---

**Proceedings Paper:**
VAZQUEZ FLORES, Gricel, Imrie, Calum Corrie, Shahbeigi Roudposhti, Sepeedeh et al. (Accepted: 2026) Formally Guaranteed Control Adaptation for ODD-Resilient Autonomous Systems. In: 21st International Conference on Software Engineering for Adaptive and Self-Managing Systems (SEAMS 2026). (In Press)

---

# Formally Guaranteed Control Adaptation for ODD-Resilient Autonomous Systems

Gricel Vázquez*, Calum Imrie*, Sepeedeh Shahbeigi*, Nawshin Mannan Proma*, Tian Gan*, Victoria J Hodge*, John Molloy*, Simos Gerasimou*†
*Department of Computer Science, University of York
UK
†Department of Electrical Engineering and Computer Science, Cyprus University of Technology
Cyprus
gricel.vazquez@york.ac.uk,calum.imrie@york.ac.uk,sepeedeh.shahbeigi@york.ac.uk,
nawshinmannan.proma@york.ac.uk,tian.gan@york.ac.uk,victoria.hodge@york.ac.uk,
john.molloy@york.ac.uk,simos.gerasimou@york.ac.uk

## Abstract

Ensuring reliable performance in situations outside the Operational Design Domain (ODD) remains a primary challenge in devising resilient autonomous systems. We explore this challenge by introducing an approach for adapting probabilistic system models to handle out-of-ODD scenarios while, in parallel, providing quantitative guarantees. Our approach dynamically extends the coverage of existing system situation capabilities, supporting the verification and adaptation of the system's behaviour under unanticipated situations. Preliminary results demonstrate that our approach effectively increases system reliability by adapting its behaviour and providing formal guarantees even under unforeseen out-of-ODD situations.

## Keywords

Situation coverage, out-of-ODD, probabilistic model checking, runtime verification

**ACM Reference Format:**
Gricel Vázquez*, Calum Imrie*, Sepeedeh Shahbeigi*, Nawshin Mannan Proma*, Tian Gan*, Victoria J Hodge*, John Molloy*, Simos Gerasimou*†. 2026. Formally Guaranteed Control Adaptation for ODD-Resilient Autonomous Systems. In *21st International Conference on Software Engineering for Adaptive and Self-Managing Systems (SEAMS '26), April 13–14, 2026, Rio de Janeiro, Brazil*. ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3788550.3794869

## 1 Introduction

The increasing integration of autonomous systems in safety-critical domains, such as healthcare [28] and maritime transportation [21], demands rigorous safety guarantees. A primary challenge in developing resilient autonomous systems is ensuring reliable performance in situations that extend beyond their predefined Operational Design Domain (ODD). At design time, the ODD defines the specific conditions under which a system is intended to function safely, often based on established standards [23]. However, the complexity of real-world environments, evolution of the working environment and domain, and emergent system behaviours mean that systems could encounter conditions outside their ODD [16].

Such out-of-ODD conditions will cause the system behaviour to become inherently uncertain, due to the ODD at design time no longer fully applicable. While improving the underlying system could help mitigate risks discovered at runtime, such updates are often not feasible at runtime, as they may require unaffordable redesign or retraining. Moreover, the system might not be equipped to detect such out-of-ODD conditions. Consequently, it becomes necessary to adapt the system's controller to avoid critical situations where safety requirements are violated at runtime. This ensures that the system can continue to operate safely even under previously unseen, and potentially unsafe conditions.

Self-adaptive Systems (SAS) identify changes and devise strategies to continue successful operation. Runtime approaches (e.g., online testing and runtime verification) are valuable for SAS as they focus on observing the system's behaviour as it executes. While these methods are effective at detecting deviations from expected behaviour, they lack a framework for the mitigation of out-of-ODD encounters, and for dynamically adapting the system's underlying models to safely handle new situations while complying with strict safety requirements. To address these limitations, this paper introduces **SAVE** (**S**ituation-**A**ware **V**erification and control synth**E**sis), an ODD-driven, situation-centric modelling and adaptation approach, specifically designed to handle out-of-ODD conditions. Our approach leverages simulation to build probabilistic models of system behaviour, and probabilistic model checking to provide quantitative guarantees about the system's performance and safety. To ground our approach, we use an autonomous maritime system as a motivating example.

The main contributions of our paper are as follows:

- A novel situation-centric approach, grounded in SAS principles leveraging system situations extracted from the ODD to generate probabilistic situation models, and uses them to generate controllers at deployment time, providing formal quantitative guarantees on safety and performance.


*SEAMS '26, Rio de Janeiro, Brazil*

ACM ISBN 979-8-4007-2445-9/2026/04
https://doi.org/10.1145/3788550.3794869

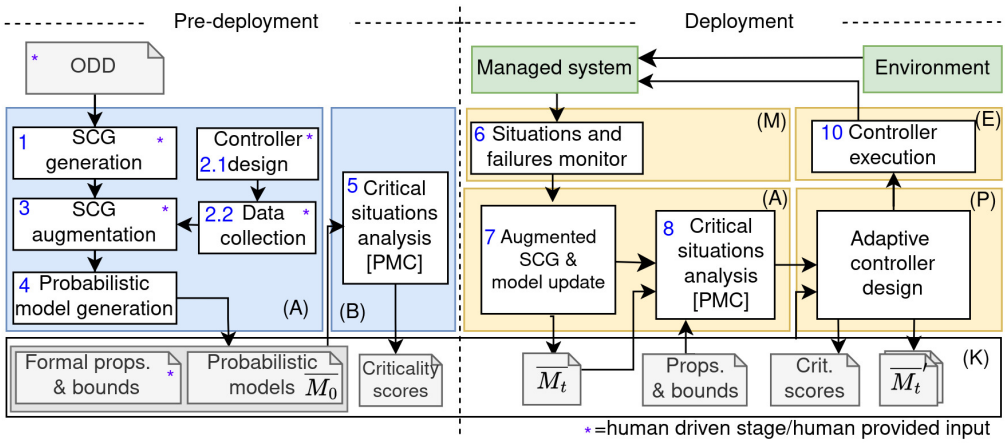


**Figure 1: SAVE approach overview: pre-deployment (A,B) and deployment (M,A,P,E) phases with shared knowledge.**

• A method for encoding probabilistic transitions between encountered situations and a criticality score metric for the assessment of such situations, incorporating data-driven updates to transition probability values.
• A runtime verification-driven adaptation loop that detects safety system violations at deployment time and synthesises updated controllers that remain safe (even under ODD drift).

## 2 Background

**Situation Coverage Grid (SCG)**. An SCG is a composite criterion used to test autonomous systems by systematically exploring a wide range of scenarios [4, 26]. This grid combines various situational elements (e.g., road junction types and entities like cars, and pedestrians) [29].
**Discrete-time Markov chain (DTMC).** A discrete-time Markov chain (DTMC) models a system that moves between states in discrete steps according to fixed probabilities [19]. Probabilistic Computation Tree Logic (PCTL) is a formal language for the specification of requirements [19, 20]. Probabilistic model checking (**PMC**) is the process of automatically verifying whether the DTMC model satisfies such requirements [6]. This automation is allowed by tools such as PRISM [20].

## 3 The SAVE approach

An overview of our SAVE approach is depicted in Figure 1 divided into pre-deployment and deployment phases.[1]

### 3.1 Pre-deployment

**SCG Augmentation (A).** The ODD provides a structured representation of the operating context. From this structured representation, a discrete set of situations that the system is expected to encounter is generated. In this paper, we define a situation $\rho$ as a $n$-tuple of valid subsets of (attribute:value) of the ODD: $\rho = (v_1, v_2, \ldots, v_n)$, where $v_i \in$ ODD for $i = 1, \ldots, n$.

[1] In this section, the blue formatted text corresponds to the numbered SAVE stages in Figure 1.

SAVE generates an SCG (1) from such situations. As a motivating example in the maritime domain, an ODD consists of three attributes: density of detected vessels of type A, type B, and the shortest time to collision (TTC) with any neighbouring vessel. The first two can take values from $(none, low, high)$, while the latter $(short, long)$. A concrete situation is then defined as $\rho = (none, low, short)$.

To capture the system's dynamic behaviour as it transitions between different situations, SAVE extends the SCG by incorporating transition probabilities derived from empirical test data (2.2). Stakeholders and domain experts can design and test various controller designs (2.1). The objective during this process is to obtain system controllers that meet the pre-defined system requirements, while minimising the likelihood of entering critical (prone to requirement violations) situations during deployment. This data-driven stage results in an augmented SCG (3).

*Definition 3.1 (Augmented SCG).* Let $\overline{\xi} = \{\xi_1, \xi_2, ...\}$ be a set of failures. Given an SCG, $\overline{\rho}$, and a list of failures, $\overline{\xi}$, an augmented SCG is defined as a tuple $(S, \delta)$; where $S = \overline{\rho} \cup \overline{\xi}$, $\overline{\rho} \cap \overline{\xi} = \varnothing$ is a set of states; and the transition function $\delta : \overline{\rho} \rightarrow Dist(S)$ defines probabilistic transitions across all situations and failures.

To analyse the system properties in (4), SAVE generates a set of DTMCs $\overline{\mathcal{M}_0}$, each with a unique initial state representing one situation $\rho$, with transition probabilities derived from the augmented SCG. Each DTMC model has a set of states, where each state represents a situation or a failure, and encodes transitions between normal and failure states.

*Definition 3.2 (Probabilistic Model of $\rho_i$).* Given an augmented SCG $(S, \delta)$ and a situation $\rho_i$, we define a DTMC $\overline{\mathcal{M}[i]} = (S', \bar{s}, \delta', AP, L)$; where $S'$ denotes the set of states, consisting of the state variables from $S$ in the augmented SCG; $\bar{s} \in S'$ is initial state representing the system starting at situation $\rho_i$; $\delta'$ is the transition function $\delta' : S' \rightarrow Dist(S')$, where transitions, representing a change in situation or into a

failure state, are obtained from the augmented SCG (failure states are sink states); $AP$ is a set of atomic propositions defined for failure states; and $L$ the state labelling function (see Section 2).

**Critical Situations Analysis (B).** After eliciting the situation-aware probabilistic models, SAVE applies probabilistic model checking (PMC) to automatically identify and rank the most critical situations. For each situation $\rho_i$ with DTMC $\overline{\mathcal{M}}[i]$, the system verifies the model against the elicited safety properties $\overline{\Phi}[k]$. The degree of property violation determines the criticality score (5), where a score of 0 indicates requirement compliance, and higher values indicate increasing deviation from the property bound. As an example, consider the requirement "the minimum probability of success is 0.96." Here, 0.96 serves as the bound. A situation's DTMC model that achieves only 0.85 for this property has a higher criticality score (0.11) than a model achieving 0.94 (0.02), although both models violate such property. At this stage, normalisation techniques can be applied to ensure fair comparison when properties use different scales (falling outside the scope of this paper).

During pre-deployment, the resulting criticality scores enable stakeholders to prioritise mitigation efforts by identifying the most critical situations and leveraging detected requirement violations to iteratively refine the controllers and overall system design until all tested controllers are safe (by constraining or modifying the system's behaviour, degrading soft requirements, etc.). Once this phase is successfully completed, the system progresses to the deployment stage.

## 3.2 Deployment

At deployment, the system is continuously adapted to ensure safe operation. SAVE is aligned with the MAPE-K loop as shown in Figure 1. We define each stage as follows.

**Monitoring of the Managed System (M).** The monitoring (6) stage obtains the following data at time $t \in \mathbb{R}^{>0}$: 1) **changes in the current system's situation**, which monitors the current encountered situation $\rho_t$; and 2) **system failures**, the known failure conditions pre-identified as $\overline{\xi}$. For our maritime domain example, the vessel must be capable of detecting other vessels, classifying them as type A or B based on their characteristics, and estimating the distance to the nearest vessel in order to determine the current situation.

**Model Update and Analysis of Critical Situations (A).** SAVE uses the monitored data to construct a new augmented SCG at time $t$ updating the transition probabilities defined by $\delta$ using frequentist [3, 9] or Bayesian-based [35, 36] approaches as the system evolves from one situation into another, or into a failure state (7). A new set of models $\overline{\mathcal{M}_t}$ is then constructed as in the pre-deployment. SAVE then analyses (8) these models and, where necessary, updates them to derive a safe system controller as follows:

- SAVE obtains the model where the current situation $\rho_t$ is an initial state. This model is analysed using PMC to assigned a critically score as in the pre-deployment stage.
- If no violations were detected (criticality score$\leq 0$), the system proceeds as normal.
- Else, SAVE obtains the criticality score of each model in $\overline{\mathcal{M}_t}$. The situation $\rho$ with the model with the worst criticality score is then obtained. Finally, all outgoing transitions in $\mathcal{M}_{\rho_t}$ from the state representing $\rho$ are removed, leaving only a self-loop with probability 1. This also means that such outgoing transition probability in the augmented SCG were set to zero. SAVE systematically removes the most critical situations until no violations remain (or until a predefined maximum number of situation states become sink states, indicating failure to synthesise a safe controller). Thus, SAVE prevents the system from continuing once an out-of-ODD (potentially unsafe) situation is detected.

**Controller Synthesis (P).** As the controllers are modelled to be safe from the pre-deployment stage, at runtime, such controllers might become unsafe, for example, when out-of-ODD situations appear. When the analysis stage (A) detects that the current system configuration $\overline{\mathcal{M}_t}$ violates one or more requirements $\overline{\Phi}$, SAVE triggers an adaptive controller design (9) process. The aim is to synthesise a verifiably safe controller by iteratively excluding the most critical situations from the operational context. In each iteration, the situation's model with the highest criticality score, indicating the largest deviation from a safety requirement, is removed.

After a critical situation and its model are removed, stage (9) modifies the remaining probabilistic models by modelling that situation state as a "sink state", effectively building a barrier that prevents the system from continuing from those discovered, unsafe conditions. Since removing one situation and altering the models can change the probabilistic outcomes of the entire system, the criticality scores for all remaining situations are then re-calculated. This cycle of identifying the most critical situation, blocking access to it, and re-evaluating the system's safety continues until all situations comply with the set of requirements (i.e., criticality score of zero), or a maximum number of iterations is reached. The final output is a revised set of safe situations and models $(\overline{\rho}'_t, \overline{\mathcal{M}}'_t)$, which constitute the newly synthesised, safer controller.

When a system is in a situation where multiple candidates controllers exists from the pre-deployment stage, the selection is based on two criteria: the controller must result in no property violations, and it must minimise the probability of the system reaching an out-of-ODD situation. If more than one comply with such criteria, one is chosen at random.

**Controller Execution (E).** Finally, the selected controller is executed (10) by instrumenting the managed system. The entire execution process continues, ensuring that the system's adaptation remains responsive to changes.

# 4 Preliminary Evaluation

**Maritime domain case study.** We perform an initial evaluation of SAVE using a simplified maritime case study involving a Marine Autonomous Surface Ship (MASS) navigating among two types of crewed vessels (A and B). Each vessel type is characterised by its size, velocity, and time to collision (TTC)

**Table 1: SAVE adaptation results after a violation is detected. Without adaptation, the baseline fails in all cases.**

| ID | Property violated | Worst criticality score | SAVE success (no violations) | Critical situations avoided |
|---|---|---|---|---|
| 1 | [$\Phi_2$] | 0.03890 | True | [s2, s3] |
| 2 | [$\Phi_2$, $\Phi_1$] | 0.03482 | True | [s2, s3, s5] |
| 3 | [$\Phi_2$, $\Phi_1$] | 0.03556 | True | [s3, s4] |
| 4 | [$\Phi_2$, $\Phi_1$] | 0.04483 | True | [s5, s1, s3, s2] |
| 5 | [$\Phi_2$, $\Phi_1$] | 0.04905 | False | - |
| 6 | [$\Phi_2$, $\Phi_1$] | 0.04602 | True | [s5, s2, s1, s4] |
| 7 | [$\Phi_1$] | 0.00954 | True | [s1, s4] |
| 8 | [$\Phi_2$, $\Phi_1$] | 0.04827 | True | [s2, s4, s5, s1] |
| 9 | [$\Phi_2$] | 0.04892 | True | [s5] |
| 10 | [$\Phi_2$] | 0.04963 | True | [s5] |
| 11 | [$\Phi_2$, $\Phi_1$] | 0.04967 | False | - |
| 12 | [$\Phi_1$] | 0.00983 | True | [s3] |
| 13 | [$\Phi_2$, $\Phi_1$] | 0.04993 | False | - |
| 14 | [$\Phi_2$] | 0.04999 | False | - |
| 15 | [$\Phi_1$] | 0.00992 | True | [s5] |
| 16 | [$\Phi_2$] | 0.04994 | True | [s1] |
| 17 | [$\Phi_2$] | 0.04999 | True | [s3, s4] |
| 18 | [$\Phi_1$] | 0.00998 | True | [s3] |
| 19 | [$\Phi_2$, $\Phi_1$] | 0.04999 | False | - |
| 20 | [$\Phi_2$, $\Phi_1$] | 0.04999 | False | - |

(short or long). The MASS, controlled by an adaptive AI-based controller, must avoid collisions and maintain adequate separation from other vessels. The ODD is deliberately simplified and discretised to keep the scenario tractable while preserving the essential dynamics of encounter situations such as head-on, crossing, and overtaking.

There are two interlinked monitored failures: *f1* corresponding to inadequate time to react to avoid collision when the TTC is too short; *f2* signifying a a near catastrophic collision when the distance between vessels in near to zero units. In extreme scenarios such as when a vessel is travelling at a high speed, both might be detected at the same time. Finally, the following system properties are considered:

- ($\Phi_1$) The probability of failure $f_1$ occurring within the next 50 situations must be less than $0.99$: $\mathrm{P}_{=?}\,[\,F^{\leq 50}\, f_1\,] < 0.99$.
- ($\Phi_2$) The probability of failure $f_2$ occurring within the next 50 situations must be less than $0.95$: $\mathrm{P}_{=?}\,[\,F^{\leq 50}\, f_2\,] < 0.95$.

**Research questions (RQs)**. We evaluate SAVE on 3 RQs.

**RQ1** [Effectiveness]. How effective is SAVE in reducing requirement violations caused by out-of-ODD situations?

**RQ2** [Adaptation]. How effective is SAVE's adaptation in synthesising violation-free controllers compared to a baseline?

**RQ3** [Scalability]. Given that the most computationally expensive part of SAVE is obtaining the criticality score via PMC, how computationally effective is SAVE in synthesising new controllers as the number of situations increase?

**Results and discussion**. We compare SAVE with a baseline system in which requirement violations may occur (assuming a fixed controller) to assess its effectiveness in reducing both requirement violations and collisions (**RQ1**). The number of situations to avoid is limited to four. For each run, vessel positions and speeds are randomly initialised to generate diverse

**Table 2: SAVE adaptation example.**

| Event | Baseline Outcome | SAVE Response |
|---|---|---|
| Vessel deceleration (out-of-ODD) | Collision expected within the next 50 timesteps ($f2$), detected at time $t_1$. Collision happens in situation $i$ at time $t_2$. | Controller adapted at $t_1$. Critical situation $i$ and collision avoid. |
| Requirement $\Phi_1$ (safe TTC) | Violated at time $t_2$. | Satisfied post-adaptation at time $t_3$ |
| Requirement $\Phi_2$ (near collision) | Violated at time $t_4$. | Satisfied post-adaptation at time $t_5$. |

situations and transition probabilities in the underlying probabilistic augmented SCG. Table 1 reports 20 variants where out-of-ODD perturbations (e.g., increased vessel velocity) lead to violations of one or both properties. In these experiments, we introduce randomly generated disturbances to the transition probability matrix to model such uncertainty. Code and complete experimental results are available in our **GitHub** [1].

We use PRISM [20] to obtain the criticality scores. The results show that SAVE was able to avoid requirement violations in 14 of the 20 variants (column 4) by proactively avoiding high-risk situations (column 5). This improvement arises from SAVE's ability to identify and adapt the system's controller to critical situations. In the maritime case study, this corresponds to performing a crash stop COLREGs, i.e., a collision avoidance manoeuvre when critical situations are detected.[2] All variants where SAVE failed to maintain a safe MASS controller involved violations of property $\Phi_2$, with their criticality score exceeding 0.04 (column 3). These insights from SAVE can support stakeholders in the re-design and refinement of autonomous control strategies.

For **RQ2**, Table 2 illustrates SAVE's controller adaption strategy during a representative out-of-ODD event that causes a system violation, as well as individual system violations caused by the model updates as new data are available after deployment. In this scenario, the baseline system leads to a collision within the next 50 timesteps ($\Phi_2$), detected at time $t_1$ and occurring at time $t_2$. In contrast, SAVE identifies the event at $t_1$, computes the associated criticality score, and adapts the controller configuration to exclude the high-risk situation. This intervention prevents the collision entirely, demonstrating SAVE's capacity to react to unforeseen dynamic changes in real time.

Continuing, SAVE restores compliance with both safety requirements. Requirement $\Phi_1$ (safe time-to-collision) and $\Phi_2$ (near-collision avoidance), which are violated in the baseline case at times $t_2$ and $t_4$, respectively, are both satisfied post-adaptation at times $t_3$ and $t_5$. These results confirm that SAVE not only mitigates imminent risks but also ensures sustained adherence to safety requirements under dynamic and uncertain conditions.

Finally, as the most expensive part of SAVE is the generation of criticality scores using PMC, Figure 2 shows the

[2]The *International Regulations for Preventing Collisions at Sea (COLREGs)*, published by the International Maritime Organization (IMO), provide guidance on how vessels should operate, including the rules governing typical encounters, such as head-on (rule 14); crossing (rule 15); and overtaking (rule 13).

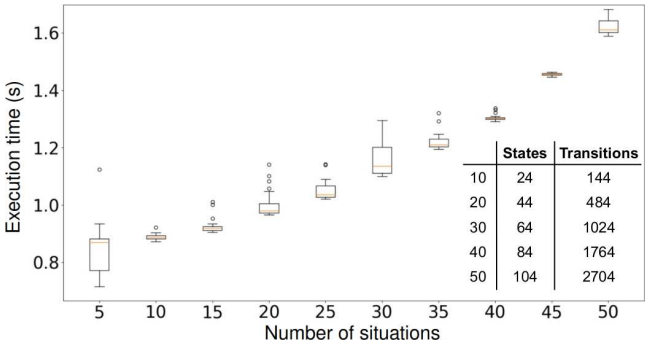


**Figure 2: SAVE execution times to get criticality scores.**

scalability results (**RQ3**) by measuring the execution times for different numbers of situations. The time-scale in seconds show preliminary insights into the feasibility of SAVE for running verification at runtime for critical systems such as in the maritime domain.

The results include the number of states and transitions as the number of situations increase. As expected, the state size increases linearly with the number of situations, and the number of transitions increases exponentially. However, every state in these models have transitions to every state with probability greater than 0. This can therefore be considered the worst case with regards to scaling, with some applications not having this aspect, resulting in far fewer additional transitions with increasing number of situations. Overall, these preliminary results indicate that SAVE has the potential to effectively mitigate safety risks in dynamic environments, outperforming static controllers and demonstrating the feasibility of situation-aware, verifiably safe adaptation.

## 5 Related Work

Adapting autonomous systems (AS) to safely handle unknown situations at runtime "*is the ultimate challenge for self-adaptive systems*" [12]. Recent studies [5, 17, 27, 28, 31] present self-adaptive mechanisms enabling AS to adjust their planned paths and system controllers under uncertainty. However, these approaches do not verify the autonomous systems (ASs) correctness at runtime.

Runtime quantitative verification using PMC has been extensively studied as a means to provide formal guarantees for adaptive and self-adaptive systems (e.g., [7, 10, 11, 22]). Existing approaches typically assume a given system model at runtime—such as Markov decision processes or stochastic games—and focus on synthesising adapted controllers. In contrast, SAVE adopts an ODD-grounded, situation-centric modelling approach in which probabilistic models are derived directly from semantically meaningful situations extracted from the system's ODD. We also establishes explicit traceability between pre-deployment ODD and specifications, testing data, and runtime verification, an aspect not addressed by existing frameworks.

Ideally, ASs learn and adapt over time, detecting and managing uncertainty [32, 34], while being verified throughout their lifecycle to ensure acceptable safety [15, 28]. A feedback loop is proposed in [2] to integrate runtime verification insights and iteratively refine mission parameters, system architectures, and safety analyses. Out-of-distribution (OOD) detection identifies uncertainty in ASs which can lead to unpredictability [16, 33]. Simulation-based testing and formal methods can then be used to verify ASs [30].

In the maritime domain, ODDIT [18] uses digital simulation with ML models to assess if AS states are OOD, while [13] propose an uncertainty-aware OOD detection method combining global and local trajectory models. Neither approach performs verification, and both rely on situation coverage (SitCov) testing [24, 26], which grids the operational space to track tested conditions and quantify exposure to expected and novel situations.

High-level decision-making and control of ASs often use finite-transition systems, suitable for verification via model checking [14, 30]. SitCov is used at design time for verification of situation models in [25]. The proposed SAVE approach builds on our previous work [25] by bridging the gap between formally verifying system compliance with safety properties and identifying violations of these requirements arising from deviations in the ODD relative to the expected values observed during pre-deployment testing. Wider adoption of such techniques depends on the development of comprehensive software frameworks, such as SAVE.

## 6 Conclusions and Further Work

The paper introduces SAVE, a novel approach for adapting probabilistic models for autonomous systems to handle out-of-ODD situations that can impact system requirements, while providing quantitative guarantees. SAVE uses situation coverage and PMC to ensure that autonomous systems can comply with safety and reliability requirements, even when encountering novel or edge-case situations. Preliminary results show the feasibility of our approach to synthesise violation-free controllers at runtime.

Future work will incorporate our end-to-end simulated maritime example provided by our industrial partner. We aim to implement comprehensive safety mechanisms that address potential failures, such as adaptive control and crash stop manoeuvre, in critical situations. Additionally, we will extend our evaluation on the adaptation of different control strategies; and explore techniques such as [8] for the verification of large-scale systems. Finally, we will evaluate our approach across other cyber-physical system domains to provide deeper insights into the adaptability of our framework in diverse real-world scenarios.

**Acknowledgements.** This research was supported by the Centre for Assuring Autonomy (CfAA), a partnership between Lloyd's Register Foundation and the University of York (https://www.york.ac.uk/assuring-autonomy/).

## References


[1] Project's GitHub. https://github.com/Gricel-lee/RV-OutOfODD/tree/main-v1-SEAMS26.

[2] Dhaminda B. Abeywickrama, Michael Fisher, Frederic Wheeler, and Louise Dennis. 2025. Towards Patterns for a Reference Assurance Case for Autonomous Inspection Robots . In *2025 IEEE/ACM 22nd International Conference on Software and Systems Reuse (ICSR)*. IEEE Computer Society, 95–100. doi:10.1109/ICSR66718.2025.00016

[3] Naif Alasmari, Radu Calinescu, Colin Paterson, and Raffaela Mirandola. 2022. Quantitative verification with adaptive uncertainty reduction. *Journal of Systems and Software* 188 (2022), 111275.

[4] Rob Alexander, Heather Rebecca Hawkins, and Andrew John Rae. 2015. Situation coverage–a coverage criterion for testing autonomous robots. (2015).

[5] Héctor Avilés, Marco Negrete, Alberto Reyes, Rubén Machucho, Karelly Rivera, Gloria de-la Garza, and Alberto Petrilli. 2024. Autonomous Behavior Selection For Self-driving Cars Using Probabilistic Logic Factored Markov Decision Processes. *Applied Artificial Intelligence* 38, 1 (2024), 2304942.

[6] Christel Baier and Joost-Pieter Katoen. 2008. *Principles of model checking*. MIT press.

[7] Radu Calinescu, Simos Gerasimou, Kenneth Johnson, and Colin Paterson. 2018. Using runtime quantitative verification to provide assurance evidence for self-adaptive software: advances, applications and research challenges. In *Software Engineering for Self-Adaptive Systems III. Assurances: International Seminar, Dagstuhl Castle, Germany, December 15-19, 2013, Revised Selected and Invited Papers*. Springer, 223–248.

[8] Radu Calinescu, Sinem Getir Yaman, Simos Gerasimou, Gricel Vázquez, and Micah Bassett. 2025. Verification and External Parameter Inference for Stochastic World Models. *2026 IEEE/ACM 48th IEEE International Conference on Software Engineering* (2025). https://arxiv.org/abs/2503.16034

[9] Radu Calinescu, Carlo Ghezzi, Kenneth Johnson, Mauro Pezzé, Yasmin Rafiq, and Giordano Tamburrelli. 2015. Formal verification with confidence intervals to establish quality of service properties of software systems. *IEEE transactions on reliability* 65, 1 (2015), 107–125.

[10] Radu Calinescu, Calum Imrie, Ravi Mangal, Genaína Nunes Rodrigues, Corina Păsăreanu, Misael Alpizar Santana, and Gricel Vázquez. 2022. Discrete-event controller synthesis for autonomous systems with deep-learning perception components. *arXiv preprint arXiv:2202.03360* (2022).

[11] Javier Cámara, David Garlan, Bradley Schmerl, and Ashutosh Pandey. 2015. Optimal planning for architecture-based self-adaptation via model checking of stochastic games. In *Proceedings of the 30th annual ACM symposium on applied computing*. 428–435.

[12] Nicolás Cardozo and Ivana Dusparic. 2021. Adaptation to Unknown Situations as the Holy Grail of Learning Based Self-Adaptive Systems: Research Directions. In *16th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, SEAMS@ICSE 2021, Madrid, Spain, May 18-24, 2021*. IEEE, 252–253. doi:10.1109/SEAMS51251.2021.00041

[13] Shang Gao, Zhixin Huang, Ghassan Al-Falouji, Bernhard Sick, and Sven Tomforde. 2025. Towards Cognitive Situational Awareness in Maritime Traffic Using Federated Evidential Learning. In *2025 IEEE Conference on Cognitive and Computational Aspects of Situation Management (CogSIMA)*. 9–16. doi:10.1109/CogSIMA64436.2025.11079521

[14] Simos Gerasimou, Radu Calinescu, and Alec Banks. 2014. Efficient runtime quantitative verification using caching, lookahead, and nearly-optimal reconfiguration. In *Proceedings of the 9th international symposium on software engineering for adaptive and self-managing systems*. 115–124.

[15] Victoria J Hodge and Matt Osborne. 2025. Agile Development for Safety Assurance of Machine Learning in Autonomous Systems (AgileAMLAS). *Array* 27 (2025), 100482.

[16] Victoria J. Hodge, Colin Paterson, and Ibrahim Habli. 2025. Out-of-Distribution Detection for Safety Assurance of AI and Autonomous Systems. arXiv:2510.21254 [cs.AI] https://arxiv.org/abs/2510.21254

[17] Calum Imrie, Rhys Howard, Divya Thuremella, Nawshin Mannan Proma, Tejas Pandey, Paulina Lewinska, Ricardo Cannizzaro, Richard Hawkins, Colin Paterson, Lars Kunze, et al. 2024. Aloft: self-adaptive drone controller testbed. In *Proceedings of the 19th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*. 70–76.

[18] Erblin Isaku, Hassan Sartaj, and Shaukat Ali. 2025. Digital Twin-based Out-of-Distribution Detection in Autonomous Vessels. arXiv:2504.19816 [cs.RO] https://arxiv.org/abs/2504.19816

[19] Marta Kwiatkowska, Gethin Norman, and David Parker. 2007. Stochastic model checking. In *International School on Formal Methods for the Design of Computer, Communication and Software Systems*. Springer, 220–270.

[20] Marta Kwiatkowska, Gethin Norman, and David Parker. 2011. PRISM 4.0: Verification of probabilistic real-time systems. In *Computer Aided Verification: 23rd International Conferenc (CAV). Proceedings 23*. Springer, 585–591.

[21] J. Lee et al. 2025. Enhancing Safety in Autonomous Maritime Transportation Systems with Real-Time AI Agents. *Applied Sciences* 15, 9 (2025), 4986. doi:10.3390/app15094986

[22] Gabriel A Moreno, Javier Cámara, David Garlan, and Bradley Schmerl. 2015. Proactive self-adaptation under uncertainty: a probabilistic model checking approach. In *Proceedings of the 2015 10th joint meeting on foundations of software engineering*. 1–12.

[23] Society of Automotive Engineers. 2018. Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles. https://www.sae.org/standards/j3016_201806-taxonomy-definitions-terms-related-driving-automation-systems-road-motor-vehicles

[24] Nawshin Mannan Proma and Rob Alexander. 2023. Systematic Situation Coverage versus Random Situation Coverage for Safety Testing in an Autonomous Car Simulation. In *Procs of the 12th Latin-American Symposium on Dependable and Secure Computing* (<conf-loc>, <city>La Paz</city>, <country>Bolivia</country>, </conf-loc>) *(LADC '23)*. 208–213. https://doi.org/10.1145/3615366.3625077

[25] Nawshin Mannan Proma, Gricel Vazquez Flores, Sepeedeh Shahbeigi Roudposhti, Arjun Badyal, and Victoria J Hodge. 2025. Probabilistic Safety Verification for an Autonomous Ground Vehicle: A Situation Coverage Grid Approach. In *2025 IEEE International Conference on Vehicular Electronics and Safety*. IEEE.

[26] Nawshin Mannan Proma, Victoria J. Hodge, and Rob Alexander. 2025. SCALOFT: An Initial Approach for Situation Coverage-Based Safety Analysis of an Autonomous Aerial Drone in a Mine Environment. In *Accepted for, 44th International Conference on Computer Safety, Reliability and Security (safecomp 2025)*. https://arxiv.org/abs/2505.20969.

[27] Salil Purandare, Urjoshi Sinha, Md Nafee Al Islam, Jane Cleland-Huang, and Myra B. Cohen. 2023. Self-Adaptive Mechanisms for Misconfigurations in Small Uncrewed Aerial Systems. In *2023 IEEE/ACM 18th Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*. 169–180. doi:10.1109/SEAMS59076.2023.00030

[28] Yasmin Rafiq, Gricel Vázquez, Radu Calinescu, Sanja Dogramadzi, and Robert M Hierons. 2025. Symbolic Runtime Verification and Adaptive Decision-Making for Robot-Assisted Dressing. In *Euromicro Conference on Software Engineering and Advanced Applications*. Springer, 290–308.

[29] Zaid Tahir and Rob Alexander. 2021. Intersection focused situation coverage-based verification and validation framework for autonomous vehicles implemented in Carla. In *International Conference on Modelling and Simulation for Autonomous Systems*. Springer, 191–212.

[30] Tobias Rye Torben. 2023. Formal approaches to design and verification of safe control systems for autonomous vessels. *PhD Thesis, NTNU: Norwegian University of Science and Technology, https://hdl.handle.net/11250/3059350* (2023).

[31] Gricel Vázquez, Alexandros Evangelidis, Sepeedeh Shahbeigi, and Simos Gerasimou. 2025. Adaptive Human-Robot Collaborative Missions using Hybrid Task Planning. In *2025 IEEE/ACM 20th Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*. IEEE, 73–84.

[32] Danny Weyns. 2020. *An introduction to self-adaptive systems: A contemporary software engineering perspective*. John Wiley & Sons.

[33] Danny Weyns and Jesper Andersson. 2023. From self-adaptation to self-evolution leveraging the operational design domain. In *2023 IEEE/ACM 18th Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*. IEEE, 90–96.

[34] Danny Weyns, Radu Calinescu, Raffaela Mirandola, Kenji Tei, Maribel Acosta, Nelly Bencomo, Amel Bennaceur, Nicolas Boltz, Tomas Bures, Javier Camara, et al. 2023. Towards a research agenda for understanding and managing uncertainty in self-adaptive systems. *ACM SIGSOFT Software Engineering Notes* 48, 4 (2023), 20–36.

[35] Xingyu Zhao, Radu Calinescu, Simos Gerasimou, Valentin Robu, and David Flynn. 2020. Interval change-point detection for runtime probabilistic model checking. In *Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering*. 163–174.

[36] Xingyu Zhao, Simos Gerasimou, Radu Calinescu, Calum Imrie, Valentin Robu, and David Flynn. 2024. Bayesian learning for the robust verification of autonomous robots. *Communications Engineering* 3, 1 (2024), 18.