessary for the occurrence of a hazard. Focus of testing effort on configurations where UCAs (or components thereof) occur would increase the likelihood of discovering latent hazards.

We consider all 6000 configurations generated during our test campaigns, with the operator, arm and assembly in random states, irrespective of the heuristic. Each configuration is run through the simulation for a full cycle, the arm traversing all its way points. Processing the simulation output classifies runs into ones where no safety situation occurred, either solely UCAs or Hazards occurs, or both artefact types occur. The distribution of runs across these categories is presented in Figure 3.

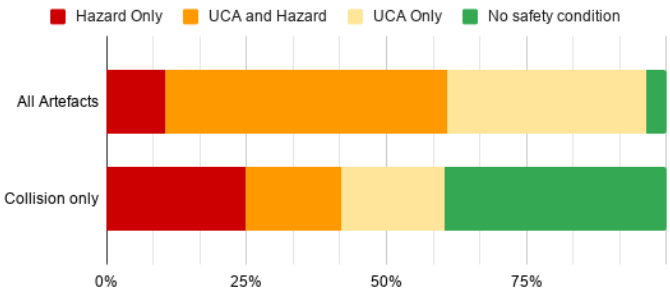


Fig. 3: Classification of situations detected across 6000 generated configurations.

We observed multiple cases where a hazard occurred without a UCA being observed as well. Such observations occurred under all heuristics. These results highlight gaps in our monitoring and safety analysis. None of the monitored UCAs in the current setup relates to the velocity of the arm, and there is no indicator that the related hazards might occur. Our initial safety analysis, at a

[1] Each batch of 1000 runs took on average 5 hours to complete on a 1.8GHz i5 laptop.

larger scale, does capture velocity-related UCAs. However those are confined to specific, un-modelled actions of the arm, e.g. during a handover, and they still do not cover all occurrences of a hazard.

Under the "Collision only" row in Figure 3, we focus on monitored MH-3 (a collision with the arm), and MUCA-1 (a moving arm in an obstructed cell). Our heuristics discovered solutions where the operator collided with an immobile arm (MH-3 and ¬ MUCA-1). The operator moving was not identified as a control action in our STPA analysis, given the lack of a controller on the operator in our model. As such, no related unsafe control action was identified.

We identified reasonable hazard occurrences without a related UCA. Regions where UCAs occur do result in a high likelihood of safety hazards. However, a good heuristic should consider the hazard situation components on their own to guide its result. Automated testing and classification of such occurrences helped us identify gaps in our safety analysis and monitoring, despite the explored scene and configuration space abstracting our use case.

Are situation-based heuristics a good guide for testing? We now consider the benefits of safety testing guided by situation-based metrics, in particular the occurrence of safety artefacts. All heuristics, except for **Ran**, use some form of feedback to identify the best configurations, and direct the search to maximise or minimise such occurrences. We compare in Table 2 the configurations generated by each heuristic, their artefacts and components coverage, and the combinations of such components they encountered.

Table 2: Coverage and fitness achieved under the different search heuristics

Heuristic	Artefact	Component (1)	Component (2)	(Min, Max) Fitness	# Niches
Ran	100%	75%	54.82%	(1, 32)	6
GA-min	100%	75%	54.10%	(1, 31)	6
GA-max	100%	75%	54.69%	(1, 32)	6
QD	100%	75%	54.82%	(1, 32)	6
QD-NS	100%	75%	54.96%	(14.64, 32)	6
QD-NF	100%	75%	54.82%	(0, 31)	6

The results across heuristics are very similar. All heuristics managed to trigger all monitored artefacts in our evaluation (Artefact), cover similar combinations of them (# Niches), and the same configurations of situation components considered on their own (Comp. (1)). Considering concurrent occurrences of the safety components (Comp. (2)), the differences between heuristics remain marginal with a slight benefit for **QD**-based ones, **Ran**, then **GA**. Our simple setup has a high rate of safety occurrences, and a better comparison would be provided by a less hazard-prone setup.

No heuristic achieved a 100% coverage of the component-based coverage metrics. This is the result of gaps in the coverage, and infeasible combinations of components. As an example, three distinct components consider the position of the operator namely, in the cell, at the bench, and at the welder. Full coverage of the Comp. (2) metric would require observing the operator as being at the bench and the welder at the same time. This is infeasible under the current configuration space: the operator cannot lay down across the cell to reach both regions.

Unless he climbs onto the welder, the operator will also always be considered in the cell when at the welder, $(at\_welder, in\_cell) = (True, False)$ is infeasible.

Comparing the best and worst fitness encountered by each method provides a similar insight. All methods managed to trigger all artefacts in a single configuration (a fitness of 32), although across different configurations. Evidence suggests the use of a presence sensor did reduce the likelihood of collisions between the operator and the arm, as exemplified with the high minimum fitness under **QD-NS**. Only the **QD-NF** managed to observe a safe configuration, with all configurations coming close with only a single UCA triggered.

All heuristics exploring our cage-less use-case have shown evidence of hazardous configurations. Some occurrences may be attributed to limitations of the simulated environment and setup, but they still highlight safety issues in the system. As identified in the previous section, there are few mechanisms to prevent or identify velocity constraint violations, except assessing the configuration of the arm controller, as outlined by the lack of related UCA. These could be mitigated by external control or sensing.

The proposed metrics provide little benefit to guided search heuristics, but they can help trim the configurations that need to be reviewed following an initial assessment. As an example focusing on the individuals in each of the niches discovered by **QD**, we identified a safety issue in our setup where the arm could reach outside of the cell and collide with the operator. Similarly, the arm in the confines of the cell could still collide with an operator at the bench.

8 Related Work

Combinatorial or Design of Experiments (DoE) methods [15] scope the configuration space to generate a set of complete or partial covering experiments. Gleirscher [14] suggests the use of model-based techniques to generate tests ensuring hazardous states of the system cannot be reached. Yu et al. [28] automatically derive such a model from the safety analysis results. Our approach does not rely on a model or a-priori knowledge on which situations a specific configuration triggers. We use state-of-the-art search heuristics, guided by safety-relevant metrics, to identify hazardous states.

Fontaine et al. [11] successfully propose the use of quality diversity algorithms [22] to discover failure scenarios in a human-robot shared autonomy problem. Other simulation-based approaches for testing automated vehicles show similar promising results [23]. These approaches tend to rely on metrics tailored to the modelled interaction. Our metrics instead derive from safety analyses, and fuzzy evaluation provides for a gradual progression between an unlikely and guaranteed occurrence. Neither property requires domain knowledge outside of the safety analysis.

Metrics such as MCDC [13] focus on code coverage, ensuring as an example the absence of dead code. However structural metrics are not adapted to evaluating functional requirements even more so the emergent behaviours that may arise from autonomous robots [17]. We propose instead the use of safety-centric

coverage metrics, based on the observed situations, with a similar division of the decision/situation into its condition/components.

9 Conclusion

We proposed a method to assess the safety of a cobot system, relying on the artefacts captured during safety analysis to (1) inform the design of a simulation-based environment, (2) guide search heuristics towards unsafe configurations, and (3) assess confidence in the observed configurations. Our initial evaluation highlighted genuine safety issues in our setup. It shows a high hazard likelihood, and gaps in our safety analysis lead to hazards without early warning signs.

A strong safety baseline, increasing the challenge of triggering hazardous behaviours, would be required to better assess the benefits of search heuristics guided by safety components. This would also provide a fair evaluation of the proposed coverage metrics. Further refinements of said coverage metrics to trim infeasible combinations of safety components would provide improved coverage and confidence.

Acknowledgements

This project is supported by the Assuring Autonomy International Programme, a partnership between Lloyd's Register Foundation and the University of York.

We would like to thank the project partners at the University of Sheffield the University of York for numerous insightful discussions. In particular, we would like to thank James Douthwaite for his expertise with the Digital Twin.

References

1. AAIP Body of Knowledge: 1.2.1 Considering human/machine interactions. https://www.york.ac.uk/assuring-autonomy/body-of-knowledge/required-behaviour/1-2/1-2-1/cobots/, accessed: 2021-02
2. Afzal, A., Le Goues, C., Hilton, M., Timperley, C.S.: A study on challenges of testing robotic systems. In: 2020 IEEE 13th International Conference on Software Testing, Validation and Verification (ICST). pp. 96–107. IEEE (2020)
3. Alexander, R., Hawkins, H.R., Rae, A.J.: Situation coverage–a coverage criterion for testing autonomous robots. Tech. rep., University of York (2015)
4. Bauer, A., Wollherr, D., Buss, M.: Human–robot collaboration: a survey. International Journal of Humanoid Robotics **5**(01), 47–66 (2008)
5. Bobka, P., Germann, T., Heyn, J.K., Gerbers, R., Dietrich, F., Dröder, K.: Simulation platform to investigate safe operation of human-robot collaboration systems **44** (2016), 6th CIRP Conference on Assembly Technologies and Systems (CATS)
6. Cazenille, L.: Qdpy: A python framework for quality-diversity. https://gitlab.com/leo.cazenille/qdpy (2018)
7. CSI: Cobot. https://www.sheffield.ac.uk/sheffieldrobotics/about/csi-cobots
8. Dosovitskiy, A., Ros, G., Codevilla, F., López, A.M., Koltun, V.: CARLA: an open urban driving simulator. CoRR **abs/1711.03938** (2017), http://arxiv.org/abs/1711.03938

9. Emerson, E.A.: Temporal and modal logic. In: Formal Models and Semantics, pp. 995–1072. Elsevier (1990)
10. Falcone, Y., Krstić, S., Reger, G., Traytel, D.: A taxonomy for classifying runtime verification tools. In: International Conference on Runtime Verification (2018)
11. Fontaine, M., Nikolaidis, S.: A quality diversity approach to automatically generating human-robot interaction scenarios in shared autonomy (2021)
12. Frigeri, A., Pasquale, L., Spoletini, P.: Fuzzy time in LTL. CoRR **abs/1203.6278** (2012), http://arxiv.org/abs/1203.6278
13. Ghani, K., Clark, J.A.: Automatic test data generation for multiple condition and mcdc coverage. In: 2009 Fourth International Conference on Software Engineering Advances. pp. 152–157. IEEE (2009)
14. Gleirscher, M.: Hazard-based selection of test cases. In: Proceedings of the 6th International Workshop on Automation of Software Test. pp. 64–70 (2011)
15. Grindal, M., Offutt, J., Andler, S.F.: Combination testing strategies: a survey. Software Testing, Verification and Reliability **15**(3), 167–199 (2005)
16. Guiochet, J., Machin, M., Waeselynck, H.: Safety-critical advanced robots: A survey. Robotics and Autonomous Systems **94**, 43–52 (2017)
17. Helle, P., Schamai, W., Strobel, C.: Testing of autonomous systems–challenges and current state-of-the-art. In: INCOSE international symposium. vol. 26, pp. 571–584. Wiley Online Library (2016)
18. Huck, T.P., Ledermann, C., Kröger, T.: Simulation-based testing for early safety-validation of robot systems. In: 2020 IEEE Symposium on Product Compliance Engineering-(SPCE Portland). pp. 1–6. IEEE (2020)
19. Robotics — Safety requirements for robot systems in an industrial environment — Part 1: Robots. Standard, International Organization for Standardization (2011)
20. Leveson, N.G., Thomas, J.P.: STPA Handbook. Cambridge, MA, USA (2018)
21. Norden, J., O'Kelly, M., Sinha, A.: Efficient black-box assessment of autonomous vehicle safety. arXiv preprint arXiv:1912.03618 (2019)
22. Pugh, J.K., Soros, L.B., Stanley, K.O.: An extended study of quality diversity algorithms. In: Proceedings of the 2016 on Genetic and Evolutionary Computation Conference Companion. p. 19–20. GECCO '16 Companion, ACM (2016)
23. Riedmaier, S., Ponn, T., Ludwig, D., Schick, B., Diermeyer, F.: Survey on scenario-based safety assessment of automated vehicles. IEEE Access **8**, 87456–87477 (2020)
24. Solgi, M.: geneticalgorithm: a python library for elitist genetic algorithm. https://github.com/rmsolgi/geneticalgorithm (2020)
25. Vazquez-Chanlatte, M.: mvcisback/py-metric-temporal-logic: v0.1.1 (Jan 2019). https://doi.org/10.5281/zenodo.2548862
26. Villani, V., Pini, F., Leali, F., Secchi, C.: Survey on human–robot collaboration in industrial settings: Safety, intuitive interfaces and applications. Mechatronics **55**
27. Whitley, D.: A genetic algorithm tutorial. Statistics and computing **4**(2) (1994)
28. Yu, G., wei Xu, Z., wei Du, J.: An approach for automated safety testing of safety-critical software system based on safety requirements. In: 2009 International forum on information technology and applications. vol. 3, pp. 166–169. IEEE (2009)
29. Zou, X., Alexander, R., McDermid, J.: Testing method for multi-uav conflict resolution using agent-based simulation and multi-objective search. Journal of Aerospace Information Systems **13**(5), 191–203 (2016)