Assignment WASP SE 2025: Enhancing Safety Assurance for Autonomous Surface Vehicles by Model-Based Test Generation

Dominik Frey

1 Introduction

Background In recent years, significant efforts have been directed towards developing *autonomous surface vehicles* (ASVs) for applications such as oceanography and cargo transport. A critical requirement for ASVs in these contexts is the ability to complete missions without posing risks to other maritime traffic. To prevent collisions, the *International Maritime Organization* (IMO) has established the *International Regulations for Preventing Collisions at Sea* (COLREGS) [3], a set of 41 rules that guide vessels in close-encounter situations to avoid collisions.

Challenges Since these regulations were designed with human operators in mind, many rules are subject to subjective interpretation and rely on human judgment. This presents a major challenge when it comes to ensuring that ASV control systems comply with COLREGS. Effective methods for ensuring COLREGS compliance in complex maritime traffic scenarios, such as those involving multiple vessels or static obstacles, are lacking. These scenarios may involve rare event combinations that traditional (stochastic) simulations fail to cover adequately. Nevertheless, sophisticated naval simulators offer a promising means of identifying potential safety risks (e.g., near-collision situations) without endangering human lives or maritime operations.

Added value My doctoral project seeks to improve system-level safety assurance of ASVs by developing a multi-level, model-based test generation approach utilizing qualitative abstractions. This method offers two key advantages: (1) improved scalability by reducing redundant testing within the same equivalence class of scenarios and (2) enhanced safety assurance by enabling formal guarantees of coverage and scenario diversity. Leveraging ASV's formal safety assurances and scenario diversity in testing would enhance the reliability and robustness of autonomous control systems under challenging conditions. Finally, this project's work with COLREGS compliance could directly support organizations that develop control software to prioritize safety and regulatory adherence.

Objectives To realize this vision, a list of objectives is set that serves as a road map for this research project: **O1:** Provide a specification language or refine and incorporate existing ones for system-level ASV test scenarios. **O2:** Automatically generate diverse test scenarios for multi-vessel encounters. **O3:** Integrate and execute test scenarios on real autonomous control software within existing naval simulators. **O4:** Ensure robustness by varying environmental conditions, identifying uncovered behaviors and introducing a feedback loop to improve test generation.

2 Lecture Principles

2.1 Static, Dynamic, and Process V&V Techniques

Software V&V encompasses multiple dimensions of quality assurance. *Static* V&V refers to analyses done without executing the system: requirements and design reviews, code inspections, and automated static analysis checks. These techniques ensure conformance to specifications early on. For example, we perform code reviews of our ASV scenario-generation software and use static analysis tools to catch errors before runtime. Static checks also include formal model verification of COLREGS rules, which can detect inconsistencies in the abstract scenario models.

Dynamic V&V involves executing the system under test. This includes unit testing, integration tests, system tests, and simulations. In our project, dynamic testing means running the ASV control software in a simulator or real environment with generated scenarios. Techniques like Monte Carlo [8] simulation or stochastic testing can be applied here using our generated high-level scenarios as seeds. We validate ASV behavior by simulating scenarios where multiple vessels follow COLREGS rules. Dynamic tests reveal how the ASV responds to actual ship movements and speeds.

The third V&V category is *process evaluation*. This assesses the quality of the software development process itself. Process-based assurance (e.g., CMMI [2], ISO standards [1], or the V-model) involves audits, peer reviews, and adherence to defined workflows. As we are in an early stage of non-commercial software development, we do not comply with a high process maturity level; however, we do our best to follow best coding practices, transparency, and documentation guidelines.

2.2 Behavioral Software Engineering

Behavioral Software Engineering [7] (BSE) studies the cognitive, social, and human aspects of software development. It emphasizes how engineers' behaviors, decisions, and interactions affect project outcomes. In the context of ASV testing, BSE tells us that humans remain involved: maritime experts define COLREGS-based goals; operators may interact with the ASV; and developers must interpret these requirements. While our tested ASVs are largely autonomous, human-in-the-loop elements like remote operators can be viewed through BSE. Recognizing this, we design our requirements and tools to accommodate human interpretations: we collaborate with domain experts to interpret underspecified COLREGS rules, and we provide visualization tools to help developers understand generated scenarios, acknowledging behavioral factors (such as the tendency to overlook corner cases).

3 Guest-Lecture Principles

3.1 Problem-Space vs. Solution-Space

A key requirement-engineering principle is to distinguish the *problem space* from the *solution space*. The problem space represents the users' or stakeholders' needs and goals, while the solution space encompasses the designs and implementations that could address those needs. As one discussion notes, for a given problem, there may be many possible solutions, and vice versa. A vague problem definition leads to an unfocused solution space; a narrow one may exclude valid solutions. Lean and product strategies stress thorough problem discovery, as poor understanding often results in low-value products.

For our ASV project, the problem space involves maritime safety goals: for example, ”ensure no collisions in open water under COLREGS constraints.” The solution space includes candidate test-generation techniques, simulation architectures, and collision avoidance algorithms. Navigating the problem space might involve interviewing naval experts or analyzing accident reports and the CORLEGS to identify hazardous encounter patterns. Only once the safety requirements are well-scoped can we systematically explore solutions (which abstraction level to use, what algorithms to generate scenarios). In practice, we perform problem discovery (e.g., refining the COLREGS model) and solution prototyping (e.g., testing scenario generation ideas), iteratively.

3.2 Requirements Elicitation for System Goals

Requirements elicitation is the process of gathering and defining the system's goals and constraints from stakeholders. It typically involves communication, collaboration, and iterative refinement. For ASV testing, elicitation means uncovering the precise interpretation of COLREGS rules and safety priorities. This might include workshops with maritime experts, interviews with potential ASV operators, and a review of regulations. In our case, we elicit system goals such as: ”The test suite must cover all semantically distinct scenarios defined by the COLREGS.” We then translate these high-level goals into formal scenario requirements. Missing requirements can lead to failure. By conducting thorough elicitation (prototyping simulations, reviewing COLREGS, etc.), we aim to capture all critical goals and avoid gaps in the ASV safety specification.

4 Data Scientists vs. Software Engineers

I agree with the distinction in the CMU book[6] between data scientists and software engineers. The first ones focus on modeling, experimentation, and extracting predictive value from data, while the second ones focus on scalable systems, reliability, and robust code. Data scientists typically evaluate success based on statistical metrics, model accuracy, and insights in notebook-based environments, often with limited concern for deployment latency, fault tolerance, or integration constraints. Software engineers, on the other hand, prioritize software architectures, reusable code bases, performance optimization, and rigorous testing. These differences reflect real-world tensions: teams frequently struggle over responsibilities and mutual expectations due to separate tooling and educational backgrounds.

In the future, I expect further specialization, not the complete convergence of roles. New positions, such as Machine-Learning Engineer, MLOps Specialist, and AI Platform Engineer, have been created to bridge the gap between modeling and production operations. These roles sit at the intersection of both fields, combining model evaluation, infrastructure design, and automated CI/CD. Organizations are building T-shaped teams, each member has deep domain expertise plus a broad spectrum of superficial literature knowledge to collaborate effectively. As this trend grows, data scientists will learn infrastructure best practices, while engineers will grow more comfortable with model drift and validation, without requiring every individual to master both disciplines fully.

5 Paper Analysis

I selected two full CAIN conference papers that illustrate AI engineering principles in some ways relevant to my research: *LLM-Based Safety Case Generation for Baidu Apollo: Are We There Yet?* [9] and *Approach for Argumenting Safety on Basis of an Operational Design Domain*[10].

5.1 LLM-Based Safety Case Generation for Baidu Apollo: Are We There Yet? (CAIN 2025)

Core ideas and their SE importance The paper introduces and evaluates both manual and LLM-based automated methods for generating safety cases for ML-enabled components of autonomous systems, using *Baidu Apollo*'s trajectory prediction module as a case study.

It uses *AMLAS-based* [5] design methodology to construct assurance cases manually and evaluates the use of *GPT-4o* to generate safety cases from assurance case patterns automatically. The study demonstrates that combining human expertise and LLMs results in faster, semi-automatic, and high-quality assurance case creation.

The work addresses (1) a critical bottleneck in certifying safety-critical AI systems; (2) a path to scale safety engineering practices to complex AI systems; (3) traceability, formal argumentation, and compliance with safety standards in AI-intensive domains.

Relation to my research We focus on system-level safety assurance for ASVs, particularly on COLREGS-compliant test generation through model-based approaches. Both works target safety-critical autonomous systems involving ML-based decision making in dynamic environments.

Our scenario generation and formal guarantees of coverage align well with their aim of formally structured safety cases backed by argument patterns and evidence. Their use of AMLAS and argumentation patterns could complement our test generation pipeline by helping to structure and justify the safety relevance of generated test scenarios under regulatory constraints like COLREGS.

Integration into a larger AI-intensive project Let us take a hypothetical project that is an autonomous maritime logistics platform, an AI-driven system enabling fleets of ASVs to autonomously transport cargo in congested sea lanes, coordinating with ports, ships, and maritime traffic control systems.

The paper's LLM-assisted methodology could be used to semi-automatically generate safety cases for each AI-intensive component (e.g., path planning, intent prediction, collision avoidance) of ASVs. By instantiating assurance cases from COLREGS-compliance patterns, the platform could demonstrate adherence to international maritime safety laws.

Our model-based test generation would feed into these safety cases as evidence of behavioral robustness and coverage of hazardous scenarios in multi-vessel interactions. Our scenario abstractions and coverage guarantees could be used to populate and refine assurance case arguments automatically or semi-automatically, closing the loop between testing and safety documentation.

Adaptation of my research To better align with the AI-engineering ideas in the paper, we could (1) integrate AMLAS into our workflow to explicitly structure the rationale behind scenario generation and test execution as safety arguments; (2) adopt *Goal Structuring Notation* [4] (GSN) to document and communicate the traceability between test objectives, scenarios, and COLREGS rules; (3) use their approach to formalize scenario specification templates as assurance case patterns, making test generation artifacts reusable in assurance cases.

5.2 Approach for Argumenting Safety on Basis of an Operational Design Domain (CAIN 2024)

Core ideas and their SE importance The paper proposes a structured approach to defining and maintaining a complete and consistent *Operational Design Domain* (ODD) to serve as a foundation for safety assurance in AI-based *Autonomous Driving Systems* (ADS). The paper introduces a set of evidences that support claims of completeness and consistency in the ODD definition, which is essential for developing test cases, verifying AI components, and constructing sound safety cases.

ODD completeness/consistency directly affects the validity of test scenarios, performance of AI models, and credibility of safety arguments. It supports traceability, standard compliance, and risk management in AI systems where real-world environmental conditions influence behavior. It also helps overcome the "black box" challenge in ML by contextualizing system behavior within defined operational boundaries.

Relation to my research We aim to enhance ASV safety through model-based, diverse test generation with COLREGS compliance, which aligns with this paper, as ADS and ASVs operate in open, uncertain, multi-agent environments where clearly defined operational limits are important.

Our scenario generation depends on understanding environmental and contextual parameters, just like building an ODD for maritime domains. The ODD-based safety argumentation and scenario-based testing discussed in the paper share our objectives to guarantee scenario diversity and coverage formally.

Integration into a larger AI-intensive project Let us take the previous hypothetical project. The proposed approach in the paper could contribute to the development of a maritime ODD framework that captures environmental conditions (sea state, weather), dynamic elements (other vessels, port traffic), and regulatory constraints. Furthermore, it enables the integration of ODD-related evidence, such as in-field data and expert reviews, to support the safety certification of ship AI systems.

Our test generation system provides scenario coverage and diversity over the maritime ODD. The use of qualitative abstractions and equivalence classes helps define the characteristics of a complete and non-redundant test space. Formal guarantees can enhance verification of control software in the project by formalizing COLREGS compliance.

Adaptation of my research To better align with the AI-engineering ideas in the paper, we could (1) explicitly define a maritime ODD using a domain-specific language or ontology; (2) incorporate ODD completeness and consistency checks into our scenario generation pipeline; (3) compare generated scenarios with in-field ASV data (AIS logs, simulated trajectories); (4) add ODD-aware feedback loops in our scenario generator when a near-miss or uncovered behavior occurs, then enrich the ODD and regenerate affected scenarios.

6 Research Ethics & Synthesis Reflection

Search and screening process To find the two CAIN papers, I used web search and conference archives. I first examined the official CAIN 2024–25 program pages (via the researchr.org site) to list accepted papers and located titles that matched my interests, like topics in safety and verification of autonomous systems. I prioritized papers with available PDFs or preprints. Each candidate was screened by reading their abstract and introduction to assess relevance to AI and/or autonomous safety.

Pitfalls and mitigations I mostly encountered titles that looked promising, containing keywords like ”debugging”, ”runtime analysis”, ”dataset generation”, etc. However, some of the papers were using an AI model-specific adversarial image input or random data augmentation to invoke faulty behavior, which is not of interest to my research. Some other papers were not accessible due to paywalls. To mitigate these issues, I had to screen more papers until I found the preferred two.

Ethical considerations I adhered to academic honesty by citing all external sources that were mentioned. I have not included any confidential or private information, and any software or datasets mentioned are either hypothetical or from published sources. I have used AI tools only to summarize and paraphrase, double-checking all outputs against source materials to prevent hallucination.

References

[1] Iso/iec/ieee international standard - systems and software engineering – life cycle processes – requirements engineering. *ISO/IEC/IEEE 29148:2018(E)*, pages 1–104, 2018.

[2] Mary Beth Chrissis, Michael Konrad, and Sandra Shrum. *CMMI for Development: Guidelines for Process Integration and Product Improvement, 3rd Edition*. Mar 2011. Accessed: 2025-Aug-4.

[3] UCG Commandant. International regulations for prevention of collisions at sea, 1972 (72 colregs). *US Department of Transportation, US Coast Guard, Commandant Instruction M*, 16672, 1999.

[4] G. S. N. S. W. Group. GSN (Version 3), 2023. Accessed on November 30, 2023.

[5] Richard Hawkins, Matt Osborne, Mike Parsons, Mark Nicholson, John McDermid, and Ibrahim Habli. Guidance on the safety assurance of autonomous systems in complex environments (sace), 2022.

[6] Christian Kastner. *Machine learning in production*. MIT Press, London, England, April 2025.

[7] Per Lenberg, Robert Feldt, and Lars Göran Wallgren. Behavioral software engineering: A definition and systematic literature review. *Journal of Systems and Software*, 107:15–37, September 2015.

[8] Nicholas Metropolis and S. Ulam. The monte carlo method. *Journal of the American Statistical Association*, 44(247):335–341, September 1949.

[9] Oluwafemi Odu, Alvine Boaye Belle, and Song Wang. Llm-based safety case generation for baidu apollo: Are we there yet? In *2025 IEEE/ACM 4th International Conference on AI Engineering – Software Engineering for AI (CAIN)*, page 222–233. IEEE, April 2025.

[10] Gereon Weiss, Marc Zeller, Hannes Schoenhaar, Christian Drabek Fraunhofer, and Andreas Kreutz. Approach for argumenting safety on basis of an operational design domain. In *2024 IEEE/ACM 3rd International Conference on AI Engineering – Software Engineering for AI (CAIN)*, pages 184–193, 2024.