

SCENE GEMS: Scenario Generation and Execution Framework for the Safety Assurance of Maritime Autonomous Surface Vehicles

Dominik Frey
Linköping University
Linköping, Sweden
dominik.frey@liu.se

Ulf Kargén
Linköping University
Linköping, Sweden
ulf.kargen@liu.se

Dániel Varró
Linköping University
Linköping, Sweden
daniel.varro@liu.se

Abstract

Autonomous surface vehicles (ASVs) must comply with the International Regulations for Preventing Collisions at Sea (COLREGS) to operate safely in shared maritime environments at open sea. While simulation-based testing is widely used to assess compliance, existing approaches lack a complete and scalable assurance workflow. Instead, they typically focus on scenario generation, simulation, or compliance monitoring in isolation. SCENE GEMS is an integrated modular framework for systematic assurance of ASV systems. It (1) uses model-driven techniques to generate functionally complete assurance suites from high-level specifications. Furthermore, SCENE GEMS (2) instantiates and executes generated scenarios in a realistic simulator, and (3) monitors safety and COLREGS compliance at runtime, generating real-time feedback and exportable assurance evidence for later analysis. The simulated multi-vessel encounters are displayed through interactive visualization. Practitioners can configure the simulation environment, communication interfaces, and vessel-specific physics models. The ASV system under test is integrated via software-in-the-loop or hybrid setups that involve both real and simulated vessels.

Demonstration: https://www.youtube.com/watch?v=8S7yY4I_81M
Repository: <https://github.com/PELAB-LiU/SceneGEMS>

ACM Reference Format:

Dominik Frey, Ulf Kargén, and Dániel Varró. 2026. SCENE GEMS: Scenario Generation and Execution Framework for the Safety Assurance of Maritime Autonomous Surface Vehicles. In *ACM/IEEE 29th International Conference on Model Driven Engineering Languages and Systems (MODELS Companion 2026)*, October 04–09, 2026, Málaga, Spain. ACM, New York, NY, USA, 5 pages. <https://doi.org/10.1145/3837062.3838872>

1 Introduction

The development of *autonomous surface vehicles* (ASVs) for civilian and military applications requires rigorous assurance mechanisms to ensure safe operation in shared maritime environments. To prevent near misses or collisions, the International Maritime Organization (IMO) established the *International Regulations for Preventing Collisions at Sea* (COLREGS) [5], which define navigation rules for vessels in open-sea encounters. Furthermore, in May 2026, IMO adopted a new *International Code of Safety for Maritime Autonomous Surface Ships* (MASS Code)¹, which highlights the growing need

¹<https://www.imo.org/en/mediacentre/hottopics/pages/autonomous-shipping.aspx>



This work is licensed under a Creative Commons Attribution 4.0 International License. *MODELS Companion 2026, Málaga, Spain*
© 2026 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-2903-4/2026/10
<https://doi.org/10.1145/3837062.3838872>

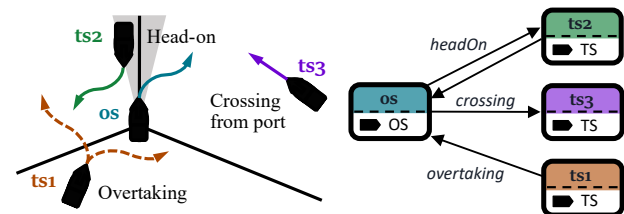


Figure 1: Example 4-vessel COLREGS scenario from [7].

for assurance frameworks capable of systematically demonstrating the safety and COLREGS compliance of ASVs.

Since the exhaustive real-world testing of COLREGS compliance is infeasible, ASV systems are typically evaluated using simulation-based testing [11, 16, 17]. As discussed in [10], such evaluations rely on test scenarios representing multi-vessel encounters that are challenging from a COLREGS perspective.

COLREGS scenarios. The COLREGS apply when two vessels are within visual range and on a potential collision course at open sea. In such cases, the encounter is classified into a COLREGS *situation* [5], based on relative bearing sectors defined by the COLREGS. The COLREGS stipulate different maneuvering responsibilities depending on whether the situation is classified as *overtaking* (Rule 13), *head-on* (Rule 14), or *crossing from port* (Rule 15).

Multiple situations may coexist in a multi-vessel encounter, referred to as a COLREGS *scenario*. On the left side of Figure 1, we illustrate these concepts in a four-vessel scenario adapted from [7]. In this example, each *target ship* ts_1 , ts_2 , and ts_3 is in a COLREGS situation with the *own ship* (os) forming a COLREGS scenario.

A COLREGS scenario is *relevant* for ASV testing if the own ship is exposed to high risk by being in a COLREGS situation with each target ship. Additionally, the environment must remain COLREGS-compliant by avoiding situations among the target ships [10]. This reduces the complexity of constructing test cases by focusing on the interactions between the own ship and target ships.

A test case typically consists of (1) an initial scene of a COLREGS scenario defining vessel positions, headings, and speeds, (2) a set of target ship trajectories, and (3) a test oracle that evaluates the COLREGS compliance of an ASV system throughout the encounter.

Unlike a test suite, which is designed to detect bugs, an *assurance suite* is intended to provide evidence of compliance. It comprises a *functionally complete* set of semantically distinct, relevant, and feasible COLREGS scenarios [7].

Problem statement. To the best of our knowledge, no existing framework provides end-to-end simulation-based assurance for ASVs. Although automated scenario generation techniques exist [4,

18], they do not integrate simulation execution and compliance evaluation. Tools that do combine some of these capabilities are not designed to execute assurance suites that systematically cover all semantically distinct cases required for COLREGS compliance. They either rely on simplified simulators that sacrifice realism [11, 14], limiting the validity of assurance results, or they are implemented as monolithic systems that cannot execute realistic multi-vessel scenarios due to limited scalability [2, 16].

The proposed tool. We present SCENE GEMS, an integrated modular framework for systematic ASV assurance. The tool provides (1) generation of functionally complete assurance suites from high-level specifications, (2) execution of test scenarios in a state-of-the-art maritime simulator, and (3) runtime safety and COLREGS compliance monitoring. The architecture is designed to support distributed execution for large-scale, realistic, and reproducible simulations with software-in-the-loop and hybrid setups involving both real and simulated vessels.

SCENE GEMS supports an end-to-end assurance workflow. First, the user selects one or more high-level functional scenario specifications (or predefined presets) and a set of domain constraints. SCENE GEMS then generates a corresponding scene for each selected specification, providing live feedback during scene generation and interactive visualization of the resulting scenarios. Generated scenarios can be instantiated for simulation-based testing by configuring the simulation environment, communication interfaces, and optionally vessel-specific physics models. The ASV system under test is integrated through a well-defined communication interface. During execution, the tool continuously monitors and evaluates COLREGS compliance at runtime, providing real-time visualization and exportable assurance evidence for post-analysis.

End users. SCENE GEMS targets ASV system developers, testers, and maritime safety analysts seeking simulation-based assurance evidence for the COLREGS compliance of their ASV systems.

2 Frontend and Usage

The frontend is a single-page REACT application that communicates with the SCENE GEMS integration layer via a WebSocket interface. On the web interface, the user can configure the required inputs, start and manage assurance workflows, and visualize the results.

Functional scenario specifications as input. To model COLREGS scenarios on a higher level of abstraction, a metamodel is defined in [7]. We show a simplified metamodel in Listing 1, expressed in *Refinery*² [15] specification language. COLREGS situations and vessel classes are represented as abstract binary relations.

```
abstract class Vessel {
  Vessel[0..*] overtaking
  Vessel[0..*] crossing
  Vessel[0..*] headOn opposite headOn
  VesselType[1] vesselType
class OS extends Vessel. class TS extends Vessel.
enum VesselType {PassengerShip, SailingBoat, Tanker}
```

Listing 1: Metamodel for functional COLREGS scenarios.

²<https://refinery.services>

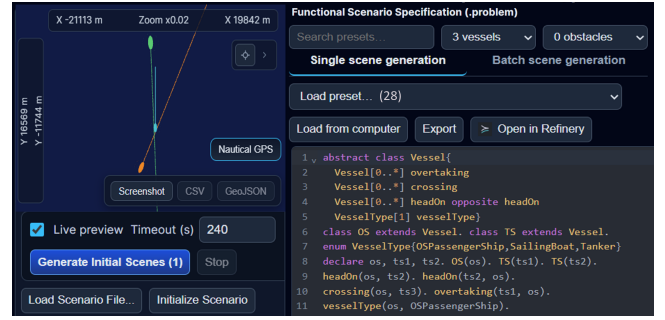


Figure 2: Scenario generation interface using Listing 2.

Functional scenarios are instance graph models over the meta-model, specified using (first-order logic) constraints and assertions. The graph representation of our example COLREGS scenario is shown on the right side of Figure 1 along with the corresponding Refinery specification in Listing 2. Apart from the COLREGS situations, *os* is explicitly assigned to be a *passenger ship* while the type of the target ships is left undefined.

```
declare os, ts1, ts2, ts3.
OS(os). TS(ts1). TS(ts2). TS(ts3).
headOn(os, ts2). headOn(ts2, os).
crossing(os, ts3). overtaking(ts1, os).
vesselType(os, PassengerShip).
```

Listing 2: Example functional scenario specification.

Functional scenarios represent relevant COLREGS test cases. A functionally complete assurance suite is achieved by first generating functional models with semantically distinct graph shapes, and then deriving a concrete test scenario for each model [6].

Domain constraints as input. To specify how functional models are interpreted in concrete maritime test scenarios, we use *domain constraints* that consist of vessel and COLREGS constraints. *Vessel constraints* define the physical scale and capabilities of vessel types as numeric boundaries for length, beam, speed, angular velocity, and acceleration.

Different collision avoidance rules apply to vessels, depending on their type and the respective COLREGS situation. These rules are not explicitly parametrized in the COLREGS (e.g., the time for a vessel to start giving way, the minimum turn considered readily apparent, or the safe spatial and temporal distance between vessels). These parameters are called *COLREGS constraints*, and they are configured based on scenario scale and assumptions used in existing literature on the COLREGS.

Scenario generation workflow. An overview of the scenario generation interface is shown in Figure 2. Our tool provides a complete set of functional scenario models, which are available as presets to avoid manual construction from scratch. Functional scenarios are defined using Refinery partial models, with editable presets and import/export support. Domain constraints can be loaded, edited, and exported in YAML with syntax highlighting and default values.

Scenario generation can be executed either as single runs or in batches, producing multiple concrete scenarios in a ZIP archive.

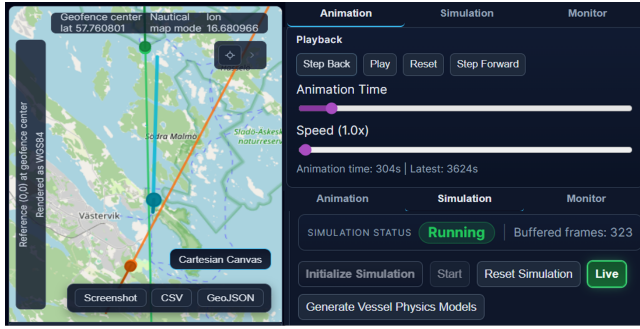


Figure 3: Simulation and animation control interface.

To support debugging and improve responsiveness, the generation status is continuously updated and visualized. Users can observe intermediate candidate scenes during sampling until a scene satisfying the configured constraints and specifications is found.

Simulation workflow. The interface allows users to initialize simulations from a selected scenario and configure the required simulation parameters, including vessel-specific connection settings.

External physics models can be uploaded for each vessel, or otherwise, the server generates estimated models from vessel attributes.

Before running the simulation, users can explore the loaded trajectories through a *scenario preview*, navigating each time step using the playback controls shown in Figure 3. Further controls are provided for starting and resetting runs, with continuous status feedback. During execution, the frontend displays a detailed visualization of the evolving scenario. The user can visually compare the scenario preview and the live simulation as they are displayed within a shared Cartesian or nautical GPS map. Preview and simulation frames can be exported in JSON, CSV, and GeoJSON formats.

Monitoring workflow. Monitor configuration similarly provides explicit connection controls for a specified runtime compliance monitor. The computed monitor results can be observed at each time step, for preview and live simulation streams side by side, allowing direct comparison. As the outline in Figure 4 shows, key metrics are displayed as evolving interactive plots, allowing users to track metrics for any selected actor pair, such as Euclidean distance, distance at *closest point of approach* (DCPA) and *time to closest point of approach* (TCPA) [8]. Vessel states, rule compliance, maneuver, and COLREGS specific status are displayed in a textual format that the user can export for later analysis.

3 Architecture

The architecture is composed of three subsystems responsible for *scenario generation*, *simulation execution*, and *runtime monitoring*, depicted in Figure 5. Subsystems are designed as loosely coupled components to support future integrations and extensions. Components are deployed in a containerized fashion to enable distributed execution across multiple hosts, which is essential for scalable simulation. A central *integration layer*, exposed through a web-based frontend, dynamically spawns and connects subsystems on demand.

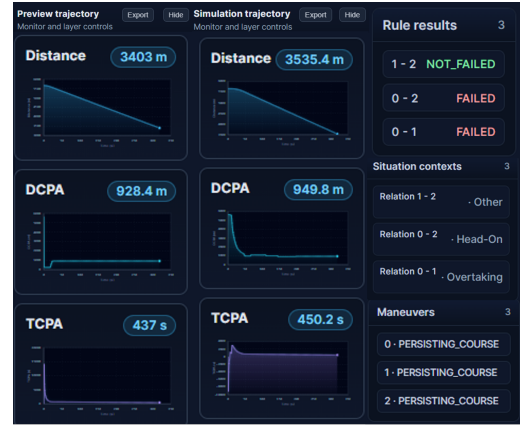


Figure 4: Runtime monitor result display interface.

3.1 Main Subsystems

Scenario Generation Subsystem (SGS). The SGS is responsible for deriving concrete initial scenes from abstract scenario specifications. It listens for generation requests containing (1) high-level functional scenario specifications and (2) domain constraints.

Upon request, the SGS generates a concrete initial scene. The implementation uses sampling-based multi-step scene generation by refinement [6], a state-of-the-art approach that provides guarantees for efficiently deriving a functionally complete assurance suite. Multiple concurrent requests can be processed, supporting fast batch scene generation.

The Scenario Execution Subsystem (SES). The SES is responsible for instantiating and executing simulations. It takes a concrete scenario, *vessel physics models*, and *simulation configurations* as input. Vessel physics models define attributes such as dimensions, mass, propulsion, hydrodynamics, and onboard sensors. Simulation configurations specify the simulation speed and environmental conditions (e.g., wind and waves). They also define the connection parameters and *control mode* for each vessel. A vessel is controlled either by automatic guidance along predefined trajectories in the scenario or by an externally connected ASV system.

The SES adopts a hybrid simulation architecture with centralized physics simulation and distributed vessel control. A *simulation instance* is organized into an agent layer, a control layer, and a physics layer. *Agents* are systems that publish heartbeat signals and sensor data (position, speed, heading, etc.), while capable of directly executing tasks (e.g., following a path). At the agent layer, agents receive missions as series of waypoints from concrete scenarios, translate them into executable tasks, and forward them to the control layer. At the control layer, *controllers* implemented as ARDUPILOT SITL³ instances provide both low- and mission-level control and state estimation from simulated sensor data. At the physics layer, a single *Gazebo Harmonic*⁴ instance runs the *simulated world* that includes hydrodynamics [3], buoyancy, weather effects, sensors, and *physics*

³<https://github.com/ArduPilot/ardupilot>

⁴<https://gazebo.org/docs/harmonic>

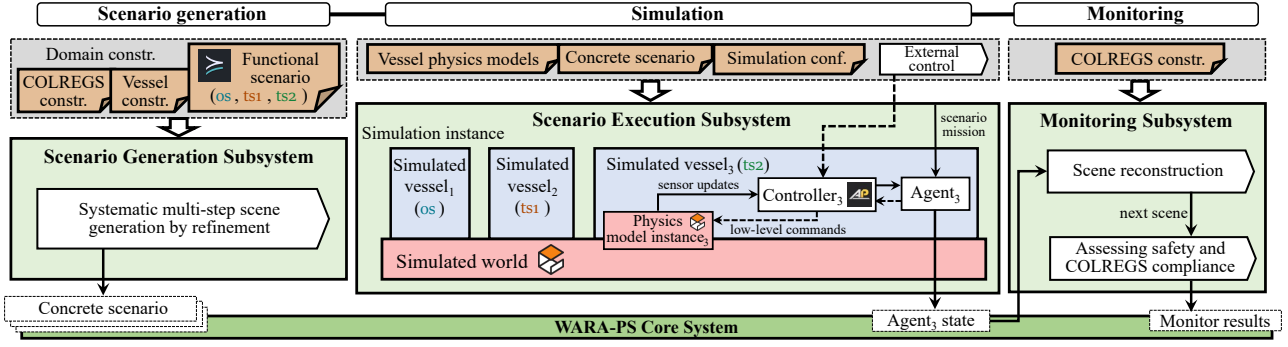


Figure 5: Architectural overview of the three subsystems showing inputs, main functionalities, and outputs.

model instances (PMIs) for each vessel. Gazebo simulates PMI responses to the assigned controller’s commands. This architecture reflects real-world deployments, where all vessels interact in the same environment but are controlled independently.

Upon request, the SES launches the simulated world together with a dedicated agent, controller, and PMI for each vessel in the scenario in standalone *Docker* containers. Agents communicate with their corresponding controller through MAVLink⁵, a widely used protocol in robotics, and publish vessel states (e.g., position, heading, speed). Communication between controllers and the simulated world is realized through the official ARDUPILOT–Gazebo plugin⁶.

The advantage of the controller-agent architecture is that it abstracts away complicated interfacing with the external simulator and provides an out-of-the-box controller for waypoint navigation, while allowing low-level control for ASV systems. External ASV systems can connect to selected controllers through an exposed MAVLink interface to take over control during execution. The architecture is agnostic to the underlying ASV platform, which enables integration of any external ASV systems, whether it controls a simulated vessel or a real-world vessel. Simulations execute in lockstep to ensure determinism, reproducibility, and consistent high-speed execution for future large-scale simulations. However, faster-than-real-time executions require external controllers to update at a sufficiently high frequency.

Monitoring Subsystem (MS). The MS is responsible for monitoring and evaluating simulation execution at runtime. It listens for scene updates constructed from snapshots of the published states of the vessels in a scenario. Upon request, the MS returns maritime-specific evaluation data, referred to as the *monitor result*.

We provide an internal runtime MS that evaluates COLREGS rules for individual maneuvers and pairwise COLREGS situations, as well as maritime safety metrics, such as distance, DCPA, and TCPA. Our MS maintains a trajectory history for temporal reasoning over maneuvers and the evolving COLREGS scenario.

3.2 Subsystem Integration

To achieve end-to-end simulation-based assurance, SCENE GEMS implements an *integration layer* for the presented subsystems shown in Figure 6. The integration layer has the following responsibilities:

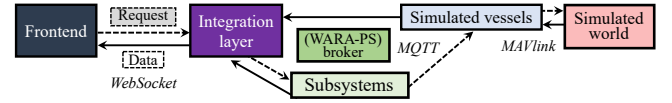


Figure 6: High-level subsystem integration overview.

- It exposes functionality to the frontend through WebSocket.
- It manages the lifecycles of subsystems and simulated vessels, coordinating data flow through an MQTT⁷ broker that enables lightweight communication.
- It forwards scene-generation requests to the SGS via an MQTT client, then returns status and generated scenes to the frontend.
- It spawns and initializes the SES, then controls and continuously reports simulation status.
- It spawns our internally provided MS or connects to an external one with a dedicated MQTT client. It creates monitoring requests to the MS from sensor data published by the agents. Monitor results are fetched from the MS and streamed to the frontend.

For external ASV control during simulation, the user must specify the *base topic*⁸ of the external agent, which corresponds to the MQTT topic used for communication through the broker. To receive state updates, the user must implement an MQTT subscriber that listens to the specified topic. In addition, a MAVLink port must be configured, enabling the user to connect and issue either low-level control commands or high-level mission commands.

For demonstrating simulations as part of a larger ecosystem, SCENE GEMS connects to the WARA-PS Core System⁹ [1], which is a common integration platform for heterogeneous autonomous agents deployed in public safety missions. To connect to the WARA-PS infrastructure, the user can choose to use the WARA-PS MQTT broker instead of our internal broker, given the right credentials.

4 Discussion

SCENE GEMS will be demonstrated at the WARA-PS Demonstration Week¹⁰ in September 2026, a major annual event in Sweden

⁵<https://github.com/mavlink>

⁶https://github.com/ArduPilot/ardupilot_gazebo

⁷<https://mqtt.org/>

⁸https://api-docs.waraps.org/#/agent_communication/topics/l1_topics

⁹<https://portal.waraps.org/page/the-core-system>

¹⁰<https://portal.waraps.org/page/event-demonstration-week-2026>

Table 1: Maritime simulation framework comparison.

Aspect / framework	SCENEEMS	[16]	[3]	[2]	[11]	[14]
Systematic scenario gen.	●	○	○	●	●	●
High-fidelity sim.	●	●	●	●	○	○
Runtime monitoring	●	●	○	○	●	●
Scalable architecture	●	●	○	○	○	○
End-to-end integration	●	○	○	○	○	○

● = Provides, ● = Partially provides, ○ = Does not provide

that brings together researchers, industry partners, government agencies, and end users to evaluate AI, autonomous systems, and robotics in realistic public-safety scenarios. The tool was conceptually validated and presented as a poster at the 2025 edition of the event. The development of the tool has been guided by continuous discussions with practitioners in autonomous-system safety.

Related work. Several tools support parts of the scenario-based assurance workflow, but none integrate it within a single end-to-end framework. A model-driven tool for autonomous road vehicle testing is provided in [9], combining abstract scenario specification, scenario concretization, execution in the external CARLA simulator, and evaluation. However, since the work targets road vehicles, it is not applicable in a maritime setting. For maritime, a simulation framework comparison summary is shown in Table 1. Existing platforms [2, 3, 16] provide realistic environments to evaluate ASV controllers, but they have limited scalability and leave scenario construction largely to the user. In [14] and [11], the authors combine scenario generation with simplified simulators and COLREGS monitoring, but they use scenarios that are extracted from historical encounters and thus insufficient for scenario-based assurance [7].

In contrast, our openly available tool integrates systematic generation of functionally complete assurance suites, execution in a realistic simulator, visualization, and runtime COLREGS compliance monitoring within a single easy-to-use interface.

Limitations and future work. The tool is prepared to support batch execution of simulations in the future, but this feature has not been implemented yet. It supports faster-than-real-time execution (run in lockstep), which is a practical prerequisite for batch execution.

With increasing scale, distributing simulations across multiple machines becomes necessary to maintain efficient execution. Although distributed deployment is left for future work, our architecture readily supports such scaling as both SES and the simulated vessels run in standalone Docker containers.

The SGS is currently limited to generating initial scenes, which is valuable in its own right, e.g., for evaluating COLREGS situation awareness [12]. From single initial scenes, we compute straight trajectories, but scenarios with full trajectories can be loaded externally for execution. Producing COLREGS compliant trajectories from initial scenes at scale is a challenging research problem [13]. In the future, we will integrate design-time and runtime trajectory planning for target ships, ensuring COLREGS compliance even with unpredictable ASV behavior.

Acknowledgments

This work relates to the Department of Navy Award N62909-24-1-2006 issued by the Office of Naval Research. The third author has

also been partially supported by the Wallenberg AI, Autonomous Systems, and Software Program (WASP), which is funded by the Knut and Alice Wallenberg Foundation, and Vetenskapsrådet (VR). The first author has been supported by the Linköping University Graduate School in Computer Science and is affiliated with WASP.

References

- [1] Olov Andersson, Patrick Doherty, Mårten Lager, Jens-Olof Lindh, Linnea Persson, Elin A. Topp, Jesper Tordenlid, and Bo Wahlberg. 2021. WARA-PS: a research arena for public safety demonstrations and autonomous collaborative rescue robotics experimentation. *Autonomous Intelligent Systems* 1, 1 (2021). doi:10.1007/s43684-021-00009-9
- [2] Mustafa Bayrak and Haluk Bayram. 2023. COLREG-Compliant Simulation Environment for Verifying USV Motion Planning Algorithms. In *OCEANS 2023 - Limerick*. IEEE, 1–10. doi:10.1109/oceanslimerick52467.2023.10244676
- [3] Brian Bingham, Carlos Agüero, Michael McCarrin, Joseph Klamó, Joshua Malia, Kevin Allen, Tyler Lum, Marshall Rawson, and Rumman Waqar. 2019. Toward Maritime Robotic Simulation in Gazebo. In *OCEANS 2019 MTS/IEEE SEATTLE*. IEEE, 1–10. doi:10.23919/oceans40490.2019.8962724
- [4] Victor Bolbot, Christos Gkerekos, Gerasimos Theotokatos, and Evangelos Boulougouris. 2022. Automatic traffic scenarios generation for autonomous ships collision avoidance system testing. *Ocean Engineering* 254 (2022), 111309. doi:10.1016/j.oceaneng.2022.111309
- [5] UCG Commandant. 1999. International regulations for prevention of collisions at sea, 1972 (72 COLREGS). *US Department of Transportation, US Coast Guard, Commandant Instruction M 16672* (1999).
- [6] Dominik Frey, Ulf Kargén, and Dániel Varró. 2026. Automated Generation of Functionally Complete Assurance Suites for COLREGS-Compliance of Autonomous Surface Vehicles. In *Proc. ACM/IEEE Int. Conf. Model Driven Eng. Lang. Syst. (MODELS)*. Accepted for publication.
- [7] Dominik Frey, Ulf Kargén, and Dániel Varró. 2025. Assessing Scene Generation Techniques for Testing COLREGS-Compliance of Autonomous Surface Vehicles. *Proc. ACM Softw. Eng.* 2, ISSTA (2025), 985–1008. doi:10.1145/3728919
- [8] Yixiong He, Yi Jin, Liwen Huang, Yong Xiong, Pengfei Chen, and Junmin Mou. 2017. Quantitative analysis of COLREG rules and seamanship for autonomous collision avoidance at open sea. *Ocean Engineering* 140 (2017), 281–291. doi:10.1016/j.oceaneng.2017.05.029
- [9] Jerry Hou-Liu, Zhekai Jiang, and Aren A. Babikian. 2024. Concretize: A Model-Driven Tool for Scenario-Based Autonomous Vehicle Testing. In *Proceedings of the ACM/IEEE 27th International Conference on Model Driven Engineering Languages and Systems (MODELS Companion '24)*. ACM, 66–70. doi:10.1145/3652620.3687793
- [10] Ulf Kargén and Dániel Varró. 2024. Towards Automated Test Scenario Generation for Assuring COLREGS Compliance of Autonomous Surface Vehicles. In *Proc. ACM/IEEE Int. Conf. Model Driven Eng. Lang. Syst. (MODELS) (MODELS '24, Vol. 222)*. ACM, 249–256. doi:10.1145/3640310.3674098
- [11] Hanna Krasowski and Stefan Schärddinger. 2025. CommonOcean-Sim: A Traffic Simulation Environment for Unmanned Surface Vessels. In *Proc. IEEE Int. Conf. Intell. Transp. Syst. (ITSC)*. IEEE, 966–972. doi:10.1109/itsc60802.2025.11423548
- [12] Yoshiaki Kuwata, Michael T. Wolf, Dimitri Zarghitsky, and Terrance L. Huntsberger. 2014. Safe Maritime Autonomous Navigation With COLREGS, Using Velocity Obstacles. *IEEE Journal of Oceanic Engineering* 39, 1 (2014), 110–119. doi:10.1109/JOE.2013.2254214
- [13] Changyu Lee, Jinwook Park, and Jinwhan Kim. 2026. Efficient COLREGS-compliant collision avoidance using turning circle-based control barrier function. *Mechatronics* 117 (2026), 103495. doi:10.1016/j.mechatronics.2026.103495
- [14] Eunkyu Lee, Junaid Khan, Umar Zaman, Jaebin Ku, Sanha Kim, and Kyungsup Kim. 2024. Synthetic Maritime Traffic Generation System for Performance Verification of Maritime Autonomous Surface Ships. *Applied Sciences* 14, 3 (Jan. 2024), 1176. doi:10.3390/app14031176
- [15] Kristóf Marussy, Attila Ficsor, Oszkár Semeráth, and Dániel Varró. 2024. Refinery: Graph Solver as a Service: Refinement-based Generation and Analysis of Consistent Models. In *Proc. IEEE/ACM Int. Conf. Softw. Eng. (ICSE), Companion Proc. (ICSE-Companion '24, Vol. 21)*. ACM, 64–68. doi:10.1145/3639478.3640045
- [16] Tom Arne Pedersen, Jon Arne Glomsrud, Else-Line Ruud, Aleksander Simonsen, Jarle Sandrib, and Bjørn-Olav Holtung Eriksen. 2020. Towards simulation-based verification of autonomous navigation systems. *Safety Science* 129 (2020), 104799. doi:10.1016/j.ssci.2020.104799
- [17] Tobias Rye Torben, Jon Arne Glomsrud, Tom Arne Pedersen, Ingrid B Utne, and Asgeir J Sørensen. 2022. Automatic simulation-based testing of autonomous ships using Gaussian processes and temporal logic. *Proc. Inst. Mech. Eng. Part O: J. Risk Reliab.* 237, 2 (2022), 293–313. doi:10.1177/1748006x211069277
- [18] Weiqiang Wang, Liwen Huang, Kezhong Liu, Yang Zhou, Zhitao Yuan, Xuri Xin, and Xiaolie Wu. 2024. Ship encounter scenario generation for collision avoidance algorithm testing based on AIS data. *Ocean Engineering* 291 (2024), 116436. doi:10.1016/j.oceaneng.2023.116436