Available online at www.sciencedirect.com

ScienceDirect

Procedia CIRP 00 (2021) 000–000

www.elsevier.com/locate/procedia

32nd CIRP Design Conference

Situational Risk Assessment Design for Autonomous Mobile Robots

Manuel Müller[a], Golsa Ghasemi[a,*], Nasser Jazdi[a], Michael Weyrich[a]

[a]*University of Stuttgart, Pfaffenwaldring 47, 70550 Stuttgart, Germany*

* Corresponding author. Tel.: +49-711-685-67320. Fax: +49-711-685-67302. *E-mail address:* golsa.ghasemi@ias.uni-stuttgart.de

Abstract

The emerging autonomous mobile robots promise a new level of efficiency and flexibility. However, because these types of systems operate in the same space as humans, these mobile robots must cope with dynamic changes and heterogeneously structured environments. To ensure safety new approaches are needed that model risk at runtime. This risk depends on the situation and that is a situational risk. In this paper, we propose a new methodology to model situational risk based on multi-agent adversarial reinforcement learning. In this methodology, two competing groups of reinforcement learning agents, namely the protagonists and the adversaries, fight against each other in the simulation. The adversaries represent the disruptive and destabilizing factors, while the protagonists try to compensate for them. The situational risk is then derived from the outcome of the simulated struggle. At this point, the system's Digital Twin provides up-to-date and relevant models for simulation and synchronizes the simulation with the real asset. Our risk modeling differentiates the four steps of intelligent information processing: sense, analyze, process, and execute. To find the appropriate adversaries and actors for each of these steps, this methodology builds on Systems Theoretic Process Analysis (STPA). Using STPA, we identify critical signals that lead to losses when a disturbance under certain conditions or in certain situations occurs. At this point, the challenge of managing the complexity arises. We face this issue using training effort as a metric to evaluate it. Through statistical analysis of the identified signals, we derive a procedure for defining action spaces and rewards for the agents in question. We validate the methodology using the example of a Robotino 3 Premium from Festo, an autonomous mobile robot.

Keywords: Situational Risk Assessment; Digital Twin; Design Approach; Complexity; System Theoretic Process Analysis (STPA)

1. Introduction

The emerging autonomous mobile robots promise a new level of efficiency and flexibility. They are used to connect the respective workstations flexibly and allow for dynamically adding, removing, and changing manufacturing processes. They do not need specific work areas but share the same floor with other vehicles and workers. Because these types of systems operate in the same space as other vehicles and humans, these mobile robots have to cope with dynamic changes and heterogeneously structured environments. Therefore, they rely on the characteristics of autonomous systems, namely "(1) systematic process execution, (2) adaptability, (3) self-governance and (4) self-containedness" [1], and artificial intelligence. However, in order to ensure safety, new design patterns are required. The risk of these systems is no more predominantly defined by the reliability of the components but rather on the scenario and how well the system is prepared to handle it. Therefore, the safety evaluation has to focus on the situation the system is in rather than globally trying to rate the reliability and conclude to safety – a paradigm shift.

In this paper, we propose a novel design approach as a situational risk assessment allowing for runtime risk assessment based on the situation, the system is currently in. We use the idea of system theoretic process analysis and derive reinforcement learning-based fault injectors, so-called adversaries. These adversaries learn to apply worst-case disturbances reasonable for the respective situation and challenge the system in simulation. If the system is able to survive these disturbances, it can be considered safe. In order to execute this simulation, we exploit the Digital Twin with its main properties: synchronization with the asset, model orchestration, and simulation capability [2]. In this way, we

ensure that the simulation is always up-to-date and represents the real system and its environment.

Section 2 introduces several related works on this topic. We continue in Section 3 with the design pattern for the adversaries and explain how to choose the appropriate ones and train them. In Section 4, we tackle the topic of complexity management. Then, we present a case study in Section 5 and finally draw the conclusion and outlooks in Section 6.

2. Background

In the field of safeguarding autonomous systems, there are mainly three approaches: formally proving the absence of critical behavior, trying to improve robustness and to show superhuman performance, and analyzing the autonomous systems in certain scenarios. One approach of the first category is working with set-based methods like [3] claiming to provide provable safe navigation. However, formal proving requires strong assumptions and is computationally expensive. The second approach is taken especially in the domain of reinforcement learning. The community celebrated breakthroughs in superhuman performance in playing Go [4]. One particularly interesting approach is the auto-curriculum [5], where two types of reinforcement learning agents, called protagonist and adversary play against each other in order to improve self-supervised. This approach was transferred to the technical domain, building robust adversarial reinforcement learning [6]. Several approaches extend this original idea, e.g. using model-checking-based falsification as adversaries [7], considering model uncertainties [8], or even including the policy's variance into the optimization process [9]. However, as studies from the autonomous driving domain show, it is hard to show superhuman performance in dynamically changing and heterogeneously structured environments [10]. The design approach we propose belongs to the third category. Conventionally as addressed in the PEGASUS project, a list of relevant scenarios is defined and tested [11]. As Hata et al. [12] propose, fuzzy methods combined with deep learning-based semantic segmentation helps interpolating between scenarios. However, if trying to make a claim for any possible scenarios, similarly to the proving approach strong assumptions are needed.

The complex environment regarding the higher complexity of input data, complex software due to required complex logic, and non-deterministic behavior all factors show the importance of analyzing the complexity of safety [13]. So, the unpredictability of the behavior of autonomous systems causes a complexity increase of simulation techniques [14] Making complex decisions and meeting many requirements brings a new class of complexity problems for software of autonomy [15]. In [16], runtime validation is proposed as an approach for addressing the complexity problem.

We, therefore, follow a novel approach. As [17] suggests, we determine the risk at runtime based on the situation, the system is currently in. As this is a concrete situation, we just need to consider scenarios, which can arise from the given situation, limiting it to a handy set. However, in contrast to [17], the proposed design pattern aims to set up a reinforcement learning-based system that derives the risk from the outcome of simulated development of the current situation rather than a rule-based approach. Moreover, like [18], we determine the risk in a probabilistic sense. Our overall architecture is presented in [19]. Our design approach is based on the idea of environment-centric judgment starting from control-theoretic modeling [20]. Therefore, we use the System Theoretic Process Analysis [21] rather than component-centric safety analysis. Moreover, we rely on the Digital Twin as the information source, with the main properties: *synchronization with the asset*, *model orchestration*, and *simulation capability* [2]. It models the system and its environment. In our work, we focus on the simulation capability but need the other parts as well. Indeed, it is a challenge in itself to build a good Digital Twin. Kousi et al. [22]therefore investigate the use of digital world modeling in hybrid production systems, Staczek et al. [23] show the important role of DT technology to test the operating environment of an autonomous mobile robot and early detection of design defects. In [24, 25], we contribute guidelines of building system's Digital Twin and its environment and address the challenges of synchronization.

Summing up the state of the art, there is a research gap in designing systems that determine the situational risk at runtime. Therefore, we contribute a novel design approach for building these systems (Section 3). As this comes at the cost of computational complexity, we consider this arising issue regarding the learning models and run time validation in Section 4 and introduce the drivers of complexity as a preliminary step for managing complexity.

3. Situational Risk Assessment

Digital Twin as a digital representation of a physical asset is a new concept to address the challenges of having flexible systems with shorter life cycles by providing the required information. It consists of data and models of the system to simulate, predict and optimize the system processes in the virtual environment [2]. As introduced in the latter section, our analysis is based on the system theoretic process analysis (STPA) [21]. The Digital Twin provides the STPA with the needed information, specifically the control-theoretic models as well as context information and relations to other models. The STPA results in a set of scenarios, where a control action potentially leads to a hazard. By taking the Digital Twin's information model, all inputs involved in processing the respective control actions reveal the basis of unexpected inputs. Note that this analysis is not requiring for a manual search of scenarios where these constellations of unexpected control actions might occur. These unexpected inputs are produced by adversaries. The adversaries are reinforcement learning agents injecting faults in terms of disturbances and environment features. They are therefore designed based on the relevant signals, the potential losses, and optionally loss scenarios. The relevant signals define the action space where the losses and ideally the loss scenarios form the basis of the adversary's reward function. Then, the adversaries are trained in the Digital

Twin's simulation environment. They get a reward if they manage to cause the control actions leading to hazards in the simulation. We suggest sampling over the possible values of the input vectors thus building a look-up table of trained policies with different strengths. The fault injection (i.e. the actions) of the adversaries now manipulates the original signals thus destabilizing the system in simulation. The strongest fault injection must be chosen such that the adversary always wins in terms of destabilizing the system to cause a hazard. However, especially in the interplay of different adversaries, weaker adversaries will also be able to cause hazards. The question of risk is now: how likely is a disturbance strong enough to force the system to fail? In order to determine this question at runtime, the monitored space is considered. For this reason, we instantiate Monitored Space Observation. Fig. 1 visualizes this analysis. It is exemplified in Section 5.

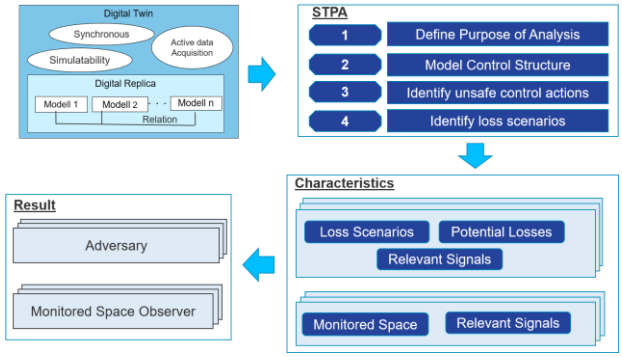

Fig. 1. Design Process for Monitored Space Observer and Adversaries

Note that the Monitored Space Observation does not only consider how much of the coverable area is covered in the respective situation but also which quality the signal has. These two dimensions cause different types of adversaries. In the first case, the adversaries try to hide important information within the uncovered space in the latter case they try to exploit the signal uncertainty in order to manipulate the outcome. Taking the mobile robot as an example, uncovered space might occur when obstacles shadow the sensors. In this case, the adversary simulates other vehicles or human workers in the unseen field crossing the mobile robot's trajectory. In the case of signal uncertainty, e.g. imprecise estimation of the orientation, the adversary manipulates the orientation such that the mobile robot crashes into an obstacle rather than driving around them.

In this way, the STPA points to several agents. We suggest following the divide-and-conquer principle and identifying more but simpler adversaries. The adversaries then all have different action spaces according to the various models, the Digital Twin organizes. However, for each adversary, the goal remains the same: injecting a fault such that the system runs into a loss. Therefore, some patterns occur.

To analyze these patterns, we cluster the adversaries in the MAPE scheme [26] ("Monitore" is replaced with "Sense" to avoid confusion with the monitored space). Moreover, we differentiate between the adversaries exploiting limited observation range and those exploiting limited precision. Fig. 2 depicts the design pattern of the adversaries.

The sense adversaries disturb the preprocessing of the raw sensor signals. One classic example is simultaneous localization and mapping (SLAM). The adversaries in this cluster mainly focus on the precision of the signals. The Digital Twin provides statistical analytics for this purpose. The action spaces range from a light barrier signal, where the adversary just manipulates the signal intensity to complex image manipulation in the sense of adversarial examples [27]. In order to avoid sparse reward issues, the adversaries observe the effects in the next layer as intermediate goals. For example, the reward is provided when fault injection leads to the misdetection of an object. Again it is a service of the Digital Twin that provides the loss matrix of the respective misclassifications. In dead, in the following analysis part typically object detection takes place. It takes the point map of the step sense and extracts objects from it using pattern recognition. The patterns and the transformation models from point cloud to 3d object are retrieved from the Digital Twin. The result is an object map building the environmental understanding of the system. As manipulations in the data leading to misclassifications are already covered in the previous step, the fault injection in the analysis step focuses on the observation range. In our example of the mobile robot, the adversaries try to hide obstacles in the unobserved area in order to provoke collisions. We suggest building object classes like moving obstacles, static obstacles, etc., and representing each group by an adversary. As in the previous step, we recommend using intermediate goals for easing the training of the adversaries. In this case, the time, and injected obstacle that remain undetected serves as criteria. Moreover, as a simplification, the training process can provide reward already if the planned trajectory intersects an introduced object, saving time for simulating the execution of the planned trajectory.

The following planning stage involves the Guidance Protagonist, a stabilizing reinforcement learning agent to plan a cost map which is used by the subsequent execution step as a setpoint for the controller. In our case, the protagonist determines safety areas around the respective object types using the Digital Twin's object models combined with the loss models. The reward function contains a strong negative reward for collisions and a small negative reward for extending the length of the trajectory thus balancing risk vs. efficiency. The adversary counteracts the protagonist by calculating critical trajectories for the moving obstacles forcing collisions. This type of adversary is trained with the previously calculated trajectory of the guidance planning and the object map as an input. This object map contains the attribute of the objects marking them as movable. The adversary then step by step moves the obstacles towards the mobile robot in order to provoke a collision. This process is simulated in the Digital Twin, but not mirrored to the asset. The action space for each of the obstacles depends on their physical properties like max. speed etc. These properties again provide the Digital Twin. In this case, no simplification can be applied since the system's reaction on eventually tracking and estimating the behavior of the moving obstacles has to be taken into account. This aspect is taken into account in the execution step. In this step, a

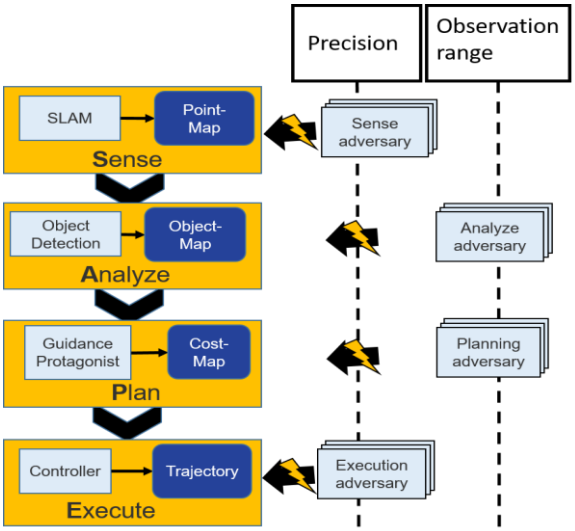

Fig. 2. Design Pattern of the Adversaries

controller generates the mobile robot's trajectory based on the cost map of the planning steps. One option of implementing this controller is model predictive control. As the risks of limited observation range are already covered by the planning adversary, this adversary again belongs to the group of precision adversaries. One example of the execution adversary is the simulation gap adversary. It is to expect that even the Digital Twin takes care of the up-to-dateness of the models, a simulation gap will remain. This simulation gap misleads the controller such that the actions chosen by the control do not really result in the calculated outcome. The respective adversary exploits these imprecisions. In the case of the mobile robot, it manipulates for example the orientation angle such that the robot deviates from the planned course. Naturally, the controller will eventually detect and compensate for the deviation. Depending on the strength of the adversary and the dead time of the controller, a collision may still occur.

As previously discussed, the concrete risk is evaluated within a series of simulated experiments according to the current situation. The adversaries exist in different strengths (i.e. different action spaces). The monitored space observation selects the currently suiting adversary in real-time from the measured parameters: signal precision and observation range. The monitored space observation therefore statistically analyzes the incoming data for its variance and the observed area at the different levels. Table 1 shows an example analysis of the respective fields. Therefore, the presented risk assessment methodology uses the Digital Twin to provide data and models for STPA to ascertain the scenarios leading to hazards and system failure. The confrontation between adversaries and protagonists as destabilizing and stabilizing reinforcement learning agents provides vulnerable and safe areas around the respective object. The introduced Monitored Space Observation is used to select the strength and coverage of fault injection processes. This evaluation scenario brings about a situational risk area reorganization mobile robot. Building all these adversaries provides on the one hand more and more precise modeling of various risk factors. However, this comes at cost of increased complexity. The analysis of this complexity is discussed in the following section.

Table 1. Observed parameters of monitored space observation in the respective processing stages

Processing stage	Observed category	Observed parameter
Sense	Signal variance	Distribution of the signals
Analyze	Observed area of the sensory	Regions in the map covered by sensory
Plan	Observable behavior of obstacles	Predictability of the obstacle's behavior. Unrecognized obstacles are treated completely unpredictable.
Execute	Simulation gap, model inprecisions	Deviation from the simulated outcome to the measured one.

4. Complexity analysis

The aforementioned approach suggests training multiple reinforcement learning agents for fault injection and then assessing the risk. While executing the policies at runtime is computationally lightweight, training the policies requires certain computational power. Therefore, we propose how to improve the quality of the fault injection process against the required effort, i.e. apply complexity management. Fig. 3 visualizes the complexity drivers.

From intuition, it is clear that safety increases with finding critical situations more efficiently which correlates with the power of the adversaries. The adversaries' power increases with the granularity of the action space, the number of agents trained together, and the number of training epochs. Generalization increases with the number of different scenarios involved in the training process, e.g. different maps containing different types of objects, different trajectories, etc. Moreover, the engineering effort decreases, less preprocessing is required for the input data, and trending to high dimensional input spaces. However, all these parameters increase the complexity and consequently the training effort. In order to limit this effort, we introduce a complexity budget. The complexity budget defines the amount of acceptable complexity up to which the system remains manageable. This complexity budget is determined from the use cases, the engineering, and the design-time risk assessment. We have to define first, how much we are

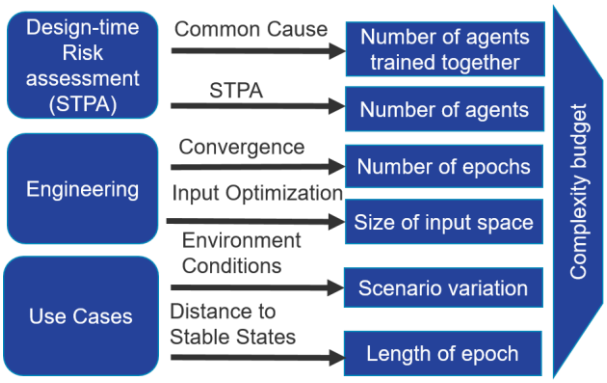

Fig. 3. Complexity Budget. Analysis from Design Phase, namely use case, engineering and risk assessment to estimate the complexity

willing to pay in order to further improve safety. This budget is limited by practical as well as economical arguments like the time spent for training, the required engineering effort, etc. In our case, we use the training complexity of reinforcement learning as a reference and measure it in training epochs. We count the epochs with respect to the already optimized policy as depicted in Tabel 3. We generate an exchange process for example to engineering effort, as further optimization in the engineering process can optimize the needed number of epochs. From the use cases, stable states can be derived, where the risk is known to be neglectable, e.g., when the mobile robot is docked to a working station in a charging- or entering a waiting position. These stable states determine the length of epochs since the runtime risk assessment simulates the activities between two stable states. Moreover, the agents are trained for the environmental conditions, the use case focuses on determining the variation of the expected situation and therefore, the number of test setups. From the engineering, the input optimization determines whether the agents get a map as bitmap or object position, etc. thus influencing complexity. Moreover, by selecting the architecture of the agents, we indirectly define convergence speed and the number of epochs we need for training. Finally, design time risk assessment provides as output the number of agents, as well as the groups of agents as common cause analysis. Table 3 provides a reference value of how much complexity is put into the complexity budget in the following case study.

5. Case Study

For our case study, we consider Robotino 3 Premium by Festo, a mobile robot. Its goal is to deliver intermediate products to the workstations of a matrix production system. For this setup, we perform a simplified STPA in order to define our adversaries. The result of the STPA is depicted in Table 2. Here, we consider the two control signals move and rotate, which can lead to two different kinds of losses: blocking the process and colliding. The collision is the worse loss and the only safety-critical one. For this reason, we give priority to collisions.

Table 2. Results of STPA. STPA identifies different loss scenarios

Control action	Wrongly provided	Wrongly not provided	Wrong timing
Move	Robot crashes into an obstacle	Production process blocked	Crash or process blocked
Rotate	Deviation from plan leads into crash	Deviation from plan leads into crash	Deviation from plan leads into crash

The next step is to derive the appropriate adversaries for the four layers: Sense, Analyze, Process, and Execute. As in STPA, losses are always allocated in the Execute part, we start with this part. We identify that a deviation in the rotation might lead to leafing the planned route and therefore crash. For this reason, we build an adversary manipulating the robot's orientation. Through backpropagate using the digital twin, we find a prediction of moving obstacles influencing the movement from the processing layer. Based on that, we insert an adversary manipulating the trajectories of moving obstacles. Another layer above, we learn from the Digital Twin, that the object model is essential for movement prediction. However, the mobile robot does not know all objects from the beginning. It has to discover them during runtime. Therefore, if obstacles are not in the field of view of the robot, it will not recognize them. The Analyze-Adversary exploits this property and inserts obstacles at the unobserved areas. Intentionally, the mobile robot detects the obstacle too late and crashes. The Sense-Adversary's manipulation points in the same direction, but even one layer above. It disturbs the input signal, e.g. the camera signal. The camera is used to identify the respective objects. Therefore, by adding adversarial noise, a misclassification takes place, which eventually leads to a crash with the respective object. Table 3 summarizes the adversaries, their parameters, and a reference value of the complexity.

Table 3. Example Adversaries for the different layers with their action spaces and complexity after optimization

Layer	Adversary's Goal	Input space	Action space	Complexity
Sense	Image manipulation	Image size	Noise matrix	~50,000 eq. epoch[27]
Analyze	Position obstacles in unobserved area	Field of View, List of Objects	Object type, position	~50,000 eq. epoch
Process	Manipulate trajectory of moving obstacles	Field of View, List of Objects, Ego perception	Movement Direction	~16,000 eq. epoch
Execute	Manipulate the orientation of the robot	Grayscale map (bitmap)	Deviation in Angles	~ 60,000 eq. epoch

We define for our complexity budget a limit of 200,000 equivalent epochs (eq. epoch), which corresponds to ca. 36 h of training on our hardware. The equivalent epochs directly correlate with the runtime behavior, since one epoch represents one simulation run. A budget of 200,000 eq. epoch therefore directly corresponds with a computational effort of 648 ms, which is acceptable for our robot's high-level planning. Then we naively set up the adversaries ending up a factor between 2 and 10 above our complexity budget. By reducing the granularity of the execute adversary to fife angles and optimizing the training environment, directly implementing it in OpenGL and c/c++, we managed to reach the results provided in the table and thus kept our budget.

The above case study successfully exemplifies the application of our design process. Moreover, reference values for the respective adversaries are provided. Indeed, the adversaries have to be designed individually for the respective system. However, this design process provides guidance in setting up a situation-based runtime risk assessment for mobile robots. In principle, it is transferrable to other domains as well.

6. Conclusion and Future Work

This paper presents a methodology enabling to find out the situational risk for mobile robots operating in dynamically changing and heterogeneously structured environments. The presented risk assessment methodology uses Digital Twin to provide data and models for System Theoretic Process Analysis to ascertain the scenarios leading to system failure. Moreover, the Digital Twin's model management capability, as well as its synchronization property, are used at runtime in order to simulate the system's current situation accurately. Multi-Agent Adversarial Reinforcement Learning forces self-play between two competing kinds of agents, namely Adversary and Protagonist. Like prosecutors and defense attorneys in court, both sides present evidence that argues for or against the safety of the system: the adversaries by fault injection and the protagonists by coping with it. Circumstances determine the strength of the parties and thus the outcome of the confrontation, i.e. the situational risk. The Monitored Space Observation provides the information of the circumstances in terms of statistical signal property. The methodology is validated in a case study using Robotino 3 Premium by Festo.
By focusing on the situational risk instead of the overall risk, this approach contributes to solve the scenario explosion. However, shifting the risk assessment to runtime causes additional software complexity. The complexity issue of required learning for this methodology is considered and investigated the factors that affect the required effort to simulate. However, more studies in situative risk assessment and complexity management in intelligent software-defined systems are required. For this reason, in future work, we are going to further study the complexity of intelligent software-defined systems and situational risk assessment at runtime.

References

[1] M. Müller, T. Müller, B. Ashtari Talkhestani, P. Marks, N. Jazdi, and M. Weyrich, "Industrial autonomous systems: a survey on definitions, characteristics and abilities," *at - Automatisierungstechnik*, vol. 69, no. 1, pp. 3–13, 2021.

[2] B. Ashtari Talkhestani *et al.*, "An architecture of an Intelligent Digital Twin in a Cyber-Physical Production System," *at - Automatisierungstechnik*, vol. 67, no. 9, pp. 762–782, 2019.

[3] A. Bajcsy, S. Bansal, E. Bronstein, V. Tolani, and C. J. Tomlin, "An Efficient Reachability-Based Framework for Provably Safe Autonomous Navigation in Unknown Environments," in *2019 IEEE 58th Conference on Decision and Control (CDC)*, 2019, pp. 1758–1765.

[4] D. Silver *et al.*, "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[5] B. Baker *et al.*, "Emergent Tool Use From Multi-Agent Autocurricula," Sep. 2019. [Online]. Available: https://arxiv.org/pdf/1909.07528

[6] L. Pinto, J. Davidson, R. Sukthankar, and A. Gupta, "Robust Adversarial Reinforcement Learning," *International Conference on Machine Learning*, pp. 2817–2826, 2017. [Online]. Available: http://proceedings.mlr.press/v70/pinto17a.html

[7] X. Wang, S. Nair, and M. Althoff, "Falsification-Based Robust Adversarial Reinforcement Learning," in *2020 19th IEEE International Conference 2020*, pp. 205–212.

[8] K. Zhang, T. A.O. SUN, Y. Tao, S. Genc, S. Mallya, and T. Basar, "Robust Multi-Agent Reinforcement Learning with Model Uncertainty," *Advances in Neural Information Processing Systems*, vol. 33, pp. 10571–10583, 2020.

[9] X. Pan, D. Seita, Y. Gao, and J. Canny, "Risk Averse Robust Adversarial Reinforcement Learning," in *2019 International Conference on Robotics*, pp. 8522–8528.

[10] N. Kalra and S. M. Paddock, "Driving to safety: How many miles of driving would it take to demonstrate autonomous vehicle reliability?," *Transportation Research Part A: Policy and Practice*, vol. 94, pp. 182–193, 2016.

[11] *Szenarien für entwicklung, absicherung und test von automatisierten fahrzeugen*, 2017. [Online]. Available: https://www.uni-das.de/images/pdf/veroeffentlichungen/2017/13.pdf

[12] A. Hata, R. Inam, K. Raizer, S. Wang, and E. Cao, "AI-based Safety Analysis for Collaborative Mobile Robots," in *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2019, pp. 1722–1729.

[13] P. Helle, W. Schamai, and C. Strobel, "Testing of Autonomous Systems - Challenges and Current State-of-the-Art," *INCOSE International Symposium*, vol. 26, no. 1, pp. 571–584, 2016.

[14] D. Harel, A. Marron, and J. Sifakis, "Autonomics: In search of a foundation for next-generation autonomous systems," *PNAS*, vol. 117, no. 30, pp. 17491–17498, 2020.

[15] P. Feth, D. Schneider, and R. Adler, "A Conceptual Safety Supervisor Definition and Evaluation Framework for Autonomous Systems," in *Computer Safety, Reliability, and Security: 36th International Conference, SAFECOMP 2017, Trento, Italy, September 13-15, 2017, Proceedings / Stefano Tonetta, Erwin Schoitsch, Friedemann Bitsch*, Cham, 2017, pp. 135–148.

[16] J. Rushby, "Runtime Certification," in *Runtime Verification*, Berlin, Heidelberg, 2008, pp. 21–35.

[17] G. Hagele and A. Sarkheyli-Hagele, Eds., *Situational risk assessment within safety-driven behavior management in the context of UAS*, 2020.

[18] J. C. Pereira and G. B. Alves Lima, "Probabilistic risk analysis in manufacturing situational operation: application of modelling techniques and causal structure to improve safety performance," *Int. J. Prod. Manag. Eng.*, vol. 3, no. 1, p. 33, 2015.

[19] M. Müller, N. Jazdi, and M. Weyrich, "An Approach for Context-Sensitive Situational Risk Evaluation of Autonomous Systems," in *26th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA) 2020 (accepted)*.

[20] Y. Luo, Y. Yu, Z. Jin, and H. Zhao, "Environment-Centric Safety Requirements for Autonomous Unmanned Systems," in *2019 IEEE 27th International Requirements Engineering Conference (RE)*, 2019, pp. 410–415.

[21] N. Leveson, *Engineering a safer world: Systems thinking applied to safety*. Cambridge, Massachusetts: The MIT Press, 2017. [Online]. Available: https://library.oapen.org/handle/20.500.12657/26043

[22] N. Kousi, C. Gkournelos, S. Aivaliotis, C. Giannoulis, G. Michalos, and S. Makris, "Digital twin for adaptation of robots' behavior in flexible robotic assembly lines," *Procedia Manufacturing*, vol. 28, pp. 121–126, 2019.

[23] P. Stączek, J. Pizoń, W. Danilczuk, and A. Gola, "A Digital Twin Approach for the Improvement of an Autonomous Mobile Robots (AMR's) Operating Environment-A Case Study," *Sensors*, vol. 21, no. 23, p. 7830, 2021.

[24] M. S. Müller, N. Jazdi, and M. Weyrich, "Self-improving Models for the Intelligent Digital Twin: Towards Closing the Reality-to-Simulation Gap," in *14 th IFAC Worksop on Intelligent Manufacturing Systems*, Tel Aviv, 2022 (accepted).

[25] D. Braun, W. Schloegl, and M. Weyrich, "Automated data-driven creation of the Digital Twin of a brownfield plant," in *2021 26th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2021, pp. 1–7.

[26] J. O. Kephart and D. M. Chess, "The vision of autonomic computing," *Computer*, vol. 36, no. 1, pp. 41–50, 2003.

[27] C. Xie, J. Wang, Z. Zhang, Y. Zhou, L. Xie, and A. Yuille, "Adversarial Examples for Semantic Segmentation and Object Detection," in *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017.