Continuous integration for the development of a COLREG-compliant decision support system

Quentin Ageneau[a], Guillaume Nulac[a]

[a] *SIREHNA, 5 rue de l'Halbrane, 44340 Bouguenais, France*
* Corresponding Author. Email: quentin.ageneau@sirehna.com

Synopsis

The recent development in the field of autonomous navigation at sea is moving the focus from theoretical work and lab experiments to industrial solutions for the market. The move from the lab to the industry raises the question of the ability to deliver a consistent, dependable autonomous navigation capability to shipowners. One aspect is the hardware architecture, the dependability of which is traditionally ensured by applying long-standing, mature norms and guidelines (IEC61508, class regulations and guidelines). Another is the confidence that can be placed in software. The three questions are:
- how do the algorithms perform in terms of computational time, relevance of the proposed routes, repeatability, accuracy? This question is addressed with quantitative performance evaluations (see paper by Martelli et al 2024, Safenav session).
- have the algorithms been implemented correctly in the embedded software that will be deployed onboard? The answer is more of a Boolean type, and the related industry standard is software tests and Continuous Integration (CI).
- how does the system perform as a whole, once the software has been deployed and integrated with other soft- and hard-ware? This is addressed with Hardware-In-the-Loop (HIL) testing and the well-known industrial processes of Factory Acceptance Tests (FAT), Harbour Acceptance Tests (HAT) and Sea Acceptance Tests (SAT).
We have focused on the second aspect of software dependability, applied to a Decision Support System (DSS) whose role is to raise alerts and issue COLREG-based recommendations whenever a hazardous situation is identified at sea. We have implemented a Software-In-the-Loop (SIL) environment that can be used with CI tools to validate automatically that the recommendations and alerts issued by the DSS are consistent with the COLREG.
A major difference with simulations used for assessing the overall performance of algorithms (e.g., with Monte Carlo methods), is the need for exact repeatability: in a software-test context, developers need to be able to replay test scenarios exactly, to investigate why tests pass or fail. In this paper we present the test environment and discuss the constraints in terms of configuration management induced by the use of simulation in a CI context, including for the multi-agent simulator and the parameterization of test scenarios.

Keywords: COLREG, CI, continuous integration, test

1. Introduction: Developing industrial-grade collision avoidance decision support systems

Preventing ship collisions at sea has been a long-standing concern. The international rule for preventing accidents is the Collision Regulations (COLREGs) issued by the International Maritime Organization (IMO, 1972). However, COLREGs application is largely dependent on their application by human operators and their interpretation of maritime situations, vessels' manoeuvring capability, or other vessels' behaviour. In the recent record of maritime accidents, "human action" still accounts for sixty percent of all accidents (Martelli, 2023). Many attempts at improving navigation safety with the help of technology have been undertaken, but the interaction between man and safety systems sometimes had unexpected, disastrous consequences (see, e.g. Perrow, 1999). One of the main challenges lies in the difficulty for humans to have a clear understanding of the maritime situation around them; adding more sensors only adds to the difficulty.

In this context, the recent progress of computer vision and data fusion has opened new opportunities for maritime safety. Computers are now able to build a maritime situation with the help of the data provided by radars, but also cameras fitted onboard ship – relieving or complementing the human brain in this task. It is also very easy for a computer to infer an evasive route from a list of rules such as, for instance, the COLREGs.

Quentin Ageneau is a Software Architect in embedded systems specialized in unmanned surface vehicle at SIREHNA
Guillaume Nulac is a Systems Architect with years of experience in the domain of embedded maritime software and systems at SIREHNA.

 https://doi.org/10.24868/11155

The Safenav project intends to leverage these new opportunities to provide ship captains with a decision-support system (DSS) able to detect hazardous situations and make COLREGs-compliant recommendations. A major difficulty is to demonstrate that the system is able to provide relevant and reliable recommendations in all situations. Any flaw in the following aspects of system design could have catastrophic consequences in operation:

- Algorithms: ineffective, irrelevant or unreliable algorithms could mean that the maritime situation provided by the system is wrong, or the evasive manoeuvre recommended by the system is irrelevant. The system shall have firmly established algorithmic grounds, mathematically demonstrated and thoroughly tested in a wide number of cases.
- Software implementation: the implementation of the algorithms in the form of industrial software is a potential source of errors in itself. “Industrial software” has a lot more features than just the evasive manoeuvre algorithm, including authentication and cybersecurity, real-time calculation capability, log and memory management, obsolescence management, etc. Developing such software is an incremental, continuing process where each increment is a source of risk and regressions that shall be addressed with the greatest care. This paper explores this topic in further detail.
- System integration: eventually, software is deployed and integrated onboard. Conventional processes such as the well-known tryptic Factory / Harbour / Sea acceptance tests, are the typical way to demonstrate, step by step, that the system as a whole performs as expected.

Continuous Integration (CI) is the state-of-the-art approach to ensure that software is compliant with its requirements at all times during the development process - thus addressing the second bullet point. In this approach, software applications are *built and tested* as often as possible within the CI pipeline – not only at each release or new feature, but virtually every time a software developer has changed a line of code – in order to provide instantaneous feedback on any error in the code. Continuous Integration is made possible by *automated* testing. An automated test is a scenario that can be run automatically on the freshly built software application, the outputs of which can be automatically evaluated with respect to a pre-defined reference. In order for the developers to make the most of a CI pipeline, the tests shall be reasonably fast and repeatable (to enable debugging whenever a test fails).

In the context of a COLREGs-compliant DSS, each test involves a maritime situation with all its complexity: the ship that is fitted with the DSS (referred to as “Own ship”) navigates in the vicinity of other ships that may have their own decision capability (see, e.g., Pedersen et al, 2023). The Own ship navigates either by following strictly the path recommended by the DSS, or with a delay (if the captain waits to apply recommendations), or without any consideration for the recommendation. Any action taken by the Own ship potentially results in a different end situation, depending on the reactions of the other ships in the simulation: the complexity of the simulation is driven not only by the DSS of the Own ship, but also by feedback loops involving the other ships.

The paper describes how to validate a COLREGs-compliant DSS implementation in a CI. The software in the loop simulator used to test the DSS will be described in section 2. Section 3 provides a description of a Continuous Integration pipeline that uses the simulator described in section 2 to run and validate the test scenarios also described in section 2. The advantages and limitations of the approach are discussed in section 4.

2. Software in the loop simulation of a Decision Support System

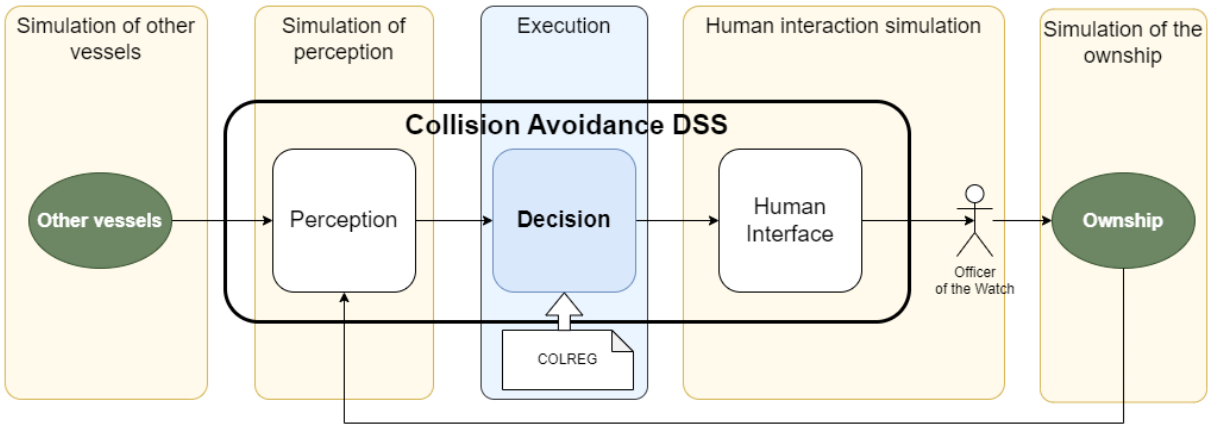

Figure 1 – Overall simulation context

 https://doi.org/10.24868/11155

Figure 1 illustrates the simulation environment used to validate a collision avoidance decision support system, and more specifically the Decision support module wherein.

The collision avoidance decision support system is made of three main modules:

- Perception: this part of the system includes the sensors and algorithm that works together in order to build a consistent and relevant knowledge of the maritime situation and its environment,
- Decision: based on the perception of the ship situation and the ships' objectives, this component elaborates recommendations including path planning and avoidance manoeuvres,
- Human interface: unlike fully automated navigation systems, the quality of interactions between machines and human is essential to guarantee the effectiveness of the decision support system. If the interface is liable to misinterpretations, delayed or flawed recommendation display, the evasive manoeuvres may not be implemented as recommended by the Decision module.

In order to test a DSS (and more generally a complex software) in a CI, a specific "wrapper" need to be developed. The Safenav's SIL platform was developed to do so. It is composed of two main components:

- Maritime situation simulator: Runs the simulation and provides updated inputs to the DSS
- Validation Module: Validate the behavior of the DSS

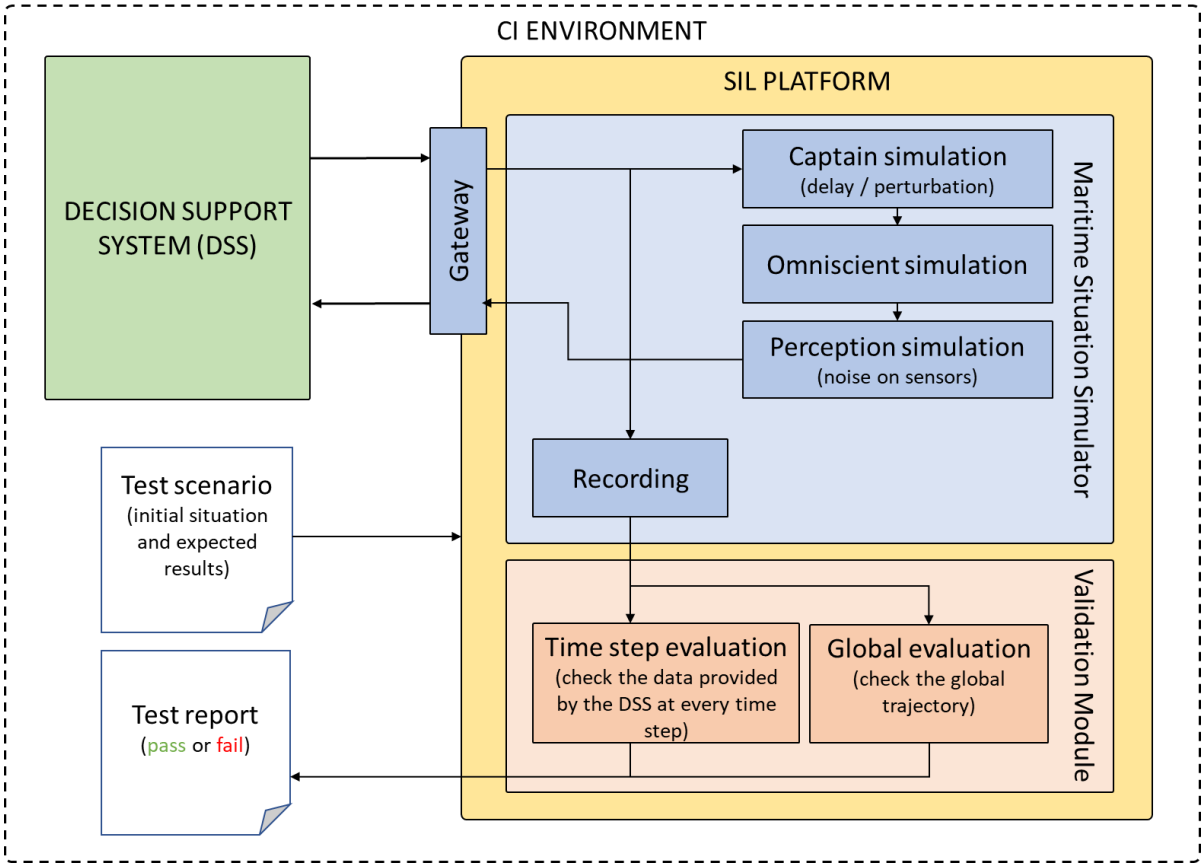

Figure 2 Functional block diagram of the simulator of the Safenav DSS

2.1. Test scenarios

In the context described in this paper, the CI is used to ensure that changes in the source code do not induce functional regressions. The absence of a regression is verified by running a Test case, defined as:

- An initial Maritime situation,
- A behavioural model of the Own ship and of the surrounding ships (including the reaction time of the Officer of the Watch to be considered in this particular Test case),
- A time limit for the simulation,
- A validation criterion to be evaluated with respect to the simulated Maritime situation record.

 https://doi.org/10.24868/11155

For the sake of simplicity, environmental conditions have been neglected and the vessels are assumed to evolve in a perfect, open sea environment devoid of navigation channels or signals and mapped with Cartesian coordinates.

The Test cases shall be representative of actual hazardous situations encountered at sea and they shall cover as wide a spectrum of maritime situations as possible. The 55 "situations" proposed by Pedersen et al, 2023, complemented with a set of 12 scenarios devised by the University of Rijeka for the Safenav project, have been used to generate a basic set of Test cases. How to extend this set, especially by including limit cases unforeseen a priori, will be discussed in section 4.

In what follows, the surrounding ships have been assigned a trivial behavioural model – each vessel follows a straight route at constant speed regardless of the surrounding ships, including the Own ship.

As a general rule for devising the set of test scenarios, only those scenarios for which a COLREG-compliant route is known to exist have been selected: scenarios where the collision risk is detected when the Own ship is already within the CPA of the approaching vessel, for instance, have been discarded.

The total number of scenarios is on the order of a few dozens to a few hundred. With the simulator described in this section, it takes in the order of seconds to run each scenario on a standard computer, with a total testing phase in the CI pipeline that takes in the order of a few minutes. This is a reasonable time for a developer to receive a relevant feedback.

2.2. Maritime situation simulator

In this paper, a Maritime situation is defined by the positions, speeds and headings of several ships sailing in an area of interest. It is the role of the Maritime situation simulator to infer the Maritime situation record based on the initial situation and the characteristics of the ships (behaviour depending on their environment, manoeuvring capability, etc.).

The position and speed of the vessels including the Own ship are evaluated with time simulation. A perfect Perception assumption has been taken: the Decision module knows the Maritime situation exactly and with no time delay. In a further stage, sensors may be simulated to be more representative of the actual system.

The Human interface has been modelled by introducing a delay in the application of the recommended manoeuvres. Similarly, this first approach can be extended and enriched in further work, as mentioned in section 4.

The recommended route issued by the Decision module is sent to the Maritime simulation simulator that converts it into a new route increment for the Own ship – adding a time delay accounting for the reaction time of the Officer of the Watch if need be. This is done at every time step: the Own ship's route is the result of the consecutive updates received from the DSS, and does not necessarily coincides with any of the individual routes recommended by the Decision module.

After the position and speed of the Own ship have been updated, the position and speed of all the other ships is computed. This is done by taking into account the manoeuvring and behavioural characteristics of every ship, including their autonomous decisional capability. The position and speed of all the ships, including the Own ship and the other vessels, is then fed back into the Decision module for the next time step.

At the end of the simulation (specified as a predefined time limit), the entire Maritime situation record is stored for further use by the validation module (see section 2.3).

2.3. Validation module: COLREGs considered here and automated criteria

The goal of this paper is to verify that the navigation pattern followed by the Own ship is consistent with COLREGs. This is done by analysing the Maritime situation record. The simulator infers the Maritime situation record from:

 https://doi.org/10.24868/11155

- The initial Maritime situation
- The actions of the Officer of the Watch, based on the recommendations from the DSS,
- The manoeuvring characteristics of the Own ship,
- The behavioural and manoeuvring characteristics of the other vessels.

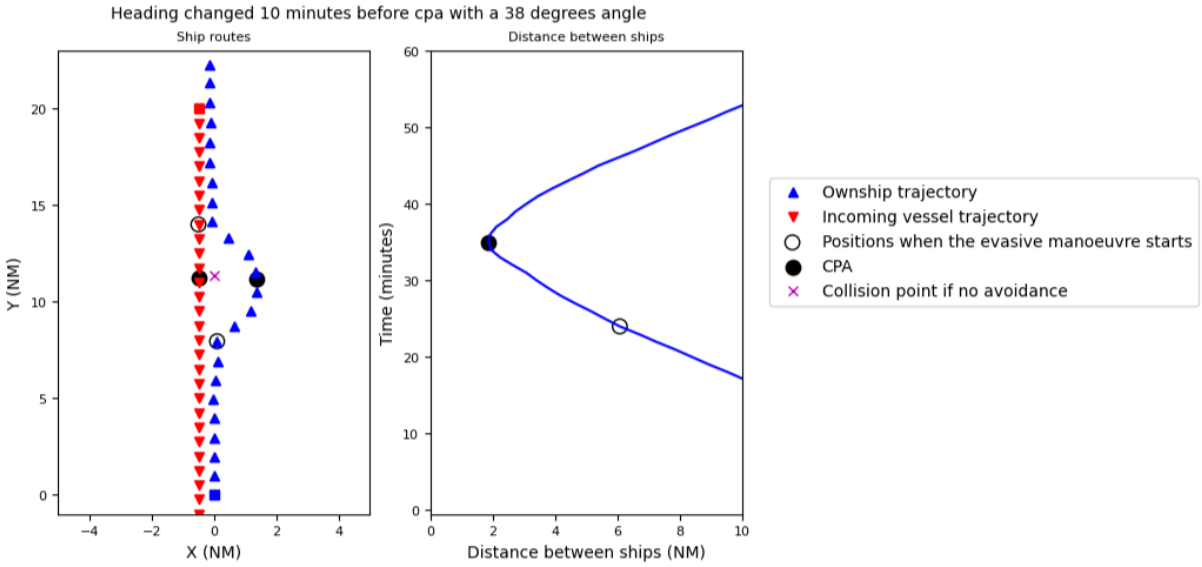

Figure 3 – Maritime situation record, as generated by the Maritime situation simulator.

The spirit of the COLREGs is not to simply prevent collisions, but to anticipate on potential collisions to come. Thus, it is not enough to check for the absence of a collision to declare an evasive manoeuvre COLREG-compliant.

"Rule 14 - Head-on situation" is one such navigation rule, and it will be used to explain the automation process in the sections that follow – together with Rule 8, that provides requirements on how the evasive manoeuvre shall be implemented.

Figure 4 to 7 illustrates how COLREGs are validated against several Maritime situation records corresponding to a "head-on" encounter between the Own ship and an incoming vessel. Each case displayed corresponds to a different behaviour of the Own ship. The following COLREG requirements (and the violation thereof) are captured:

- The route lies in the half-plane located starboard from the initial route of the ship: this criterion validates rule 14, that states that the vessel shall alter her course to starboard
- The route deviates from the original one more than 10 minutes before tCPA. This criterion validates Rule 8(a), according to which the ship shall alter her course "ample time" before collision
- after the Own ship has altered her course, the heading is 30 degrees starboard from the original route. This criterion validates Rule 8(b), according to which the evasive manoeuvre shall be sharp.
- The distance between the recommended route and the approaching vessel is above CPA at all times. Here we have chosen CPA = 0.6 nautical miles.

 https://doi.org/10.24868/11155

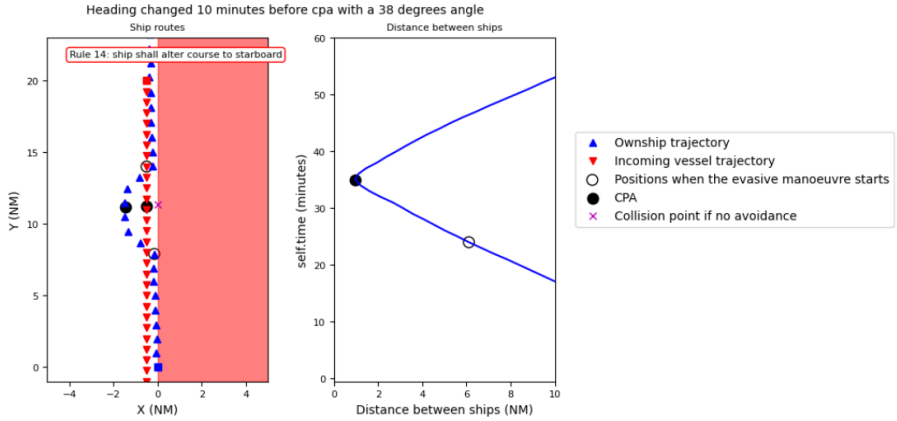

Figure 4 – Maritime situation records: The collision is avoided, but the Own ship crosses the route of the incoming vessel then leaves her on starboard.

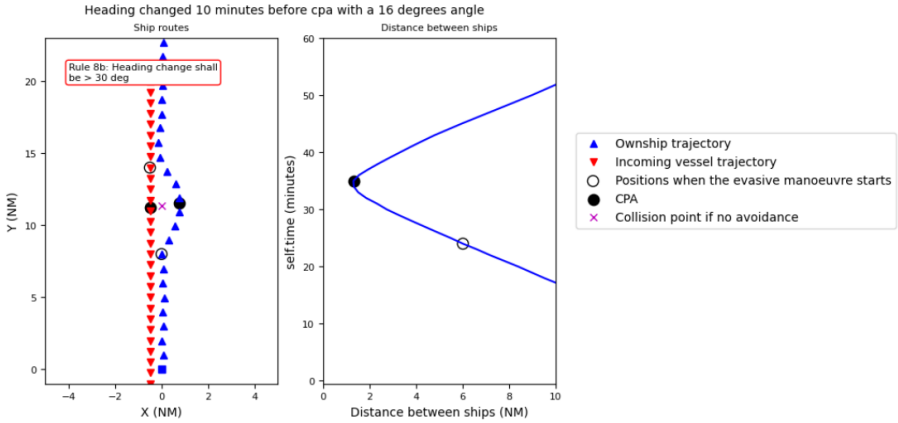

Figure 5 – Maritime situation records: The evasive manoeuvre is not sharp enough.

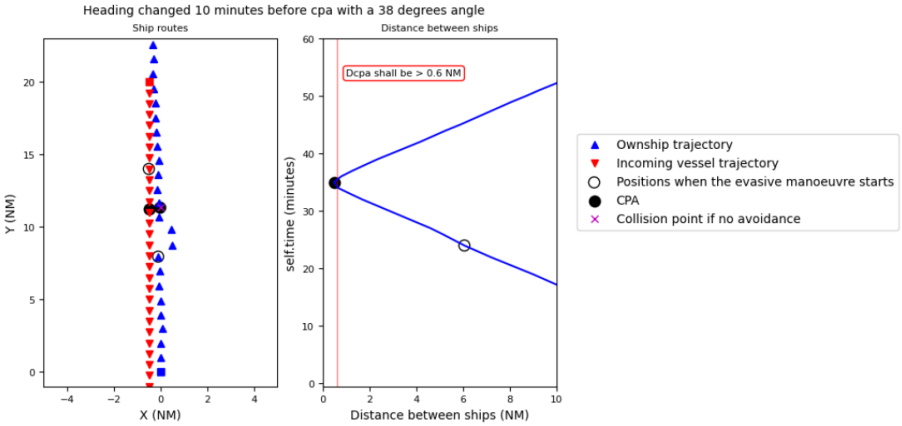

Figure 6 – Maritime situation records: The evasive manoeuvre fails to keep the vessels at a safe distance.

 https://doi.org/10.24868/11155

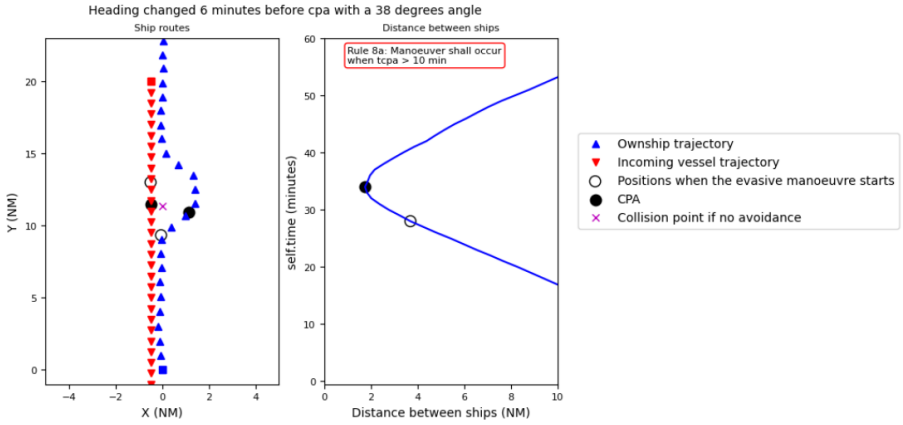

Figure 7 – Maritime situation records: The evasive manoeuvre was initiated too late.

The validation procedure implemented here has two important properties:

- It is entirely automated: there is no need of a human eye to assess the compliance of the routes to Rule 14,
- Any kind of route followed by the Own ship can be assessed, including routes that are the result of incremental updates (as would be the case if the Officer of the Watch had adjusted to several recommendations sent to her at different times).

3. Implementation of a continuous integration environment

3.1. CI methodology applied to a collision avoidance DSS

Continuous Integration (CI) is a method of software development, where code changes are continuously built and tested: the delay between the writing of the code and its validation with tests is reduced to a minimum (on the order of a few seconds to a few minutes). CI is commonly paired with Continuous Deployment (CD) to automatically deploy, and monitor changes (outside the scope of this paper). The aim of CI is to identify bugs or regressions early in the development cycle and reduce the chance that new code was developed on a buggy previous version.

In the context of the DSS development in the Safenav project, the CI is designed for all the lifecycle phases of the DSS system and not only during the initial development period. This includes:

- The initial development phase: this is especially useful during this early development phase because the software code changes a lot. However, tests must be written sparingly because requirements and specifications may also change a lot during this period, which would render many tests obsolete. A good balance between tests quantity and tests durability must be found.
- The validation phases and integration with other systems: during this period, slight adjustments may be required and fast delivered due to interfaces issues with other systems. This is very important to avoid regression during this period that may involve a lot of resources (human and hardware platforms).
- The production phase: if a bug is detected on board a ship during the effective use of DSS, it is critical to avoid any regression in the modified software that may be deployed on many platforms. At this stage of the project, the amount of tests shall cover the whole DSS functions and the maximum possible situation that may be encounter by the system in real life

 https://doi.org/10.24868/11155

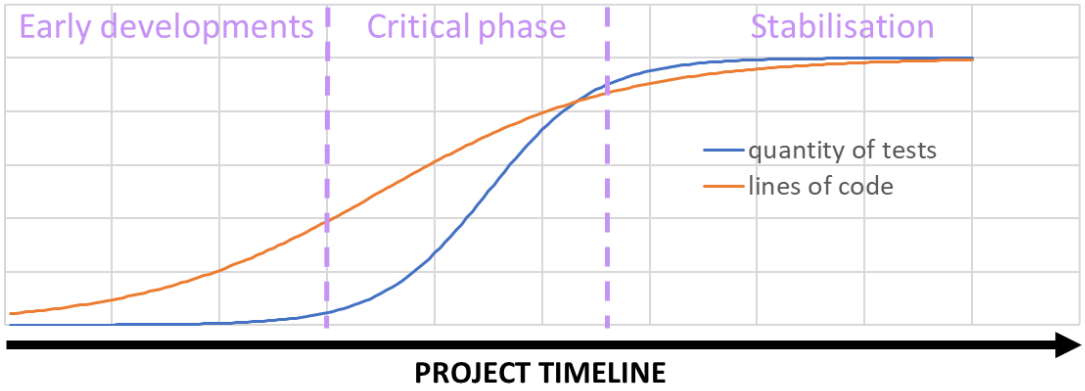

Figure 8 – Evolution of lines of code and quantity of CI tests over time in a typical software project

The first step before configuring a CI environment is to think about what is expected from the CI and what are the important requirements that shall be addressed by the CI. This will depend on:

- Organisation of the software code management
- Identification of the feared events that we must mitigate through the CI process
- Transverse requirements about the performance of tests and their hierarchy

The Gitlab SCM host at least 3 repositories for the development of the DSS:

- The DSS itself, including each submodule that may be provided by third parties
- The SIL platform simulator and associated files
- A repository containing configuration files, including the scenario used for the tests

During the DSS lifecycle, many events may require a code modification:

- New features requested by customer (or any people involved in the project)
- Refactoring of code required for engineering reason
- Bug or incident detected, during any phase of the DSS lifecycle

Every code modification can lead to regression of the existing functions of the DSS and this can lead to more or less critical consequences for the projects:

- loss of time during development phase
- critical injury of people using DSS on board a ship
- any other intermediate consequences

Evaluation of a DSS using complex simulations in the context of a Continuous Integration process is challenging with respect to the three qualities of an efficient CI:

- Automated evaluation: COLREGs have been devised to guide ship captains, yet leave them some freedom of action to prevent collisions in the best ways they see fit. COLREGs leave some room for interpretation, which is not easy to consider in automated tests (see, e.g. Pedersen et al, 2023).
- Reasonably fast tests: a subset of maritime scenarios, simulated in a reasonably coarse fashion, shall be established to ensure a good testing coverage within a controlled run time.
- Repeatability: in order for the developers to be able to debug the DSS efficiently, not only the source code of the DSS, but also the entire test environment, testing parameters and test outputs shall be versioned. This places a constraint on the development and execution environments of the simulator.

3.2. Implementation of the CI pipeline

The implementation of a CI pipeline using a Gitlab environment is chosen for this paper. This choice is motivated by the extensive capacities of the Gitlab open source version (Community Edition, under MIT licence).

A pipeline is a sequential list of tasks that can be triggered by various events, like source code increments (commits or merges), or on a predefined schedule. The tasks defined for the SafeNav DSS are the following:

 https://doi.org/10.24868/11155

Stage	Task	Description
1 - Check format		Use Linters and Code Formatters that analyse source code to detect potential errors, security vulnerabilities, or coding style issues.
2 - Build		Clone project, fetch dependencies, compile and link
3 - Tests	Unit tests	Execution of the unit tests of the DSS
	External interfaces tests	Each external interface is tested using a predefined dataset ensuring that each field of the tested interfaces is consistent with the content of the scenario. This step ensures an easy integration phase afterwards
	Functional scenarios tests	The aim of this step is to validate the main functionalities of the DSS, in particular the relevance of the DSS recommendations and the correctness of the indicators sent to the GUI. Those tests are carried out using the DSS as a black-box. The SIL platform is used to emulate a complete maritime situation that will serve as a playground for the DSS. The simulation is initialized using a test scenario and evolves as the DSS provides recommendations. The success or failure of each scenario is evaluated directly after the simulation.
4 - Deploy		Package and store on a repository

The CI pipeline is completely automated, is automatically triggered, and it can run in the background, on a separate machine: it makes software testing extremely easy and harmless to the software developer, who can proceed with the development and have his code tested along the way. Rather than scheduling a few tedious, a posteriori testing phases – inevitably followed by debugging work on code that has been implemented a few days or a few weeks before – developers have numerous, instantaneous feedbacks on code they just wrote and can fix while everything is still fresh in their minds.

3.3. Output of a pipeline and debugging

The Test stage of the CI is configured to generate a test report composed for each test case of:

- a Boolean output “Pass or Fail”,
- in case of failure, a job artifact with all necessary elements to replay the job.

Suite	Test Case	Status	Time (s)	Error Message
test_maritime_scenarios.py	test_ID100_gui_indicators[11_responsibilities_between_vessels.yaml]	PASS	0.0	
test_maritime_scenarios.py	test_ID100_gui_indicators[12_heavy_marine_traffic.yaml]	PASS	0.0	
test_maritime_scenarios.py	test_ID200_colreg_detection[01_head-on.yaml]	FAIL	41.92	AssertionError: Colregs [7, 14] not cle (...)
test_maritime_scenarios.py	test_ID200_colreg_detection[02_crossing_from_starboard_side.yaml]	FAIL	48.09	AssertionError: Colregs [7, 15, 16, 17] (...)
test_maritime_scenarios.py	test_ID200_colreg_detection[03_crossing_from_port_side.yaml]	PASS	1.86	
test_maritime_scenarios.py	test_ID200_colreg_detection[04_overtaking.yaml]	PASS	1.88	
test_maritime_scenarios.py	test_ID200_colreg_detection[05_head-on_or_crossing_dilemma.yaml]	PASS	2.15	
test_maritime_scenarios.py	test_ID200_colreg_detection[06_stand_on_or_appropriate_action_dilemma.yaml]	PASS	2.22	
test_maritime_scenarios.py	test_ID200_colreg_detection[07_crossing_from_starboard_or_overtaking_action.yaml]	FAIL	61.3	AssertionError: Colregs [7, 15, 16, 17] (...)
test_maritime_scenarios.py	test_ID200_colreg_detection[08_crossing_from_port_side_and_overtaking_action.yaml]	PASS	2.41	

Showing 11 to 20 of 36 entries Previous 1 2 3 4 Next

Figure 9 - HTML test report: Summary of the test campaign results with a pass/fail status

 https://doi.org/10.24868/11155

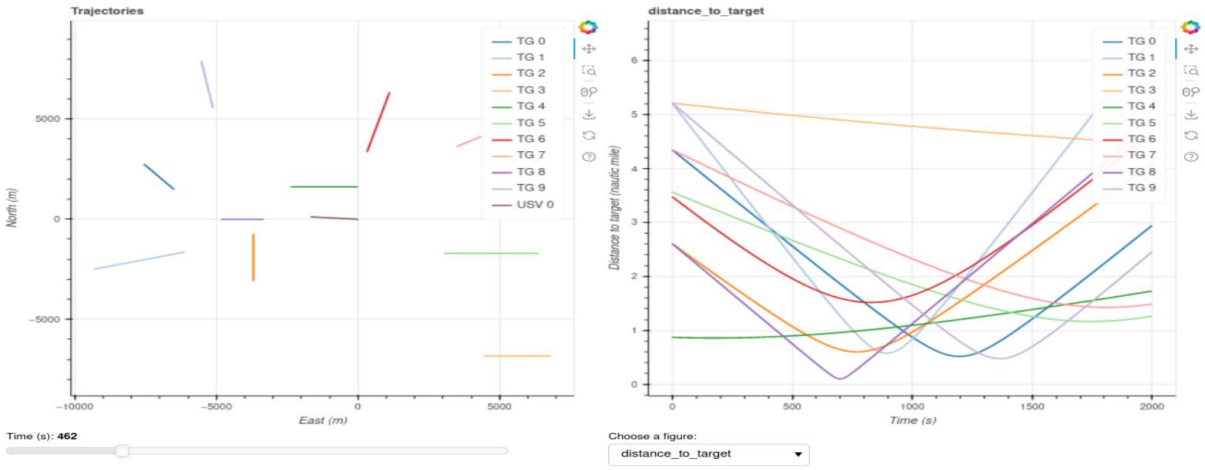

Figure 10 - HTML test report: Indicators and graphs available for each test

The interactive HTML test report makes understanding KOs much simpler. Analysis of indicators coupled with the ability to replay a scenario under the same conditions allows fails to be solve in a short loop by the developer.

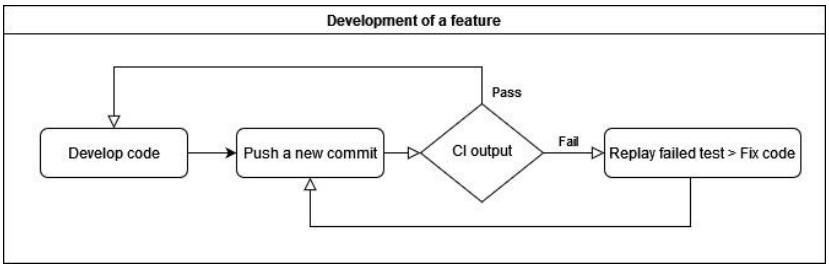

Figure 11 – Development workflow using Continuous Integration. As long as the code passes the tests, development can proceed. Whenever a new code increment fails, the developer is invited to investigate the failed test immediately, based on the artefacts logged by the CI.

4. Discussion

A fully automated validation of a COLREG-compliant DSS has been shown to enable the use of Continuous Integration to improve the robustness of software development. The CI pipeline runs a set of predefined Test cases that are intended to cover the entire range of Maritime situations that may be encountered at sea.

A common issue with complex systems is the vast diversity of situations that can be encountered. A slight deviation in the initial conditions (e.g., speed of encounter vessel) may generate widely different outputs. Another source of diversity is when the Maritime situation representativeness will be increased, either by including external elements such as environmental conditions, coastlines or navigation channels, or by enriching the model of the Own ship (taking into account sensors accuracy, human factor, etc.). By definition, the CI environment we have set up addresses only a limited number of scenarios.

This first set of scenarios could be complemented with scenarios generated at random or extracted from AIS traffic databases. Every time a failed scenario has been found, if the scenario is relevant, a new Test case shall be added to the existing set run by the CI.
It is expected that the number of failed scenarios among those generated at random should decrease over time, while test coverage naturally increases in the process.

Being able to easily visualize the output of the DSS is another challenge that need to be addressed. Generating 3D views of the simulation will be beneficial for debugging and it also brings the opportunity to add a perception

 https://doi.org/10.24868/11155

module into the testing loop. We even can imagine simulating a view of the ownship's bridge as it was during the simulation to make a sea captain challenge the DSS decisions.

A drawback of the discussed improvements is the increasing computational time that may render the CI pipeline ineffective for non-regression verification purposes. One option is to have a two-stage CI:

- During daytime and for non-regression validation purposes: test a limited number of predefined scenarios,
- Nightly: generate multiple scenarios and whenever one of them fails the validation criteria, log the results of the simulation for further analysis

Acknowledgements

This research was partially funded by European Union's Horizon Europe under the call HORIZON-CL5-2022-D6-01 (Safe, Resilient Transport and Smart Mobility services for passengers and goods), grant number 101077026, project name SafeNav. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or Executive Agency (CINEA). Neither the European Union nor the granting authority can be held responsible for them.

References

Martelli M., Žuškin S., Zaccone R. & Rudan I. A COLREGs-Compliant Decision Support Tool to Prevent Collisions at Sea, in TransNav Volume 17, Number 2, June 2023

Huang, Y., Chen, L., Chen, P., Negenborn, R. R., & van Gelder, P. H. A. J. M. (2020): Ship collision avoidance methods: State-of-the-art. Safety Science, 121, 451-473.

International Maritime Organization, 1972. International Regulations for Preventing Collisions at Sea (Consolidated edition, 2018). Archived on 03-March-2023. Retrieved from https://www.samgongustofa.is/media/log-og-reglur/COLREG-Consolidated-2018.pdf

PERROW, C. Normal Accidents: Living with High Risk Technologies - Updated Edition (REV-Revised). Princeton University Press, 1999.

Tom Arne Pedersen et al 2023 J. Phys.: Conf. Ser. 2618 012013