Mälardalen University
School of Innovation Design and Engineering
Västerås, Sweden

Thesis for the Degree of Master of Science in Computer Science with Specialization in Software Engineering 30.0 credits

AUTOMATIC PERFORMANCE TESTING OF MARITIME SIMULATION MODELS

Shaheryar Malik
ssr22007@student.mdh.se

Examiner: Eduard Paul Enoiu
Mälardalen University, Västerås, Sweden

Supervisors: Wasif Afzal
Mälardalen University, Västerås, Sweden

Company supervisors: Sebastian Lafond, Ivan Porres Paltor
Åbo Akademi University, Turku, Finland

November 14, 2023

Abstract

Simulation models are essential tools in the maritime industry for predicting ship behavior and interactions in changing marine environments. This report explores the importance of these models and outlines how they might improve ship operations, design, and safety. It highlights the need for ongoing updates to guarantee their dependability. The performance of vessels must be improved while emissions and expenses are reduced since maritime transportation is essential for global trade. Simulation models provide useful insights for this project.

This thesis provides an intelligent way for automating testing of marine simulation models in the Open Simulation Platform (OSP) environment by utilizing deep learning techniques. With the help of artificial intelligence and computer algorithms, this methodology aims to increase testing accuracy while lowering costs. Customized tests for simulation models are developed using generative adversarial networks (GANs), which mimic real-world behaviors. Although the computational requirements present difficulties, the accuracy and dependability shown by GANs surpass those of traditional techniques.

This thesis emphasizes the usefulness of deep learning in maritime software testing, improving performance evaluation through careful investigation. Extending the variety of scenarios and improving the test creation procedures are two future directions. In the end, this project helps the marine industry advance toward improved effectiveness, environmental awareness, and safety. The conclusions drawn from the paper highlight the good potential of cutting-edge technology in creating a future for maritime transportation that is sustainable.

Table of Contents

1. Introduction

Simulation models are essential in the maritime industry because they play a important part in projecting and predicting ship behavior and its interactions with the dynamic marine environment. By putting efficiency and safety first, engineers and designers can optimize ship operations and designs. They provide priceless information about how ships work in various scenarios. The accuracy and dependability of these simulation models must be maintained as a top priority because the results have a direct impact on the operation and safety of seagoing vessels. As a result, ongoing review and modifications are necessary to preserve the relevance, accuracy, and reliability of these models.

It is impossible to exaggerate the importance of marine transportation for global trade since it connects countries and promotes economic growth [1]. It is now more important than ever to optimize vessel performance and reduce operational expenses due to the rising demand for maritime services. By simulating different elements including propulsion systems, fuel consumption, and power generation, maritime simulation models intervene to tackle these problems. By analyzing ship performance in a variety of circumstances, they give shipowners and operators crucial information about how to improve operations, save fuel use, and mitigate environmental concerns.

Co-simulation has developed into a successful tool for thoroughly assessing the complex systems that distinguish modern ships. By combining several simulation models into a single framework, it provides a thorough evaluation of how different ship systems interact with one another. The total functioning of the vessel is better understood thanks to this holistic approach. This thesis heavily relies on the Open Simulation Platform (OSP) [2], an open-source co-simulation platform that is essential to the marine industry. OSP makes it easier to combine and assess different maritime simulation models, providing designers, engineers, and researchers with a stable platform to work together on improving ship performance.

Maritime simulation models can identify possible ship design and operating improvement areas through thorough testing and review within OSP. They are essential in guiding the sector toward more eco-friendly practices, with a heavy emphasis on improving fuel efficiency, lowering emissions, and protecting the safety of seafarers. They also provide helpful information that supports the creation of greener, more ecologically friendly ship designs, supporting the global push for sustainability and conservation in the marine industry.

The promise of automated performance testing looms large on the horizon as technology is developing quickly. The accuracy and dependability of marine simulation models could be greatly improved by combining computer algorithms, artificial intelligence, and machine learning approaches. By using huge databases of simulated scenarios, automatic performance testing enables researchers and developers to predict model performance. This innovative approach not only enhances the precision of simulation results but also reduces the costs associated with numerous maritime operations.

With the help of the already-available tool "Farn," [3] the Open Simulation Platform (OSP) has proven to significantly speed up simulation procedures. By utilizing these models, Farn functions as a useful OSP (co-)simulation file generator adapted to certain scenarios. This makes it possible to precisely control a number of simulation-related aspects, such as shell commands, sampling techniques, the creation of case folders, the duplication of template files, the creation of parameter files, and simulation execution. Farn enhances the testing and assessment of maritime simulation models through parameterization and simulation scenario execution. While Farn has demonstrated its usefulness, this thesis embarks on a creative exploration and looks into new possibilities for creativity. Exploring the possibilities of deep learning models in the context of creating test cases for simulation models is the target of this work.

The main goal of this thesis is to use the Open Simulation Platform (OSP) and System Testing Using Generative Models (STGEM) [4] to automatically test the performance of marine simulation models within the OSP. In order to build test cases that accurately represent numerous real-world circumstances and enable a thorough assessment of the performance of maritime simulation models, this work intends to leverage the potential of generative models. The goal of the thesis is to locate any potential errors or inaccuracies in the simulation models by utilizing deep learning techniques and examining the created test cases. This thesis project hopes to push the limits of conventional simulation testing and open the door for more advanced and trustworthy performance evaluation

approaches by utilizing STGEM and OSP.

In summary, this ambitious thesis aims to significantly test marine simulation models through generative models. This work aims to improve the reliability and accuracy of test cases in the OSP while leading the adoption of sustainable practices in the maritime domain by utilizing the full capacity of existing tools like STGEM, OSP, and Farn. This research journey carries the promise of creating a brighter and more environmentally friendly future for the oceans, and it has the ability to change the maritime industry.

2. Background

In this section, we shall examine several methods and equipment utilized in the fields of digital twins and simulation by the maritime industry. We'll start by talking about the idea of general digital twins and their importance in many businesses. We'll go into detail about co-simulation, another method employed in digital twin development. The Open Simulation Platform (OSP), an open-source program that enables the integration of numerous simulators, will be the next topic we discuss. In addition, we will go over the specific OSP use case that we will be using for our work. OSP provides a number of demo reference models with which we can work, each of which performs a particular simulation task. We will use the Gunnerus-DP model, which will be detailed further below in the OSP section. Furthermore, the creation of test cases will be covered as a general strategy for evaluating digital twins and simulations. Finally, we will discuss two specific tools that have been created for the purpose of test case generation: STGEM and FARN. By the end of this section, readers will gain a comprehensive understanding of the digital twins and simulation along with other relevant tools that have been used in this thesis.

2.1. Importance of Maritime Industry

The majority of world trade, both in terms of volume and value, is carried out by the maritime sector, which is essential to international trade. Despite this, the sector has lagged behind other industries in adopting digitization. Recent literature [5] reviews have uncovered a number of untapped fields that want immediate attention, including robots, autonomous cars, and digital security. The sector needs to invest in digitalization and catch up with other industries if it is to stay competitive and promote sustainable mobility. To ensure the industry's sustained success in enabling international trade, which makes it an essential part of the global economy, more research and funding in this sector are very necessary. In the current worldwide economic globalization there must be an agile, safe, and reliable maritime transport industry to support future economic growth which can be improved by embracing new technologies such as: the internet of things (IoT), cloud and edge computing, digital twins, simulation, and machine learning [5].

The world's seaports are crucial to the global economy because container traffic has increased by 10% annually since 1990 and larger ship sizes are creating logistical and technological challenges [6]. Big Data and digital technologies have a lot to offer the shipping and maritime logistics sectors, including increased productivity, safety, and energy savings. Digitalization does, however, raise some issues, including data misuse and cybercrime. The importance of utilizing digitization's development potential in order to benefit from it is demonstrated by a detailed examination in [6] . Even though the research is still in its early stages, theoretical and empirical studies must be looked into more in order to offer suggestions for action and restructuring in the maritime industry.

Numerous parties frequently contribute to the expansion of transport capacity in the maritime sector. Such expansion carries the danger of disruptions, though, which could result in significant disruptions to the global supply chain. For instance, the noteworthy incident in March 2021 in which the cargo ship Ever Given blocked the Suez Canal resulted in a significant disturbance and momentarily prevented more than 400 vessels from transiting the important canal [7]. Even though the canal eventually reopened, the incident's aftermath showed a new problem: a complete inability to handle the sudden increase in demand. This led to significant delays in transportation activities. This hypothetical situation serves as a stark reminder of the maritime industry's complexity and interconnection, wherein disruptions at a single point in the supply chain can cascade and have a significant impact on the entire system, necessitating strategic planning and risk management in order to maintain a reliable and effective global transportation network.

Using panel data from 2007 to 2018, Fratila et al. [1] did a study to look at the connection between maritime transport and economic growth in EU nations. They used a panel regression model to estimate the findings, which showed that financial support for maritime infrastructure has a beneficial effect on economic expansion. The report did note, however, that environmental problems also stand in the way of the maritime industry's expansion. These results highlight the significance of sustainable development and the necessity of tackling environmental issues while promoting maritime industry economic growth.

It is impossible to overstate the importance of the marine industry, which is essential to both

international trade and the expansion of the world economy. Despite playing a crucial role, the industry has been slower than other industries to adopt digitization. Recent literature studies have, however, highlighted under-researched areas that demand immediate attention, including robotics, autonomous vehicles, and digital security. The marine industry needs to invest in digitization and adopt IoT, cloud computing, and machine learning technologies if it wants to maintain its competitiveness and promote sustainable mobility. The significance of seaports in the world economy is also evident, as seen by the steadily rising volume of containers. Digital technologies have a tremendous deal of potential to boost productivity, improve safety, and save energy, but they also raise questions about data misuse and cybercrime. However, the marine sector may take advantage of digitization's development potential and apply restructuring measures to assure its ongoing success by investigating theoretical and empirical studies.

The marine industry has a huge impact on trade and economic development worldwide. Although it is still crucial to global trade, the business has been slower than other industries to adopt digital technologies. A few mentioned recent literature reviews have uncovered a number of uncharted territories that demand rapid attention, including robotics, autonomous vehicles, and digital security. Adopting digitization is essential for promoting sustainable mobility and the competitiveness of the maritime sector. The industry must spend in research, finance, and digitalization to stay up with technology improvements if it is to prosper in the global economy. Seaports also have a significant impact on international trade due to the constant increase in container traffic. The maritime industry can increase productivity, safety, and energy efficiency by utilizing big data and digital technologies.

2.2. General Digital Twins

A digital twin is typically described to consist out of three parts: a physical entity, a digital counterpart, and the data connections in between. One of the key use cases is to enable leveraging of computation through the digital entity with the goal of improving its physical counterpart. The concept of the digital twin was first introduced by Michael Grieves and John Vickers of NASA in 2003. At that time, virtual product representations were not well developed and data about physical products was often collected manually on paper. Grieves and Vickers believed that a digital model of a product could serve as the basis for product life-cycle management [8].

The term digital twin saw its popularity significantly rise in the mid 2010s and in 2016 early academic publications were published on the topic of digital twin ships. These digital twin ship implementations found in the literature can generally be grouped into two main categories. The first one being decision support for ship operations, with the focus on condition monitoring and calibration of simulation models based on real operational data [9].

Coraddu et al. in 2019 estimate speed loss caused by marine fouling using a neural network based simulation model. The network receives data collected from a physical vessel and returns an estimation of speed loss. This method demonstrated better results compared to the ISO standard for estimating fouling but with the drawback of needing large amounts of data due to the nature of its machine learning approach [10]. Furthermore, Schirmann et al. in 2018 present a digital twin for ship motion and estimation of structural fatigue caused by waves. It uses weather forecast data for a specified route and the digital twin estimates the damage its physical counterpart would take during its trip [11].

2.3. Co-simulation

A dynamical system is a model of some real system, it could be a physical system or a computer system. The system is characterized by a state and a set of evolution rules which modifies set state. The evolution rules can be seen as a set of functions. A simple example of a dynamic system are traffic lights. We can establish four distinct states: green light, red light, yellow light, and off. An evolution rule could be that after a set amount of time in the green light state it will switch to the yellow light state. This is described as simulation [12]. What differs simulation from a static model like for example a blueprint or a 3D model of a vessel used in the design and development stages of construction is the addition of the time dimension. That is the model will have a different state for each discrete time step in the simulation.

With the increase in sensors, software, and networking in ships poses challenges in design, operation, and maintenance. The design process has to be made more distributed as responsibility has to be divided among several teams with different expertise or external providers. Each party develops a partial solution to a system that needs to be integrated with all the other partial solutions [12].

Modelling and simulation is an important part to improving the development but this distributed environment poses challenges when it comes to exchanging and integrating the resulting partial solutions. This could be for example because of potential differences in tools used or worries about intellectual property which cannot be easily disclosed to system integrators. Co-simulation is to enable simulation of a coupled system via the composition of sub simulators. Performing global simulation of a coupled system by composing the simulation of its parts. Each simulator is broadly defined as a black box capable of executing some behavior, consuming inputs, and producing outputs [12].

To interact with or operate a complex system, training is often necessary. In some cases, this training must be conducted in a virtual environment for reasons such as safety or cost. Creating a virtual environment for this purpose can be a challenging and expensive task, but reusing models that were used in the development of the system can help focus efforts on the most important areas. As previously stated, it may be difficult to obtain a single model of the whole system. Additionally, a high-fidelity model of the system can be used for maintenance purposes. By using advanced sensory information collected during normal system operation, it can be used in a simulator to predict and prevent faults [12]. As systems become increasingly intricate, there will be a growing demand for co-simulation scenarios that are extensive, structured in a hierarchical manner, diverse in components, precise, and safeguarded by intellectual property rights.

The idea of digital twins, highly precise models that faithfully replicate real-world objects, offers promising prospects for the future of advanced ships. Implementing digital twins in the maritime sector facilitates data analysis and system monitoring, enabling proactive issue prevention and strategic planning through simulations. Nevertheless, there are multiple obstacles that need to be surmounted to turn this concept into a tangible reality. Two hurdles that need to be tackled involve integrating diverse systems and hardware, which poses difficulties in developing a centralized or monolithic digital twin. Additionally, the memory and CPU usage necessary for running these simulations present another challenge. To overcome these issues, co-simulation emerges as a viable solution. By independently modeling different subsystems and then simulating them collectively, co-simulation optimizes resource utilization, making the process more efficient and effective [13].

2.4. The Open Simulation Platform (OSP)

The Open Simulation Platform (OSP) serves as a pioneering open-source industry initiative, creating the co-simulation of maritime equipment, systems, and even entire ships. As modern vessels rely more heavily on software, their complexity has surged, leading to challenges in integrating diverse equipment and systems from multiple providers. Moreover, a discernible gap in systems engineering practices has further compounded the situation [2].

As a consequence, the process of designing, constructing, operating, and ensuring the safety of ships and other maritime/offshore assets has grown significantly more demanding. Balancing cost considerations with environmental impact and safety concerns has become an intricate task [2].

However, the OSP emerges as a transformative force, offering a comprehensive solution to the maritime industry. By equipping professionals with essential tools and streamlined working processes for technical systems engineering, the platform empowers efficient and effective construction and maintenance of digital twins. These digital replicas facilitate seamless system integration, rigorous testing, and meticulous verification, ushering in a new era of reliability and innovation in the maritime sector. The OSP paves the way towards enhancing performance, sustainability, and safety standards across the maritime landscape [2].

Expanding upon the foundation laid by the Functional Mockup Interface (FMI) standard, the core principles of this approach encompass:

- Facilitating Model Reusability and Digital Twin Adoption: A paramount goal is to foster the seamless re-utilization of simulation models and digital twin technologies across diverse organizations. This is achieved by safeguarding sensitive intellectual property through the

encapsulation of models and control system software within black-box executable. By preserving the security of proprietary information, the FMI standard encourages widespread adoption and collaboration.

- Unifying Diverse Simulation Tools and Programming Languages: A key facet of the FMI standard is to establish a universal framework for connecting models and control systems from various simulation tools and programming languages. Through this unified approach, large-scale co-simulations become feasible, enabling virtual system integration on a comprehensive level. This unification enhances the efficiency and effectiveness of simulation processes, promoting seamless interoperability.
- Encouraging Collaborative Cross-Organization Initiatives: Emphasizing transparency and open-source principles, the FMI standard actively encourages cross-organization cooperation. By providing an open and accessible platform, the standard paves the way for organizations to collaborate seamlessly. Such cooperation fosters innovation, knowledge sharing, and fosters a thriving ecosystem of simulation-based advancements.
- In summary, the Functional Mockup Interface standard represents a transformative approach, empowering organizations to optimize their simulation efforts while safeguarding intellectual property. Through its unifying and open principles, FMI fosters a collaborative environment, propelling the development of cutting-edge simulation technologies and their broader adoption across industries.

The primary components of OSP are outlined below:

- OSP libcosim: This is a C/C++ library designed for co-simulation, which can be seamlessly integrated into any simulation software requiring co-simulation capabilities.
- Demonstration applications and illustrative examples: These provide practical insights into utilizing the library, accompanied by a range of supportive tools.
- OSP Interface Specification: This constitutes an emerging standard and ontology aimed at simulation models of maritime equipment and systems.
- OSP Reference Models: Within this set, you will find exemplary model implementations adhering to the Interface Specification. They serve as an excellent starting point and source of inspiration for crafting your own models.

The majority of OSP components utilize the MPL 2.0 open-source license, which necessitates sharing updates and enhancements to the components with the community. However, applications incorporating these components can remain private. Additionally, certain parts of OSP are governed by the MIT license, allowing for unrestricted distribution and modification.

OSP Reference Model Gunnerus-DP The goal of this demo case [14] is to demonstrate the separation of control systems and simulators within the framework of libcosim, as well as the use of network Functional Mock-up Units (FMUs).

This type of topology is common in hardware-in-the-loop (HIL) simulations. The control system is deployed on actual production hardware and connected to an external simulator in HIL installations. This external simulator can run on any suitable hardware platform as long as it provides a compatible interface with the control system. This structure is also common in software-in-the-loop (SIL) simulations. SIL is primarily used for testing and certifying system functionality, and it does not require the control system to be deployed on production hardware.

This demo use case demonstrates the potential of doing simulations in circumstances where control systems and simulators are decoupled in this manner using libcosim and the OSP configuration format. The simulation features a Dynamic Positioning (DP) driven vessel performing a box maneuver while subjected to external stresses. Gunnerus, NTNU's research vessel (Figure 1), was chosen as a reference model for both the vessel and the control plant.

To summarize, this demo use case effectively demonstrates the successful execution of simulations using libcosim with an architecture that separates control systems and simulators. The approach entails deploying a DP-controlled vessel in a box maneuver scenario while subjected to external stresses, all while utilizing network FMUs and the OSP configuration format.

Figure 1: Gunnerus [14]

System Description The HIL/SIL structure divides the simulation into two instances: one to run the control system (Figure 2) and another to run the simulator models (Figure 3). These instances can run separately on different hardware. In addition, a single system configuration (Figure 4) is given, in which both the control plant and the simulator run in the same instance. Communication between the control system and the simulator takes place via a specifically developed network FMU, which permits state interchange. The network FMU in the control system, for example, transmits thruster commands while receiving vessel and environment states. The network FMU, however, is not used by the single system setup. Table 1 shows a list of all FMUs.

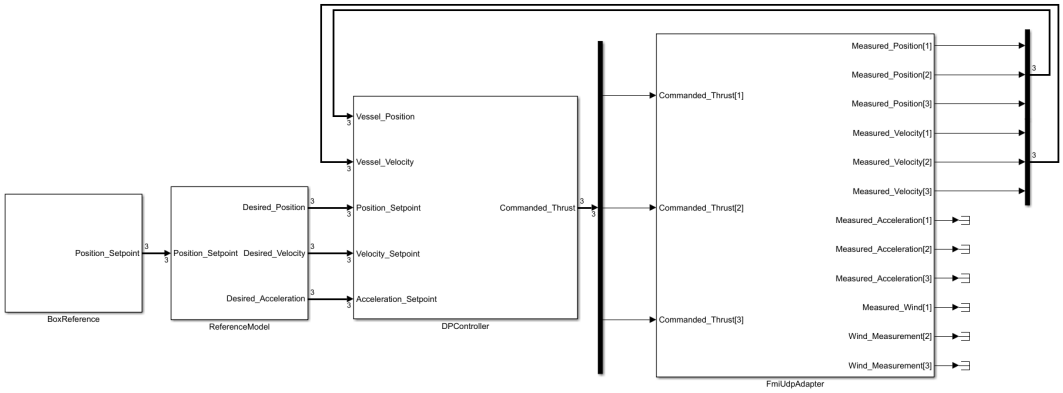

Figure 2: Control System Configuration [14]

OSP Interface Specification The OSP interface specification (OSP-IS) is used in this demonstration to simplify model connections. OSP-IS enables the grouping of FMU variables into higher-level variable groups as specified in a modelDescription.xml file. This makes it unnecessary to connect each individual FMU variable, enabling configurators to work more abstractly.

With the exception of discussing one FMU and its specifics, we won't dive into comprehensive FMU theory for the goal of understanding Gunnerus-DP as a System Under Test. For further information on all FMUs, the reader is pointed to [16]. Additionally, we shall only use single system architecture (Figure 4) for our experiments.

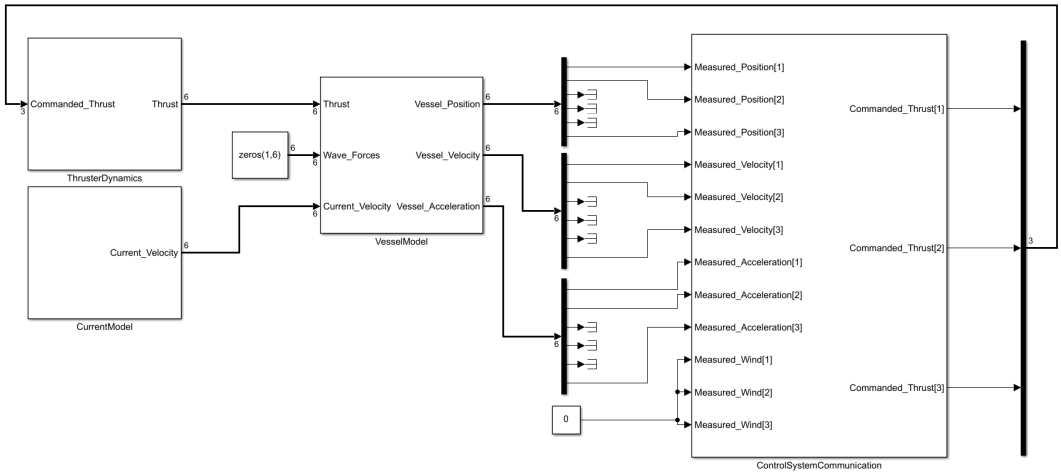

Figure 3: Simulator Models Configuration [14]

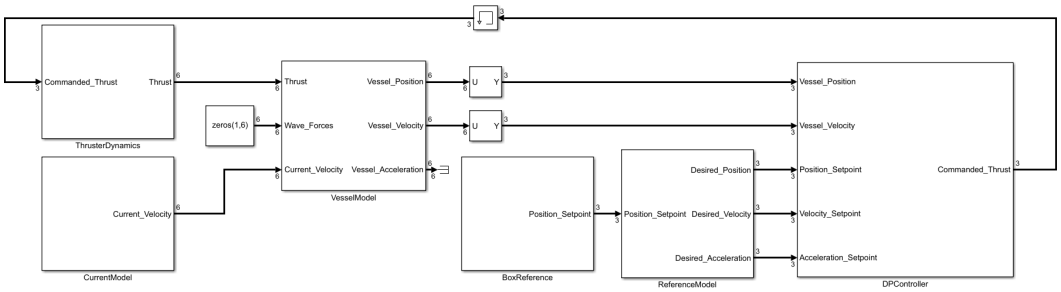

Figure 4: Single System Configuration [14]

Table 1: List of FMUs in the Gunnerus-DP case [15]

FMU name	**Description**
Box Reference	Provides setpoints for a box maneuver
Control System Communication	Network FMU handling communication between control system and simulator
Current Model	Current model generating surface current velocities
DP Controller	3-Dof DP controller
Reference Model	Reference model providing setpoints to DP
Simulator Communication	Network FMU handling communication between control system and simulator. Shown in Figure 2 as FmiUdpAdapter.
Thruster Dynamics	Thruster dynamics
Vessel Model	6-dof vessel model

Vessel Model - FMU The interface for this FMU, which implements vessel dynamics, is depicted in Figure 5. The current velocity is provided by an external simulation model, specifically the current model FMU, and it accepts inputs for wave forces and thrust. The FMU creates vessel velocities and accelerations in the BODY frame as well as the vessel's position in the NED frame based on these inputs.

Notably, since the wave forces and thrust ports are connected internally to the same summation block, extra external forces can be applied to either port. However, the present velocity port only takes into account the input vector's x and y components, allowing the other entries to be set to zero. The interfaces of the FMU are described in Tables 2 and 3 to provide a clearer understanding of the linkages and interactions with the model. By organizing FMU variables into higher-level variable groups, the OSP interface specification (OSP-IS) reduces the complexity of individual FMU variable connections and enables configurators to operate at a more abstract level, simplifying model linkages.

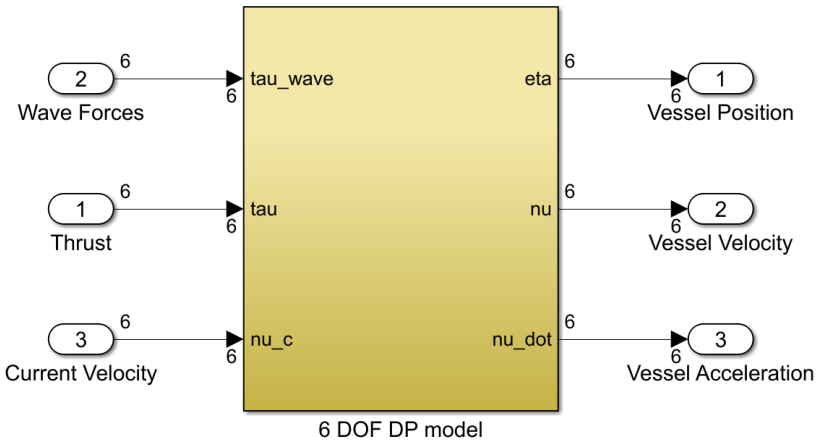

Figure 5: Vessel Model [14]

We will simply use the current velocity variable group in Figure 5 and derive its values from STGEM. We must establish input and output ranges, which will be covered in more detail in subsequent sections. The following part will look at STGEM.

2.5. System Testing Using Generative Models (STGEM)

A sophisticated technique called System Testing Using Generative Models (STGEM) [4] was created specifically for using while testing cyber-physical systems in a black-box fashion. Its main purpose is to assist in the robustness optimization-based falsification of Signal Temporal Logic (STL) requirements. An online generative machine learning model that generates system inputs with low robustness outputs is used to attain this special capacity.

STGEM [4] is a cutting-edge development that is constantly expanding. Its capabilities are being continually improved by the active integration of new algorithms and features. These modifications are intended to increase the tool's efficiency in methodically finding and correcting potential flaws and vulnerabilities in intricate cyber-physical systems. STGEM [4] provides an effective framework for carrying out rigorous testing and verification of system requirements within the context of cyber-physical domains by employing cutting-edge techniques and continuous research breakthroughs.

The tool currently uses the algorithms described in the following articles:

- Jarkko Peltomäki and Ivan Porres, Falsification of Multiple Requirements for Cyber-Physical Systems Using Online Generative Adversarial Networks and Multi-Armed Bandits. The 6th. Intl. Workshop on Testing Extra-Functional Properties and Quality Characteristics of Software Systems, ITEQS 2022.

- Jarkko Peltomäki, Frankie Spencer and Ivan Porres, Wasserstein Generative Adversarial

Table 2: Thruster Dynamics inputs and outputs (I/O) [15]

Name	**I/O**	**Unit**	**Description**
Wave_Forces[1]	I	N	Wave force in surge
Wave_Forces[2]	I	N	Wave force in sway
Wave_Forces[3]	I	N	Wave force in heave
Wave_Forces[4]	I	Nm	Wave force in roll
Wave_Forces[5]	I	Nm	Wave force in pitch
Wave_Forces[6]	I	Nm	Wave force in yaw
Thrust[1]	I	N	Thrust force in surge
Thrust[2]	I	N	Thrust force in sway
Thrust[3]	I	N	Thrust force in heave
Thrust[4]	I	Nm	Thrust moment in roll
Thrust[5]	I	Nm	Thrust moment in pitch
Thrust[6]	I	Nm	Thrust moment in roll
Current_Velocity[1]	I	m/s	Current velocity in surge
Current_Velocity[2]	I	m/s	Current velocity in sway
Current_Velocity[3]	I	m/s	Current velocity in heave (always zero)
Current_Velocity[4]	I	rad/s	Current velocity in roll (always zero)
Current_Velocity[5]	I	rad/s	Current velocity in pitch (always zero)
Current_Velocity[6]	I	rad/s	Current velocity in yaw (always zero)
Vessel_Position[1]	O	m	Vessel position in north
Vessel_Position[2]	O	m	Vessel position in east
Vessel_Position[3]	O	m	Vessel position in down
Vessel_Position[4]	O	m	Vessel position in roll
Vessel_Position[5]	O	m	Vessel position in pitch
Vessel_Position[6]	O	m	Vessel position in yaw
Vessel_Velocity[1]	O	m/s	Vessel velocity in surge
Vessel_Velocity[2]	O	m/s	Vessel velocity in sway
Vessel_Velocity[3]	O	m/s	Vessel velocity in heave
Vessel_Velocity[4]	O	rad/s	Vessel velocity in roll
Vessel_Velocity[5]	O	rad/s	Vessel velocity in pitch
Vessel_Velocity[6]	O	rad/s	Vessel velocity in yaw
Vessel_Acceleration[1]	O	m/s^2	Vessel acceleration in surge
Vessel_Acceleration[2]	O	m/s^2	Vessel acceleration in sway
Vessel_Acceleration[3]	O	m/s^2	Vessel acceleration in heave
Vessel_Acceleration[4]	O	rad/s^2	Vessel acceleration in roll
Vessel_Acceleration[5]	O	rad/s^2	Vessel acceleration in pitch
Vessel_Acceleration[6]	O	rad/s^2	Vessel acceleration in yaw

Table 3: Simulator Communication's OSP-IS variable groups [15]

Name	**Comprised variables**	**Description**
thrust	Thrust[1], Thrust[2], Thrust[3], Thrust[4], Thrust[5], Thrust[6]	Total thruster forces in BODY
wave_forces	Wave_Forces[1], Wave_Forces[2], Wave_Forces[3], Wave_Forces[4], Wave_Forces[5], Wave_Forces[6]	Total wave forces in BODY
current_velocity	Current_Velocity[1], Current_Velocity[2], Current_Velocity[3], Current_Velocity[4], Current_Velocity[5], Current_Velocity[6]	Current velocity vector
position_6dof	Vessel_Position[1], Vessel_Position[2], Vessel_Position[3], Vessel_Position[4], Vessel_Position[5], Vessel_Position[6]	Vessel position in NED frame
position_3dof	Vessel_Position[1], Vessel_Position[2], Vessel_Position[6]	Vessel position in NED frame
velocity_6dof	Vessel_Velocity[1], Vessel_Velocity[2], Vessel_Velocity[3], Vessel_Velocity[4], Vessel_Velocity[5], Vessel_Velocity[6]	Vessel velocity in BODY frame
velocity_3dof	Vessel_Velocity[1], Vessel_Velocity[2], Vessel_Velocity[6]	Vessel velocity in BODY frame
acceleration_6dof	Vessel_Acceleration[1], Vessel_Acceleration[2], Vessel_Acceleration[3], Vessel_Acceleration[4], Vessel_Acceleration[5], Vessel_Acceleration[6]	Vessel acceleration in BODY frame
acceleration_3dof	Vessel_Acceleration[1], Vessel_Acceleration[2], Vessel_Acceleration[6]	Vessel acceleration in BODY frame

Networks for Online Test Generation for Cyber Physical Systems, The 15th Intl. Workshop on Search-Based Software Testing, SBST 2022.

J. Peltomäki and I. Porres et al. Ivan [17] concentrated on how to assess whether a system in a cyber-physical system conforms with particular safety requirements. To describe the safety requirements, signal temporal logic serves as a kind of language. After recasting this task as an optimization problem, the authors suggest utilizing generative adversarial networks (GANs), a subclass of machine learning technique, to help find any breaches of these constraints. They provide a technique that employs multiple GANs, each of which is focused on a distinct demand, but only trains one GAN at a time. The requirements can be promptly determined while resources are conserved. The research' findings show that this method is better than using just one GAN or all GANs.

WOGAN [18] is a Wasserstein generative adversarial network-based online test generating algorithm. In this article, the authors have described WOGAN's operation and evaluate its results in the SBST 2022 CPS tool competition for a self-driving car AI. In [19], the algorithm is thoroughly discussed. With reference to generative adversarial networks, they have developed an original and online black-box test creation approach in this study. The algorithm's strength is in its capacity to operate, with the exception of the choice of issue feature representation, without explicit domain expertise. It also includes a generator that may be used to create high-fitness tests for the System Under Test (SUT). This generator may provide insightful information on investigating the SUT in greater detail.

WOGAN (Weighted Objective GAN) algorithm has been shown through experimentation to attain performance on par with earlier competitive algorithms. Notably, WOGAN not only finds errors in the SUT but also creates a wide range of tests that fail. The WOGAN algorithm's merits should be independently evaluated in the context of the SBST 2022 CPS (Cyber-Physical Systems) testing competition, it is crucial to highlight that evaluation is based on a single experiment.

STGEM will be used in our endeavor to test the Open Simulation Platform (OSP). It will make use of a specific OSP reference model, which will be described later in problem formulation section. It will be the responsibility of STGEM to generate specific parameters using the many models it offers. Those parameters will be passed to OSP reference model to run simulations and return results.

2.6. Farn

The main goal of Farn [3], a creative n-dimensional case generator, was to offer a solid foundation for parameterizing and running simulation scenarios. However, due to its adaptability and ability to accommodate a broad range of applications, it shows to be more versatile than its original design intent. The word "farn" is derived from the Barnsley fractal and represents the tool's aptitude for navigating intricate and varied design environments.

Fundamentally, Farn provides a wide range of capabilities to make the creation and execution of simulation cases more effective. Farn efficiently explores the design space with a variety of sampling algorithms, such as fixed, linSpace, and uniformLHS, enabling users to consider many scenarios and maximize outcomes. The application automatically creates case folder structures, ensuring that each simulation case's data is well-organized and easily accessible.

Furthermore, Farn enables users to add unique configuration files or input data tailored to certain instances by allowing users to copy arbitrary files from a template folder to case folders. The ability to fine-tune simulation settings for each scenario is made possible by the development of parameter files for each case. Farn streamlines the batch processing of several cases by automating crucial simulation operations by running user-defined shell command sets inside the case folders.

Farn is a key tool in the marine simulation space thanks to its capacity to create case-specific OSP (co-)simulation files. Because of its adaptability, which goes beyond particular applications, it is a crucial tool for academics and engineers who want to effectively investigate and evaluate complex systems. Utilizing Farn allows users to fully realize the potential of n-dimensional design spaces, effectively optimize simulation workflows, and get insightful knowledge into a variety of simulation scenarios.

Farn provides an interface to interact with OSP through Ospx. Ospx is a cutting-edge extension package that interfaces with FARN without a hitch and provides improved capabilities for building OSP (co-)simulation cases using functional mockup units (FMUs). Ospx considerably broadens the range of simulation possibilities by adding FMUs, allowing users to more accurately and flexibly represent complex systems.

By making it simple to create case-specific OSP (co-)simulation configuration files, this extension package enables users to easily adapt simulations to their particular needs. Ospx simplifies the integration of FMUs into the simulation setup by offering a user-friendly interface, enabling researchers and engineers to explore complex scenarios and thoroughly analyze the behavior of interconnected systems.

Additionally, Ospx goes above and beyond standard simulation capabilities by allowing users to keep track of cosim (co-simulation) development in real-time and save the final simulation results as a pandas dataframe. Users may more effectively examine simulation results and gain insightful information from the simulation data thanks to this strong feature. Ospx, which seamlessly integrates with farn, is a valuable tool for the maritime simulation community, providing cutting-edge capabilities to model, simulate, and analyze complex systems utilizing functional mockup units.

In our work, we will conduct OSP simulations using a variety of code snippets from Farn (attached in the appendix section). These code snippets will aid in data preparation for OSP, use case execution, data collection following simulation, and post-analysis. It's vital to note that Farn does not parameterize the parameters using generative machine learning algorithms. Instead, it generates these parameters in case folders using statistical techniques. To build those parameters for our reference model, we'll utilize STGEM (a machine learning approach), and then we'll perform simulations on top of that. This will assist us in doing in-depth performance testing of a particular OSP use case that can be applied to other use cases.

3. Related Work

S. Skjong et. al [20] has concentrated on the use of distributed co-simulations for virtual prototyping in the Norwegian maritime sector. The author discusses the difficulties in adopting co-simulations in maritime industrial applications and offers a general overview of the concept as well as information on related subjects such system modularity and stability analysis. The construction of generic domain models for marine offshore vessels and their subsystems, which may be used in various co-simulation case studies, is another key component of the thesis. The project aims to maintain technical leadership in the maritime sector while streamlining work processes and cutting expenses. The author emphasizes the value of innovation and the necessity of cost-cutting measures while maintaining the high wages and R&D expenditures in the Norwegian marine sector.

As described in the Ph.D. work [20] by S. Skjong, the case studies in this research have successfully used fmiCpp, a C++ framework created for creating co-simulation models. The results of this study show how fmiCpp has many real-world uses and how it has a big impact on the development of virtual prototyping. The researchers have successfully addressed the co-simulation issues by utilizing this C++ framework, which has encouraged the creation of large-scale system simulators. This opens the door for increased effectiveness, lower costs, and better performance across a variety of disciplines.

The thorough and original approach used in this study paper [20] adds significant knowledge to the area of multi-disciplinary integrated systems modeling and simulation. It also emphasizes how critical it is to have expertise and competency in using cutting-edge simulation technologies in order to maximize the use of virtual prototyping in practical applications.

Virtual prototyping, which combines simulation and modeling of multidisciplinary systems, has enormous potential for cutting development costs and improving system performance as a whole. The availability of appropriate simulation models, computer software that enables the quick assembly of models into complete system simulators, and—most importantly—the knowledge and experience necessary to effectively operate such simulators are, however, all prerequisites for the implementation of this strategy.

S. Skjong et. al [21] gives a review that the maritime industry is under pressure to be more environmentally friendly and technologically advanced. It also faces strong competition and pressure to reduce costs and time-to-market. Maritime industry has a strong reputation for advanced technology and expertise in complex operations. However, it also faces high costs, including high wages and R&D expenses, and is dependent on the oil industry. The ViProMa project aims to investigate the challenges and potential benefits of using distributed co-simulation technology in the maritime industry.

The study paper [21] proposes a brand-new co-simulation modeling method that is physics-based and based on the Functional Mock-up Interface. This novel method offers a more effective and streamlined solution by doing away with the necessity for time-consuming simulations of intricately interconnected power networks. The researchers make it possible to conduct extensive co-simulation investigations by integrating the real-time capabilities of the power system model into vessel dynamics models [21]. This seamless integration improves the simulations' accuracy and dependability while also opening the door to a deeper comprehension of the dynamics within complex systems.

With its broad design applications across the full product life cycle, virtual prototype technology is quickly expanding in today's fast-paced technological landscape [21]. This research gives engineers and designers the tools they need to make educated choices at every level of the product development process by leveraging the power of virtual prototyping. The virtual prototype approach reforms the design process from conceptualization to testing and optimization, resulting in decreased development costs, expedited time-to-market, and improved overall product performance. The research highlights the critical function of contemporary simulation tools in creating efficient and successful multi-disciplinary integrated systems modeling by utilizing the potential of co-simulation and the Functional Mock-up Interface. The findings in this article show the prospect of influencing a more sustainable and technologically advanced future as the globe adopts virtual prototyping.

I. Porres et. al [22] introduces a new algorithm for automatic performance testing that utilizes an online version of the Generative Adversarial Network (GAN) to optimize test generation. The

goal of this approach is to create a test suite with a large number of tests that can reveal performance defects, using a limited budget. The GAN is used to generate tests and predict their results, and it is trained online as the tests are being executed. This method does not require a pre-existing training set or model of the system being tested. The effectiveness of the algorithm is demonstrated through an initial evaluation using a sample test system, and the results are compared to other possible approaches. The authors believe that this algorithm serves as a proof of concept and hope that it will stimulate further research on the use of GANs for test generation.

The GAN is used to create tests and forecast their results, and an initial assessment on a sample test system is used to show how effective it is. To evaluate its effectiveness, the results are then contrasted with those of alternative ways. It's noteworthy that the authors stress the fact that their approach [22] is a proof of concept and that they hope it will inspire more investigation into the application of GANs for test generation. This method attempts to address the possible problem of the generator constantly delivering only one useful test, therefore satisfying the second quality criterion, by utilizing the power of online GAN training.

J. Peltomäki and I. Porres et. al Ivan in [17] focused how to determine whether a system in a cyber-physical system complies with specific safety standards. Signal temporal logic is a language used to explain the safety requirements. The authors propose using generative adversarial networks (GANs), a class of machine learning method, to help discover any violations of these constraints after converting this task into an optimization problem. They offer a method that trains just one GAN at a time while using several GANs, each of which is focused on a different requirement. Resources are conserved, and the requirements can be assessed more quickly. The results of the studies indicate that this strategy is superior to utilizing a single GAN or all GANs at once. In our case, the system under test is maritime simulation model instead of cyber-physical system and we can make use of this work particularly using algorithms implemented based on this work like STGEM (System Testing Using Generative Models) [4].

STGEM (System Testing Using Generative Models) [4], as explained in Background section, is a testing tool that can be used to evaluate the performance of cyber-physical systems through black-box testing. It allows users to falsify requirements written in Signal Temporal Logical (STL) by optimizing for robustness. This is done by training a machine learning model to generate input for the system that will result in low robustness values. The use of this generative model allows STGEM to effectively identify and target specific requirements within the system, enabling developers to more thoroughly test the correctness and reliability of their systems. In our work, we will integrate the Open Simulation Platform (OSP) as system under test into STGEM.

Farn [3], as explained in Background section, is a tool that can generate and execute simulation cases in n-dimensional spaces. Its primary purpose is to create and run simulations, but it is flexible enough to be used for a variety of applications. Farn can sample the design space using fixed, linSpace, or uniformLHS strategies, and it can generate case folders, copy template files, create case-specific parameter files, execute user-defined shell commands, and build OSP co-simulation files. Farn is named after the Barnsley fractal and is capable of running simulation cases in a batch process. In our work, we will take inspiration from Farn for running OSP simulations inside STGEM codebase. Note that STGEM and Farn are both written in python language so we do not need code translation to run in STGEM tool.

4. Problem Formulation

Innovative methods are required for research in the maritime sector to validate complex software systems on the OSP co-simulation platform. Utilizing Generative Adversarial Networks (GANs) to create pertinent test situations is a fascinating area for investigation. The goal of this research is to develop a brand-new and efficient method of validation by utilizing the power of GANs. GANs, which are renowned for their capacity to build accurate data distributions, may be able to produce a variety of difficult test situations that closely resemble real-world settings. The findings of this analysis have the potential to improve the validation process, resulting in maritime software systems that are safer and more dependable.

This thesis explores the design and implementation of a GAN-based algorithm for automatic performance assessment within the OSP co-simulation platform in an effort to develop cutting-edge performance testing approaches. The study looks at different design options and implementation tactics in an effort to maximize the algorithm's efficacy, efficiency, and adaptability. Utilizing GANs for performance testing could transform how we examine system behavior and provide insightful data on how well they function in various scenarios. The existing available algorithms has the ability to drastically minimize the human work and time necessary for performance testing, giving maritime engineers a deeper comprehension of system performance and assisting in the creation of robust and effective software systems.

We must specify the right questions we are looking for in order to give the correct responses. We shall attempt to address the following research questions in this thesis:

- Can Generative Adversarial Networks (GAN) be exploited to generate relevant test scenarios for the validation of maritime software systems modeled within the OSP co-simulation platform?

  In this reseach question, we investigate the efficacy of developing test cases specifically for evaluating maritime software systems using GANs, a subtype of deep learning models (in our case, from STGEM). The OSP co-simulation platform, which is used to model and simulate maritime systems, is the subject of the thesis. The main goal is to look at how GANs could be utilized to develop relevant and precise test scenarios to assess the functioning and robustness of these software systems in a marine environment.

- What are the most effective design and implementation approaches for a GAN-based algorithm for automatic performance testing of systems modeled within the OSP co-simulation platform?

  The design and implementation of a GAN-based algorithm meant for autonomous performance testing are examined in this research question. The objective is to comprehend the most effective and practical methods for developing and implementing such an algorithm within the framework of the OSP co-simulation platform. It might entail investigating various GAN structures, approaches, data pretreatment strategies, and other pertinent aspects essential to the algorithm's success in testing maritime systems.

- How accurate and reliable is the GAN-based algorithm at predicting the performance of simulated systems within OSP?

  Within the OSP co-simulation platform, this research question aims to assess the accuracy and dependability of the GAN-based method in forecasting the performance of simulated systems. Examining how well the test scenarios produced by GAN correspond to actual maritime software system behaviour is the main objective.

- What are the benefits and limitations of using a GAN-based algorithm for performance testing compared to other approaches?

  Investigating the advantages and disadvantages of utilizing a GAN-based algorithm for performance assessment, specifically in comparison to alternative approaches (like Farn in our case), is one of the goals of this research question. Researchers would investigate how GAN-based testing stacks up against traditional testing procedures or other cutting-edge methodologies (again, Farn in our example), in terms of effectiveness, efficiency, scalability,

and scenario-adaptiveness. The objective of the inquiry is to understand when GAN-based testing performs well and when it might not be the ideal solution for performance assessment in the maritime software industry.

In summary, the research questions concern investigating the potential of Generative Adversarial Networks (GANs) for producing relevant test scenarios, utilizing existing GAN-based algorithms for performance testing, assessing the precision and dependability of the GAN-based algorithm, and comprehending the benefits and drawbacks of using GANs for performance testing. The main goal is to figure out how GANs may be utilized to better assess and evaluate maritime software systems in the context of the OSP co-simulation platform. The thesis aims to determine the utility and efficacy of GAN-based testing in contrast to other testing methods. We'll take a general look at the responses to these study questions and then provide a summary in the conclusion section. We shall focus on finding answers to these questions under the impact of the tools (OSP, STGEM, and Farn),

5. Method

In this section, we will define our strategy for addressing the previously stated research questions. We recognize the significance of a methodical and detailed technique in properly addressing each research question. We will connect the investigated work discussed earlier using different architectures. Using this strategy, we will be able to obtain valuable information and insights relevant to the research topics, as well as find existing studies, ideas, and best practices in the field.

In addition, we will present a brief review of the existing Farn architecture. We hope to accomplish this by demonstrating the basis upon which our research is based, as well as the context in which our changes and integrations with the STGEM tool take place. The incorporation of OSP in STGEM tool is an important part of our research because it expands the capabilities of the existing system and adds to the attainment of our research goals. We will go over the changes done to guarantee a smooth integration, emphasizing the benefits and enhancements it provides to the overall architecture.

The overall strategy indicated above seeks to give a clear and well-defined framework for conducting our study, ensuring that the research subjects are well addressed. This strategy takes advantage of both Farn's current strengths and the additional capabilities made possible by the inclusion of the STGEM tool. To better understand the integration, we will look at the existing Farn architecture and its interface with OSP.

Figure 6 depicts the Farn architecture, which works in tandem with OSP. The OSP component includes a number of critical pieces such as libraries, applications, standards, and reference models. Farn's integration with OSP, on the other hand, is intended to considerably optimize the simulation process. Farn accomplishes this by loading Functional Mock-up Units (FMUs) and generating variable parameters using a statistical approach. Multiple versions of tese parameters are generated, each of which is stored in a separate test folder. Following that, OSP simulations are run for each of these test folders, allowing for a thorough study of numerous scenarios. Then, we can perform post-processing on the obtained results after the simulations are completed, providing significant insights and analyses to properly complete experiments.

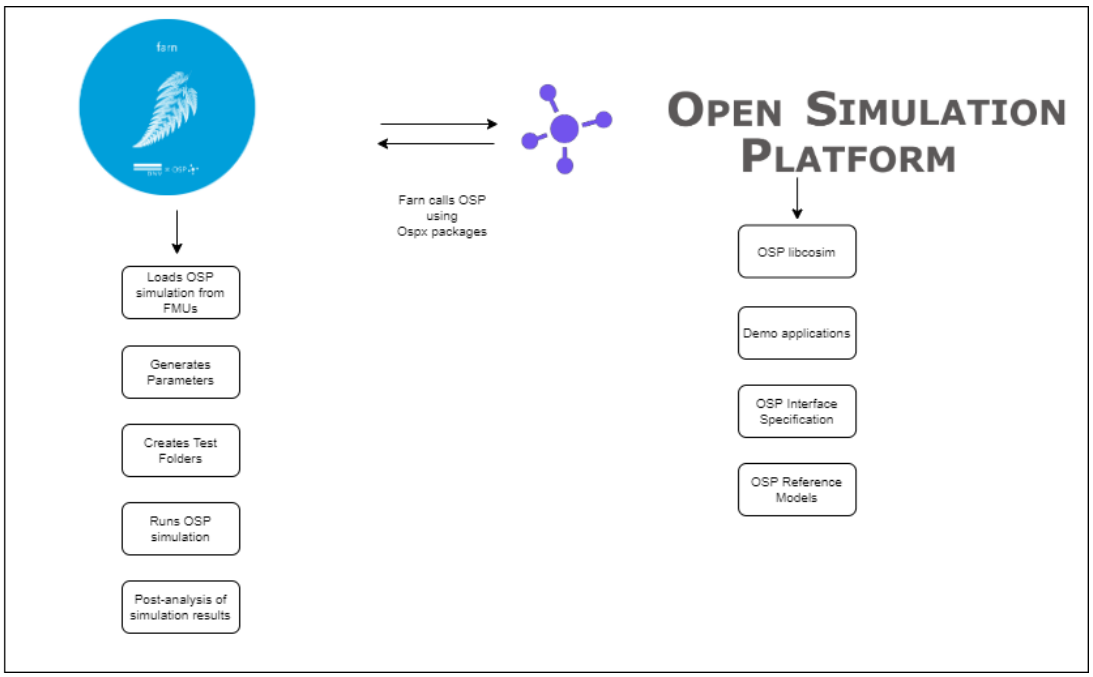

Figure 6: Architecture with Farn and OSP

The Figure 7 depicts our present architecture, which shows how we have carefully assigned the task of parameter creation to the STGEM tool. With this connection, we can import Functional Mock-up Units (FMUs) and build numerous test folders, each with its own set of settings. Following

that, OSP simulations are run on these various parameter sets. Following the simulations, our system performs post-processing analysis on the results to extract significant insights.

The STGEM tool adds value to our design by providing several models for parameter generation, each with its own set of features. In this study, we will concentrate on the smooth integration of one specific model, which will be mentioned further in the Implementation section. This targeted strategy enables us to fully utilize the capabilities of the chosen model to produce parameters, allowing for a thorough study of many scenarios throughout the OSP simulations. By using the STGEM tool and adopting a specific model, we hope to optimize the research process and gain valuable results that contribute considerably to the aims of our study.

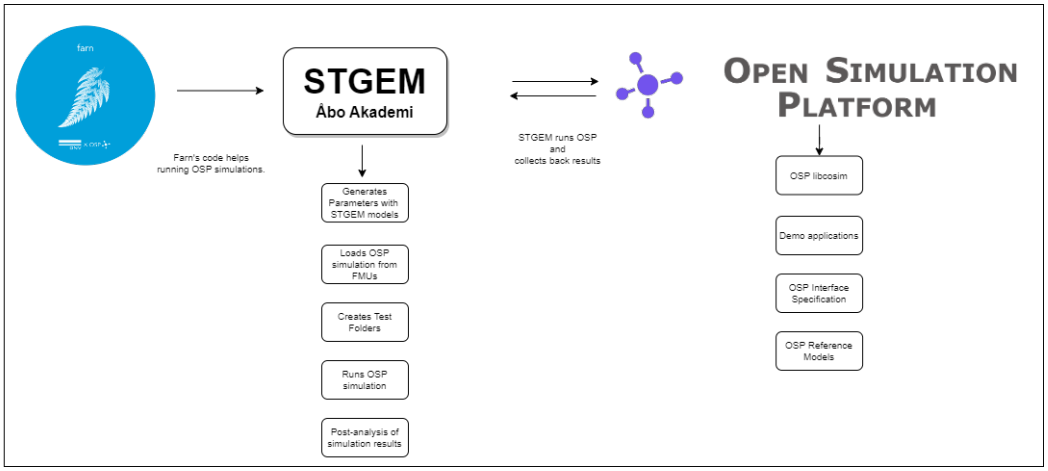

Figure 7: Our Architecture with STGEM, Farn and OSP

Furthermore, this complete strategy, which incorporates a case study as our research method, will allow us to properly address our research questions. We used a case study approach to investigate how the STGEM can be used to generate more appropriate and concise test cases in the context of OSP. We selected the case study approach because it allowed us to conduct a detailed and in-depth study of a real-world example such as "Gunnerus-DP" use case for testing OSP. We will be able to make well-defined assumptions about the success of our approach within the OSP environment by closely studying our existing architecture and its integration with the STGEM tool.

Lastly, our research will enable us to differentiate the advantages of adopting the GAN-based method over other test approaches, such as Farn. By emphasizing the GAN model's distinct strengths and capabilities, we may demonstrate its benefits in creating relevant and successful test scenarios, thereby contributing to the improvement of testing approaches in OSP. We will illustrate how the integration of the STGEM tool and OSP stands out as a powerful and efficient solution, promising to considerably improve the testing process in contrast to older approaches such as Farn.

6. Ethical and Societal Considerations

Managing the social and economic impacts on coastal communities as well as ensuring the welfare of seafarers and adopting sustainable practices to lessen environmental impact are all vital in the maritime business. A strong emphasis on transparency, accountability, and responsible governance is necessary to develop public trust and a sustainable marine business. Regarding these limitations, we concentrate on lowering fuel consumption and other financial factors that will ultimately assist the maritime sector in testing the system as effectively as possible while also lowering the likelihood of unanticipated incidents.

We put a lot of attention on the technical parts of our project and make use of readily accessible tools, which naturally fits with our dedication to the open-source community's support and contribution. We not only gain from the skills and knowledge of the community as a whole by using open-source tools in our implementation, but we also encourage a spirit of cooperation and innovation. We avoid using any specific datasets in our work and instead concentrate on using and modifying existing techniques, rather than creating new ones. This method may not include intelligent algorithmic innovations, but it enables us to concentrate on the effective use of these tools in realistic situations.

Our commitment to using open-source tools further solidifies our belief in the value of free and open access to information and technology, enabling a larger audience to build on and improve our work for the benefit of the field as a whole. Through our contributions to the open-source community, we hope to encourage openness, accessibility, and inclusion while fostering a spirit of cooperation and advancement in the field of technology.

Our undertaking in the maritime sector gives moral and societal issues a lot of weight. We put a lot of effort into minimizing the negative social and economic effects on coastal communities, assuring the welfare of seafarers, and implementing sustainable practices. For a marine business to be sustainable and to win the public's trust, transparency, accountability, and responsible governance are essential. Although open-source software and technical features are our main areas of attention, we also want to work together and share knowledge with the community. We encourage openness, accessibility, and advancement in the marine industry through our contributions to the open-source community, ensuring a sustainable and responsible future.

7. Implementation

The visual representation of the system (as shown in Figure 7) in terms of implementation involves multiple integrations and addition in existing STGEM code. We need to inherit System Under Test (SUT) base class of STGEM to add new SUT (OSP in our case). We need to choose variables for which we would like STGEM to generate test cases. Once we have chosen the variables, we need to define its input and output ranges, and their data types. We also need to then define what will we do with every test case being generated. In our scenario, we would like to run OSP simulation controlling only defined variables and keeping everything else as it is. We will use some utilities to run OSP simulations and will explain those code files. We also need to define output metric based on what STGEM will monitor the output. Below is detailed explanation of every code item in Appendix section. It is important to explain them for results section.

We took a number of actions in order to add a new System Under Test (SUT) called OSP and enhance the current System Under Test (SUT) base class of STGEM. Let's divide the procedure into distinct, standardized steps:

- Inheriting the SUT Base Class: We create a new class, which we'll refer to as OSP SUT, that derives from the STGEM SUT base class in order to integrate the new SUT (OSP) into the existing STGEM framework. It will be simpler to integrate the OSP class with the current testing framework thanks to this inheritance, which gives it access to the functions and methods already provided in the STGEM base class.

- Choosing Variables for STGEM: The variables we want the STGEM to use to create test cases were then decided. For our experiments, we decided to use the current velocity of vessel. These variables will serve as the controls for the parameters used in the OSP simulation. It is crucial to choose the pertinent variables that will affect the behavior and outcomes of the simulation.

- Defining Input and Output Variable Ranges: After deciding on the variables, we must specify the data types and input ranges for each one. During the development of test cases, input ranges specify the acceptable values that each variable may have. This makes sure that the test cases that are generated cover a variety of scenarios and edge circumstances. Furthermore, defining data types is essential for managing and validating input values in an appropriate manner.

- Handling Test Case Generation: The STGEM will be in charge of creating test cases based on these inputs once the variables and their input ranges have been set. To fully evaluate the OSP simulation, the test case generating procedure should take into account a variety of input combinations.

- Running OSP Simulation with Defined Variables: We execute the OSP simulation for each created test case while maintaining a constant state for the rest of the system. This means that during each simulation run, all other variables that are not selected for test case generation will be maintained at their default or fixed values. This makes it possible to isolate and precisely test the OSP's behavior.

- Utilizing OSP Simulation Utilities: We need to use particular tools or code files created especially for running the OSP simulation. The routines and functions required to control the OSP simulation depending on the chosen variables and their values should be present in these utility files.

- Defining Output Metrics: We must specify output metrics that STGEM will track after each simulation run in order to assess the performance of the OSP simulation. The main facets of the OSP's performance and behavior that you wish to examine and validate should be represented by these metrics.

We successfully carried out simulation-based testing on the OSP system by following the procedures above and guaranteeing correct implementation and integration of the OSP class with the

STGEM testing framework. Each piece of code, along with the necessary documentation of utility functions and output metrics, is included in the appendix section. The results section, which presents and analyzes the results of the OSP simulation testing based on the specified metrics, will depend heavily on these justifications. The Figure 8 shows the STGEM tool code hierarchy to further aid in comprehension. Multiple GAN models can be used with STGEM, and it also contains a number of different System Under Tests (SUTs) as part of its implementation. The "problems" directory has been the main focus of our most recent improvements.

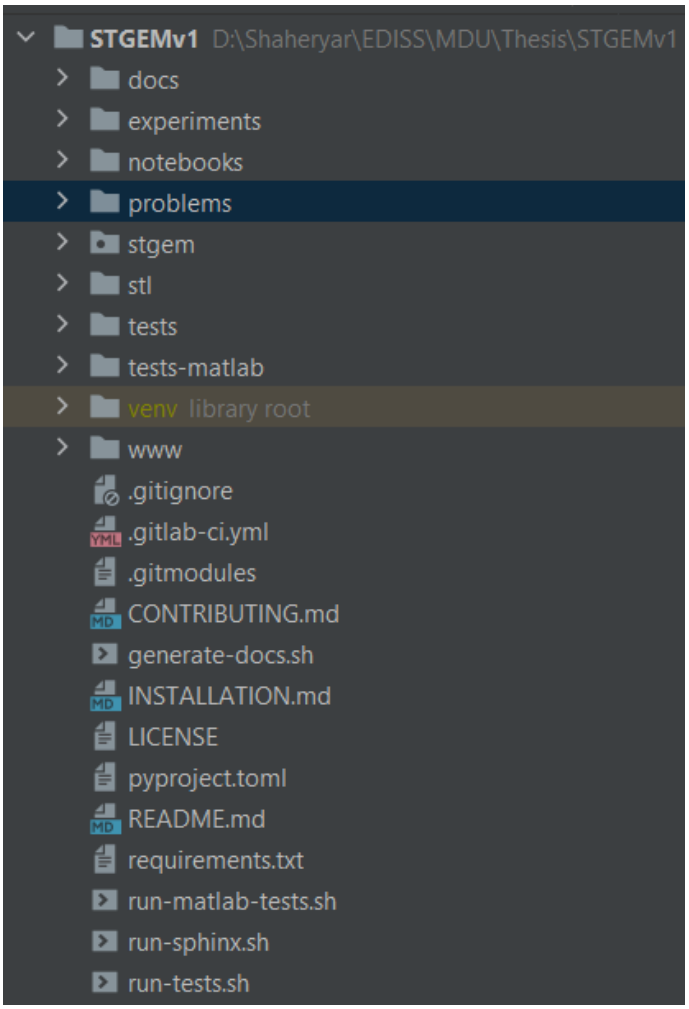

Figure 8: STGEM Tool Code Hierarchy

Figure 9 shows the "osp" directory, which is located inside the "problems" directory, contains numerous files and directories. We have configurations (we are using single system configuration described in section 2.) and FMUs for all the models that we covered in section in the directory "gennerus-dp." These models will be automatically loaded and executed by STGEM as OSP simulations. Test cases and the results they provide are contained in the directory "cases" produced by STGEM. This directory also houses three of the python files, which are described in more detail below. However, the overall code for OSP as System Under Test in STGEM is listed in Appendix (section 10.).

7.1. System Under Test (SUT) Implementation

Listing 1 shows sut.py which is main class that defines OSP as SUT. A System Under Test (SUT) class called 'OSPSUT', which represents an OSP Scenario, is defined in the code (Listing 1). The class takes vector inputs and produces vector outputs by inheriting from the parent class 'SUT'.

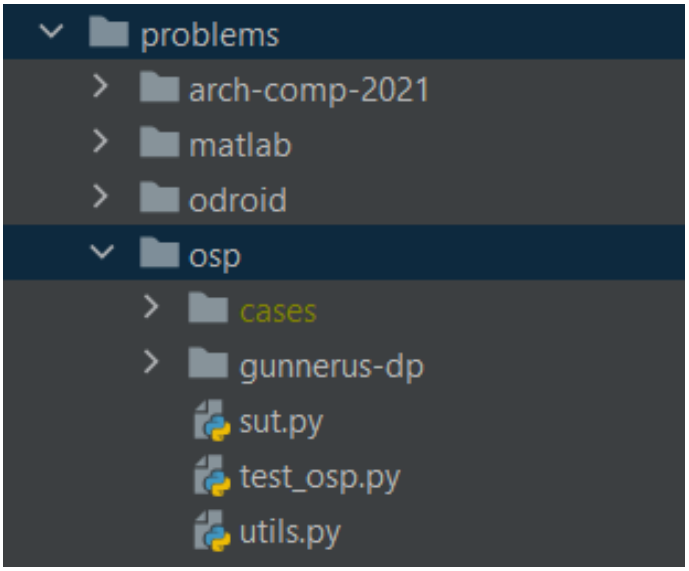

Figure 9: OSP in STGEM Tool

This class's goal is to simulate the OSP situation and allow users to interact with it. Key points about the 'OSPSUT' class:

1. The input and output ranges are set up in the class's initialization method, '__init__', which also specifies the input and output types as "vector." To maintain track of the test case number as it is being executed, it also initializes the variable 'case_number'.

2. A test case is carried out on the OSP simulation by the '_execute_test' method. One test case, represented by an object of type 'SUTInput', is all that is required as input for the function. A vector indicating the OSP's current velocity serves as the test case's input.

3. To ensure that the input values fall inside the specified range, the input_range is used to scale and descale the input values.

4. Using the 'execute_test_case' function with the inputs 'case_number' and current velocity, the method then tries to run the OSP simulation. The 'output' list contains the simulation outcome.

5. The simulation result will be included in the 'output' list if the simulation runs properly, and the method will return an 'SUTOutput' object that contains the result and contains no errors.

6. The exception is caught, and the 'error' variable is set to the caught exception, if an exception arises while the simulation is running (for example, if the simulation crashes or raises an error). The 'SUTOutput' object is then delivered, with the error containing the caught exception and the 'output' set to an empty array.

In summary, the 'OSPSUT' class is created so that it may communicate with STGEM and an OSP simulation by receiving vector inputs (which represent current velocity) and producing vector outputs. The vector outputs, which are extracted from simulation results, validate output variable range. This output vector represents absolute difference in standard deviation of actual vessel position and its desired position. The results contain the simulation execution and deals with any exceptions that could arise. In order to make integration with bigger systems easier, the class adheres to a clearly defined interface for input and output types.

7.2. OSP Utility Functions

The OSP simulation test cases' execution and post-processing utilities are displayed in the Listing 2. For simulation, file operations, and data processing, it depends on additional tools and libraries. The code is summarized as follows:

1. The code imports the essential modules and functions to run. Modules for regular expressions, data processing with NumPy and Pandas, file handling, and a few unique modules pertaining to the OSP simulation are all included in these imports.

2. The code defines a 'command_sets' dictionary that has various sets of commands for setting up, running, and processing OSP simulations. Every set has a unique key assigned to it (such as "prepare," "run," or "post") and consists of a list of command strings.

3. 'execute_test_case' function accepts two arguments, 'case_name' (a test case's unique identification) and 'case_parameters' (a set of OSP simulation settings). It establishes the case, runs the OSP instructions, and processes the simulation results afterward. The post-processing result is returned by the function.

4. 'execute_osp_commands' function accepts an object called cases, which is an assortment of OSP simulation scenarios. After producing parameter dictionary files and test case directories, it runs the defined OSP commands for preparation, running, and post-processing.

5. 'post_processing' function performs post-processing on the OSP simulation results stored in the cases object. It uses a mapping dictionary to extract certain data (standard deviation of actual vessel position and its desired position) from the outcomes, processes the data, and calculates a result metric (absolute difference in standard deviation of actual vessel position and its desired position) based on the target position and vessel position. The calculated result is returned by the function.

In total, the code provides utilities to 'OSPSUT' class in the execution of OSP simulation test cases, gathers the outcomes, analyses the information, and calculates a particular measure for each test case. The result measure shows how well the OSP simulation model anticipates the expected behavior by comparing the desired position and the vessel position.

7.3. Executing OSP SUT with STGEM

To test the functionality of an OSP with one of the STGEM GAN models (i.e., OGAN - Online Generative Adversarial Network), we created a unit test using Python's unittest framework. With the STGEM framework, the test focuses on using two distinct models (Random and OGAN) for the OSP simulation. The OSP simulation settings are investigated and optimized using the STGEM framework with the aim of reducing a particular objective (reducing distance between targeted and current vessel position) associated with the OSP scenario. Although the whole code is contained in Listing 3, we will concentrate on the class's key features below.

1. Import statements for several modules and classes needed for the test are included in the code. The imports include several parts of the STGEM framework, including 'STGEM', 'Search', 'Minimize', 'ObjectiveSelectorMAB', and the 'OSPSUT' class, as well as mathematical operations, operating system interfaces, the 'unittest' module for test cases, and others.

2. A test class called 'TestOSP', which derived from 'unittest', is defined in the 'unittest.TestCase'. Test techniques are included in this class to assess the models for OSP optimization.

3. 'test_models' tests the OSP optimization process using this test method, which is described within the 'TestOSP' class. The following setup is created by the code for a STGEM generator:

    - 'sut': The System Under Test (SUT) is an instance of the OSPSUT class, representing the OSP scenario.

- 'objectives': The objective of the optimization process is to minimize a specific metric related to the OSP scenario.
- 'objective_selector': The algorithm used for selecting the objective to optimize during the warm-up phase is an instance of 'ObjectiveSelectorMAB'.
- 'steps': Two search processes make up the optimization process i.e., one employs the Random method, while the other, OGAN (Online Generative Adversarial Network), uses a different approach. Each search step utilizes a particular model for parameter sampling and has a budget threshold for the total number of executions.

4. The 'run' method is then called to invoke the STGEM generator with the configuration specified above. Using the chosen models, the generator investigates and refines the OSP simulation parameters.

5. We can save the model as pickle file using 'os' library.

Overall, this class uses the STGEM framework and two separate models (Random and OGAN) and creates an optimization scenario for an OSP simulation. The optimization procedure is then carried out, and the models' projected performance is confirmed. The goal of the test is to validate the accuracy and dependability of the OSP optimization procedure.

7.4. Cosim Demo Application

Users of the OSP's demo application can conduct simulations on their own computers and view the outcomes in a variety of graphs. For use in simulation, the application enables loading of a specified reference model. We won't go into great depth about what the reference model does because it is outside the purview of our current research, but we are curious to look at the visuals to get a better grasp of the simulation results. The graphs given below were produced by loading and executing the GunnerusDP reference model within the simulation environment while using specific simulation settings (as stated in Figure 10). The visualizations offer insightful views into the operation and behavior of the simulation.

The figures include the following graphs for the reader's review:

- Vessel Position (Figure 11): The graph provides graphical picture of the vessel's motions and trajectory, which shows the Vessel Position throughout the simulation.
- DP Force command (Figure 12): The graph displays the Dynamic Positioning (DP) Force command, which illustrates the control forces used to keep the vessel in place.
- Desired Vs. Produced Thrust (Figure 13): This graph illustrates how well the simulation model matches the desired thrust output by contrasting the desired thrust and the produced thrust during the simulation.

Readers can learn a lot about the behavior and effectiveness of the OSP simulation e.g., using the GunnerusDP reference model as an example, by examining these graphs. The visualizations improve our comprehension of the simulation results and let us assess and analyze the simulation's accuracy and efficiency more thoroughly.

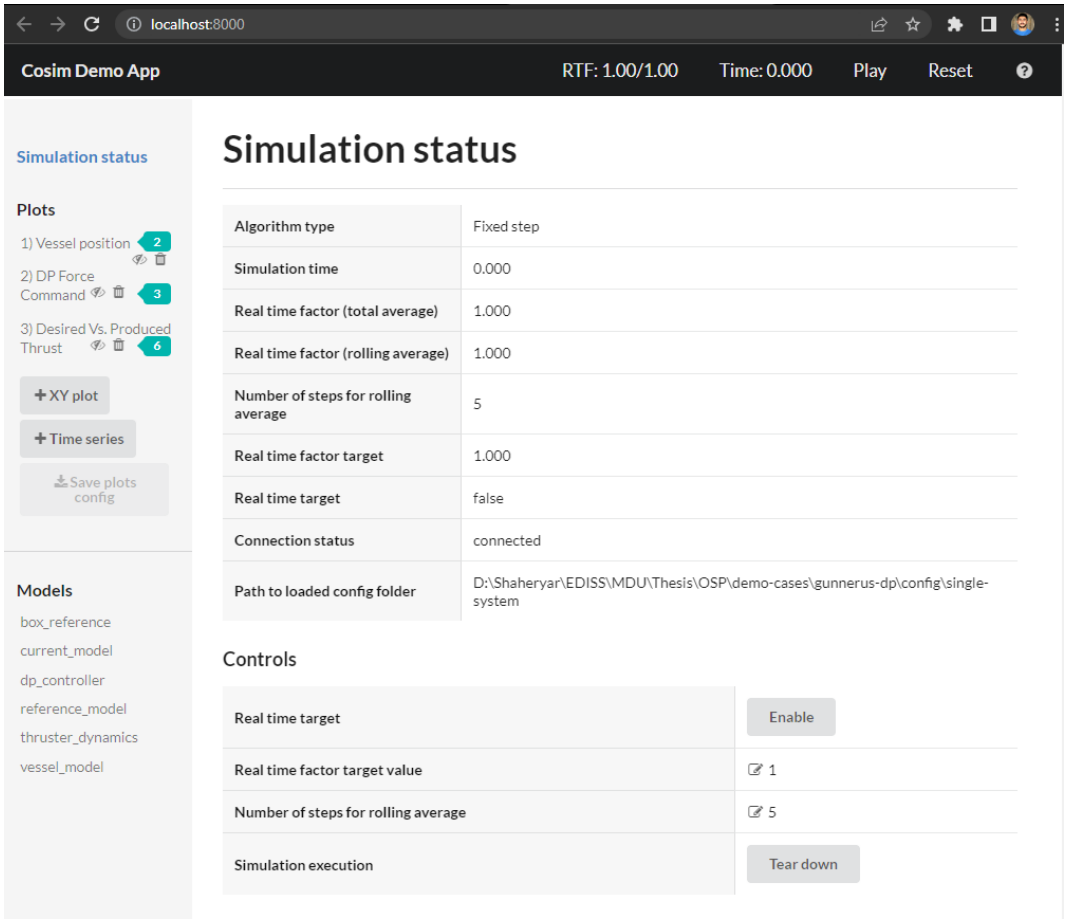

Figure 10: Cosim Demo Application

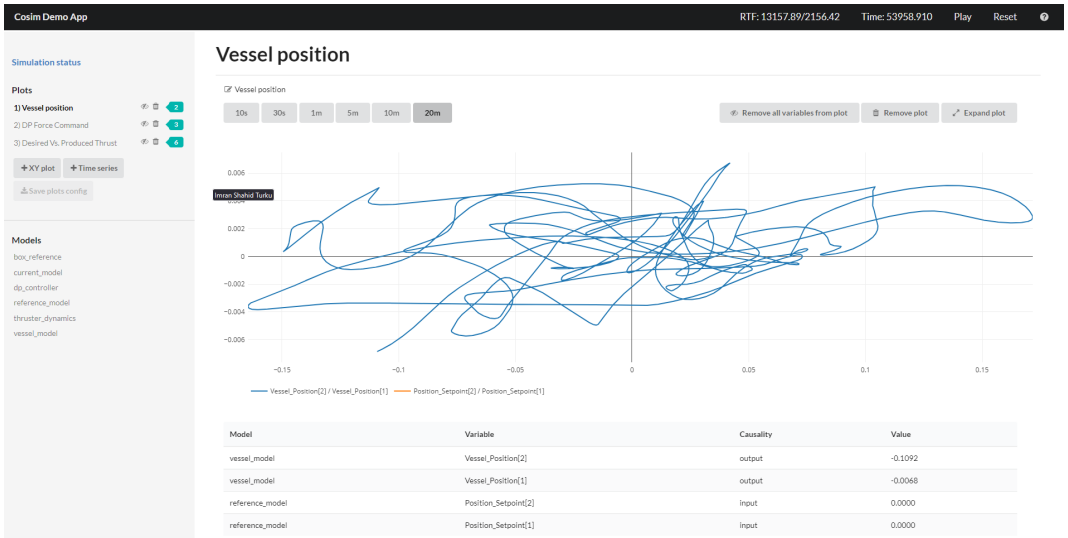

Figure 11: Vessel Position Graph

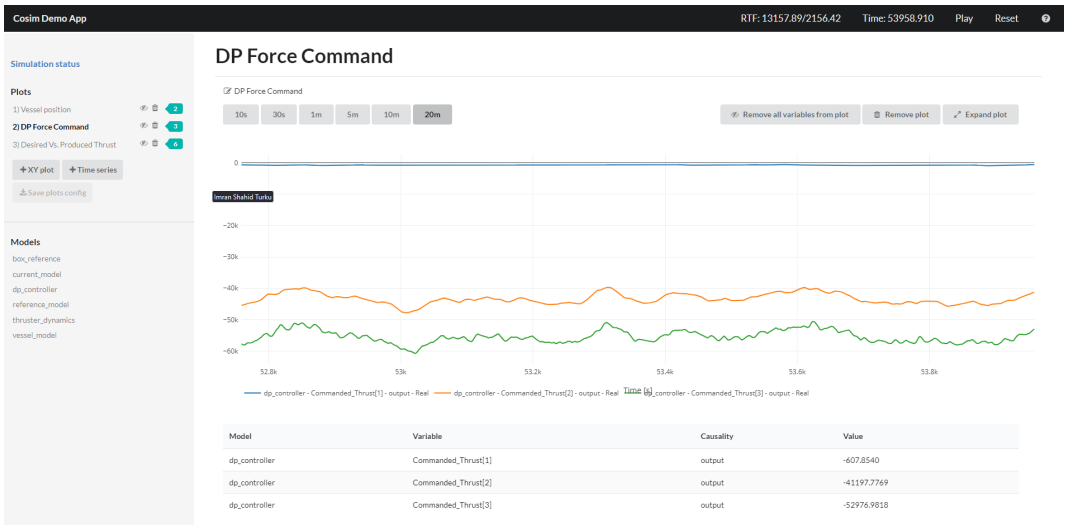

Figure 12: DP Force Command Graph

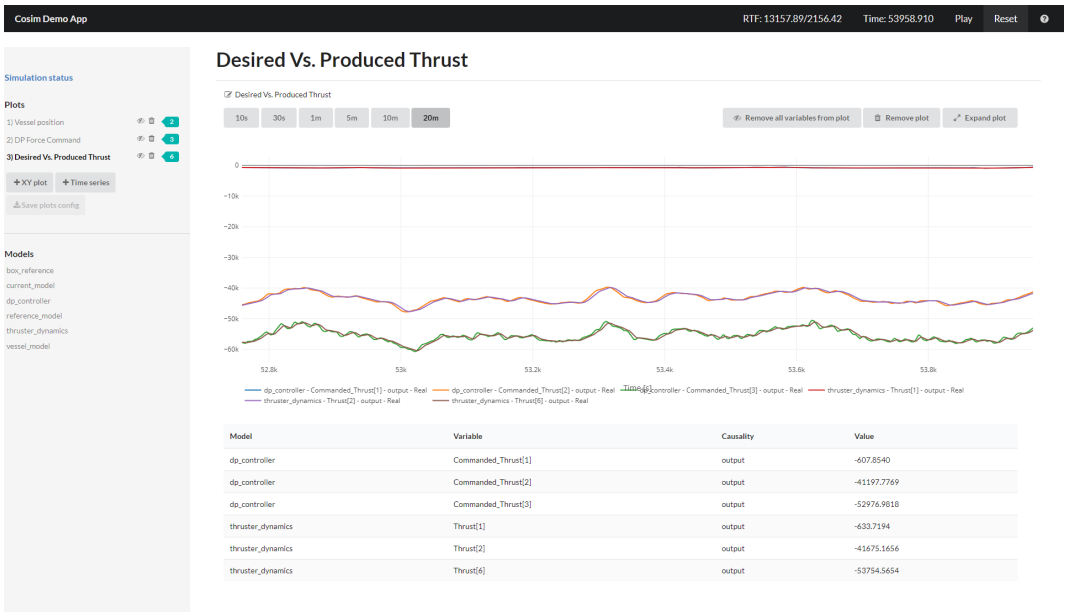

Figure 13: Desired Vs. Produced Thrust Graph

8. Results

We successfully implemented the entire end-to-end pipeline as shown in Figure 7. This enabled us to carefully examine the results generated by STGEM for our particular use case. Now, We'll look into how the parameters produced by STGEM were used, and then what results simulations produced using the OSP.

In Figure 14, we present a comprehensive view of the results for each test case, accompanied by the additional files generated by our code to facilitate the simulation process. This includes a detailed representation of all model parameters throughout the time range of the simulations, available within the results directory. This presentation, consisting of time series data, offers a comprehensive overview of parameter variations during the simulation process.

In Appendix Listing 4, we have shown STGEM output for the parameters i.e., current velocity in surge and current velocity in sway. We have named the parameters as current_velocity0 and current_velocity1. This Listing 4 shows the parameters for just one test case. It can been further explained.

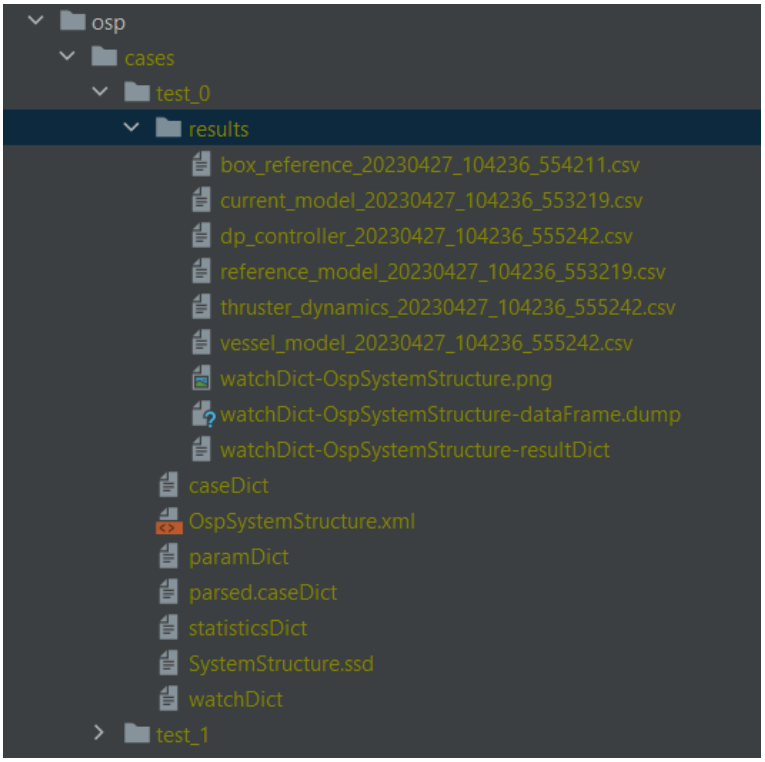

Figure 14: Simulation Results in STGEM Tool

8.1. Generated Parameters

This file contains several directives and parameters that are necessary for running simulations in the scenario that is described. The first part of the document presents numerical predictions for two different current velocities, denoted as "current_velocity0" and "current_velocity1."

A simulation case configuration starts when the "_case" block appears after that. This block includes a number of settings and properties relevant to the simulation situation. The case name is specified as "test_0" and relevant information is provided, including the related layer, level, index, path, and whether the case is regarded as a leaf node (i.e., "is_leaf" is set to true). There are no particular circumstances and there are no samples in the case, which is indicated to be zero.

Additionally, the "parameters" part of the case specifies "current_velocity0" and "current_velocity1" values that are in line with the original numerical values offered. The "commands" section lists the steps that were taken in order to run the simulation. As part of the preparation process, files

were copied, dictionaries were parsed, and the case must be built. The "run" command started the simulation itself, while the "post" function made it easier to carry out post-simulation tasks like data analysis.

The "status" field indicates the case's current state, which is indicated above as "CaseStatus.NONE," indicating that no particular status has yet been assigned. In essence, this configuration file contains the parameters, directives, and commands for preparation, execution, and post-processing that are necessary to set up and run a simulation scenario.

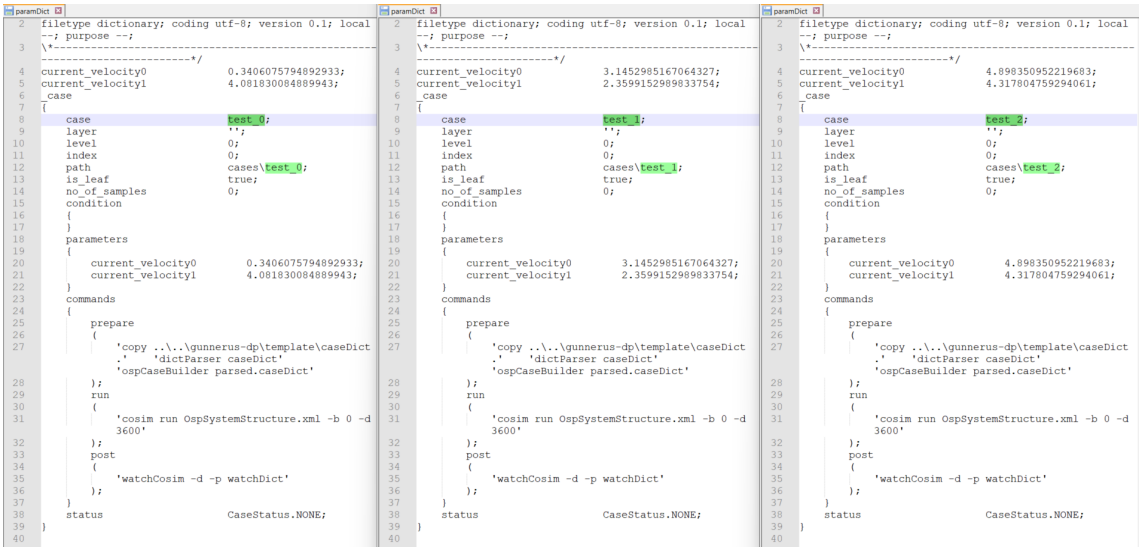

Figure 15: ParamDict file for first 3 test cases

Each test case generated by STGEM is given a number between 0 and 49, and these test cases are linked to input parameters that explore different variations in the test space. The initial paramDict files created for the OSP model are displayed in Figure 15. These three chosen test scenarios offer insightful information about the behavior of the model. Instead of methodically examining all potential combinations, they produce input values based on prior model outputs and include parameters within our preset input range of 0 to 5 for current velocities.

8.2. Description of Model Output

Table 4 shows Vessel Model output based on experiments ran for one of the test cases. It shows values of two Current Velocity parameters (surge and sway) for each time step over 2 seconds with 0.1 second interval. The table also shows how vessel position is affected based on these values. These results are generated based on our paramDict definition.

In addition, the table shows the simulation outcomes for test case 0 of a single paramDict file. These findings are important because they show how accurately and correctly the input models created for OSP were. These results point to potential improvements in the model's functionality in next test cases. Each test case attempts to assess the standard deviation of vessel position, advancing our work by assessing the performance and robustness of the system.

The main objectives of our research are substantially supported by this validation of our hypothesis, which also highlights the important part GAN-based algorithms play in the area of test case production. Notably, these algorithms show an exceptional capacity for independently navigating the system's complexities, successfully revealing its complexities. The GAN-based technique demonstrates its capacity to intelligently study the system by taking on this dynamic role, providing useful insights into how it might be carefully reviewed and appraised.

8.3. Results Interpretation

Now, let's delve into what we discovered through our research and how using GAN-based algorithms in OSP through the use of STGEM turned out.

Time	StepCount	Current_Velocity[1]	Current_Velocity[2]	Vessel_Position[1]	Vessel_Position[2]
0	0	0	0	0	0
0.1	1	0	0	0	0
0.2	2	0.00020793	2.10E-06	1.73E-07	6.00E-09
0.3	3	0.000669734	1.37E-05	1.73E-07	6.00E-09
0.4	4	0.000636791	1.95E-05	1.19E-06	7.80E-08
0.5	5	0.00178271	7.31E-05	1.19E-06	7.80E-08
0.6	6	0.00201601	0.000103587	4.87E-06	5.42E-07
0.7	7	0.00261188	0.000160097	4.87E-06	5.42E-07
0.8	8	0.00324442	0.000233111	1.44E-05	2.18E-06
0.9	9	0.00389288	0.000318484	1.44E-05	2.18E-06
1	10	0.00417425	0.000385124	3.24E-05	6.08E-06
1.1	11	0.00444243	0.000455187	3.24E-05	6.08E-06
1.2	12	0.00451093	0.000507568	6.08E-05	1.33E-05
1.3	13	0.00452141	0.000556077	6.08E-05	1.33E-05
1.4	14	0.00497944	0.000660987	0.000100502	2.49E-05
1.5	15	0.00464567	0.000665136	0.000100502	2.49E-05
1.6	16	0.00523064	0.000803904	0.000152235	4.18E-05
1.7	17	0.00517316	0.000846042	0.000152235	4.18E-05
1.8	18	0.00557919	0.000969861	0.000216609	6.52E-05
1.9	19	0.00590923	0.0010861	0.000216609	6.52E-05
2	20	0.00617258	0.00119434	0.000294501	9.62E-05

Table 4: Vessel Position based on Current Velocity

- **RQ1**: For creating test scenarios for maritime software systems within OSP, GANs (STGEM in our case) have shown to be helpful. They excel at deciphering complex system details and developing useful test cases.
- **RQ2**: For performance testing within OSP, we've discovered useful design and usage techniques for GANs i.e., using multiple GAN models for testing the system. We can develop reliable testing procedures with the use of GANs by understanding system dynamics. To put it another way, we addressed the Gunnerus-DP use case through the design and implementation of STGEM, which answers the initial research question.
- **RQ3**: GANs are good in predicting the performance of OSP systems. They are dependable because they design test cases where the system is more likely to fail rather than testing the system with brute force.
- **RQ4**: GANs are excellent for testing, but if we want to improve model training, they do require a lot of computational power and different data. Nevertheless, based on the Farn implementation, STGEM performed optimally without creating all potential test cases. In the later stages of this work, a qualitative comparison between Farn and STGEM was disregarded due to fundamental difference between these tools.

The main goal of this research project was to determine whether it is feasible to use Generative Adversarial Networks (GANs) to create useful test scenarios that are specifically adapted to validate maritime software systems integrated within the OSP co-simulation platform. We have implemented our approach to validate this goal. The study also explored the efficiency of GAN-based algorithm integration and use in the context of automating performance assessments. This inquiry is complemented by a thorough assessment of the precision and dependability of the forecasts made by the GAN model, which aims to simulate the functionality of complex systems. Additionally, a thorough examination is conducted to determine the benefits and drawbacks of using GANs for performance assessment, making interesting comparisons with other approaches.

Our research efforts have successfully used the capabilities of GAN-based algorithms to provide relevant test cases for the validation of maritime software systems running on the OSP platform. We have created and executed a refined technique based on previous research that not only improves our understanding of effective design inside the OSP environment but also aids in a more nuanced

understanding of system qualities via a thorough performance study. It is important to note that the GAN model has demonstrated an impressive degree of dependability and accuracy in the field of test case production. It performs better than other approaches like Farn in particular since it is inherently intelligent.

Although the test cases have shown their effectiveness, it is important to recognize that their practical implementation has encountered resource-intensive difficulties, notably in terms of the computational demands for running these models on top of simulations. The range of conceivable executions has necessarily been limited by this complexity. The complex nature of the systems under examination demands rigorous scalability optimization techniques. Furthermore, by examining additional characteristics in addition to vessel current velocity, the model's scope may be widened. This expansion might shed light on how the model reacts to changes in attributes other than vessel position, allowing for a more thorough evaluation of its prediction skills.

9. Conclusion and future work

The major goal of our study was to determine whether it would be feasible to design customized tests for evaluating marine software in the OSP co-simulation environment using generative adversarial networks (GANs). With an added focus on assessing how well GAN-based algorithms facilitate system performance reviews, this goal was successfully accomplished. The goal of the study was to investigate the GAN model's ability to mimic complex real-world system behaviors by measuring the predictability and accuracy of its predictions. Our inquiry was furthered by doing a close analysis on results generated by GAN-based alogorithm which is small subset of test parameters generated by traditional techniques.

Through the course of our investigation, GAN-based algorithms were successfully used to produce tests that aid in the assessment of maritime software integrated into the OSP system. Through careful testing, we improved an earlier strategy, deepening our understanding of efficient system design within OSP and enabling a more thorough performance analysis. Due to its inherent intelligence, the GAN model has shown amazing dependability and precision in test generation, demonstrating its superiority over competing approaches like Farn.

However, we ran into problems because of the computational requirements involved in producing these tests, which made it impossible to actually implement. The systems under investigation are inherently complex, which highlights the need for creative approaches to improve these techniques' scalability. Additionally, the GAN model's capacity to forecast a larger range of events could be realized by extending its range beyond vessel speed to include other variables. This line of inquiry offers hope for a more thorough assessment of the model's prediction ability in various scenarios.

From our research's findings, various directions for further work and improvements become clear. Another important step is to extend the focus of our research by including a wider variety of use cases from OSP reference models. This would offer a more comprehensive understanding of the applicability and efficacy of the GAN model across a range of scenarios, helping to establish a strong and adaptable testing framework. We may analyze the model's predicting skills more refined by taking into account various maritime software functionality and difficulties.

The production of all necessary parameters for models directly from the STGEM environment could represent a considerable advance. We can expedite the testing workflow and get rid of potential inconsistencies caused by parameter mismatches by smoothly integrating the parameter creation process. This improvement would boost productivity while also helping to depict real-world situations more accurately, which would strengthen the validity of tests carried out utilizing GAN-based algorithms.

The use of several STGEM models that are accessible for testing represents another intriguing direction for future research. We may investigate the flexibility and adaptability of the GAN-based method across a range of system topologies and functionalities by using a variety of models. This side-by-side comparison would highlight the advantages and disadvantages of the GAN model in various testing settings and offer insightful information for further optimization and improvement. Finally, these upcoming projects have the potential to expand both maritime software testing procedures and the use of GANs in system performance evaluation.

In conclusion, our investigation into the use of GAN-based test creation for marine software within the OSP configuration was successful. The effective application of this strategy produced insightful information on the model's capability to simulate complex system dynamics. Our study has paved the way for scalability enhancements and the strengthening of the GAN model's capabilities while tackling resource-intensive problems. The information gathered from this project will surely aid in the development of maritime software testing and present prospects for innovation and improvement as technology progresses.

References

[1] A. F. (Adam), "The importance of maritime transport for economic growth in the european union: A panel data analysis," *Sustainability*, 2021.

[2] OSP. (2022) Open simulation platform (osp). [Online]. Available: https://opensimulationplatform.com/

[3] D. open source. (2022) Farn. [Online]. Available: https://github.com/dnv-opensource/farn

[4] J. Peltomaki. (2022) Stgem: System testing using generative models. [Online]. Available: https://gitlab.abo.fi/stc/stgem

[5] P.-L. Sanchez-Gonzalez, D. Díaz-Gutiérrez, T. J. Leo, and L. R. Núñez-Rivas, "Toward digitalization of maritime transport?" *Sensors*, vol. 19, no. 4, 2019. [Online]. Available: https://www.mdpi.com/1424-8220/19/4/926

[6] M. Fruth and F. Teuteberg, "Digitization in maritime logistics—what is there and what is missing?" *Cogent Business & Management*, vol. 4, no. 1, p. 1411066, 2017. [Online]. Available: https://doi.org/10.1080/23311975.2017.1411066

[7] J. Lee, "Suez canal blockage: an analysis of legal impact, risks and liabilities to the global supply chain," *MATEC Web of Conferences*, 2021.

[8] D. Jones, C. Snider, A. Nassehi, J. Yon, and B. Hicks, "Characterising the digital twin: A systematic literature review," *CIRP Journal of Manufacturing Science and Technology*, vol. 29, pp. 36–52, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1755581720300110

[9] Í. A. Fonseca and H. M. Gaspar, "Challenges when creating a cohesive digital twin ship: a data modelling perspective," *Ship Technology Research*, vol. 68, no. 2, pp. 70–83, 2021.

[10] A. Coraddu, L. Oneto, F. Baldi, F. Cipollini, M. Atlar, and S. Savio, "Data-driven ship digital twin for estimating the speed loss caused by the marine fouling," *Ocean Engineering*, vol. 186, p. 106063, 2019.

[11] M. Schirmann, M. Collette, and J. Gose, "Ship motion and fatigue damage estimation via a digital twin," in *Life Cycle Analysis and Assessment in Civil Engineering: Towards an Integrated Vision*. CRC Press, 2018, pp. 2075–2082.

[12] C. Gomes, C. Thule, D. Broman, P. G. Larsen, and H. Vangheluwe, "Co-simulation: a survey," *ACM Computing Surveys (CSUR)*, vol. 51, no. 3, pp. 1–33, 2018.

[13] L. I. Hatledal, R. Skulstad, G. Li, A. Styve, and H. Zhang, "Co-simulation as a fundamental technology for twin ships," 2020.

[14] OSP. (2022) Osp reference model gunnerus-dp. [Online]. Available: https://open-simulation-platform.github.io/cosim-demo-app/Gunnerus-DP

[15] ——. (2022) Osp reference model gunnerus-dp fmu system description. [Online]. Available: https://open-simulation-platform.github.io/cosim-demo-app/Gunnerus-DP#system-description

[16] ——. (2022) Osp reference model gunnerus-dp fmu descriptions. [Online]. Available: https://open-simulation-platform.github.io/cosim-demo-app/Gunnerus-DP#fmu-descriptions

[17] J. Peltomaki and I. Porres, "Falsification of multiple requirements for cyber-physical systems using online generative adversarial networks and multi-armed bandits," in *2022 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, 2022, pp. 21–28.

[18] J. Peltomäki, F. Spencer, and I. Porres, "Wogan at the sbst 2022 cps tool competition," in *Proceedings of the 15th Workshop on Search-Based Software Testing*, 2022, pp. 53–54.

[19] ——, "Wasserstein generative adversarial networks for online test generation for cyber physical systems," in *Proceedings of the 15th Workshop on Search-Based Software Testing*, 2022, pp. 1–5.

[20] S. Skjong. (2017) Modeling and simulation of maritime systems and operations for virtual prototyping using co-simulations. [Online]. Available: https://ntnuopen.ntnu.no/ntnu-xmlui/handle/11250/2484964

[21] S. Skjong, M. Rindarøy, L. T. Kyllingstad, V. Æsøy, and E. Pedersen, "Virtual prototyping of maritime systems and operations: applications of distributed co-simulations," *Journal of Marine Science and Technology*, vol. 23, pp. 835–853, 2018.

[22] I. Porres, H. Rexha, and S. Lafond, "Online gans for automatic performance testing," in *2021 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*. IEEE, 2021, pp. 95–100.

10. Appendix

```python
import numpy as np
from stgem.sut import SUT, SUTOutput
from utils import *


class OSPSUT(SUT):
    """
    A SUT which encapsulates a OSP Scenario which we assume to take vectors
    as inputs and output vectors.
    """

    def __init__(self, parameters=None):
        super().__init__(parameters)

        # Current velocity variable ranges
        self.input_range = [[0, 5], [0, 5]]
        # Max distance b/w Desired_Position and Vessel_Position
        self.output_range = [[0.5, 3]]
        self.input_type = "vector"
        self.output_type = "vector"
        self.case_number = 0

    def _execute_test(self, test):
        # Get test case from SUTInput object and extract the required information
        current_velocity = self.descale(test.inputs.reshape(1, -1), self.
            input_range).reshape(-1)

        output = []
        error = None
        # Exception handler
        try:
            # Run OSP simulation
            sim_result = execute_test_case(self.case_number, current_velocity)
            output.append(sim_result)
            self.case_number = self.case_number + 1
        except Exception as err:
            error = err

        test.input_denormalized = current_velocity

        return SUTOutput(np.asarray(output), None, None, error)
```

Listing 1: "sut.py"

```python
import re
from farn.farn import execute_command_set, logger
from typing import Any
from numpy import dtype
from pandas import DataFrame, Series
from dictIO import DictReader
from dictIO.utils.path import relative_path
from farn.core import Cases
from farn.farn import create_param_dict_files
from farn.farn import create_case_folders
from pathlib import Path
from typing import Dict, List
from farn.core import Case, Parameter
from numpy.linalg import norm

command_sets: Dict[str, List[str]] = {
    "prepare": [
        # copy a (case-agnostic) ospx config file from a template directory into
            the case folder
        "copy ..\\..\\gunnerus-dp\\template\\caseDict .",
        # parse the ospx config file. This will make it case-specific
        "dictParser caseDict",
        # build the (case-specific) OspSystemStructure.xml
```

```
        "ospCaseBuilder parsed.caseDict",
    ],
    "run": [
        # run OSP cosim
        "cosim run OspSystemStructure.xml -b 0 -d 3600",
    ],
    "post": [
        # optional post-processing. watchCosim creates a sub-folder 'results' in
            the case folder
        "watchCosim -d -p watchDict",
    ],
}


def execute_test_case(case_name, case_parameters):
    global command_sets

    cases: Cases = Cases()
    case_name = "test_" + str(case_name)
    case_folder: Path = Path("./cases") / case_name

    case_parameters: List[Parameter] = [
        Parameter("current_velocity0", case_parameters[0]),
        Parameter("current_velocity1", case_parameters[1])
    ]

    case: Case = Case(
        case=case_name,
        path=case_folder,
        is_leaf=True,
        parameters=case_parameters,
        command_sets=command_sets,
    )
    cases.append(case)
    execute_osp_commands(cases)
    result = post_processing(cases)
    return result


def execute_osp_commands(cases):
    global command_sets

    _ = create_case_folders(cases)
    _ = create_param_dict_files(cases)

    _ = execute_command_set(
        cases=cases,
        command_set="prepare",
    )

    _ = execute_command_set(
        cases=cases,
        command_set="run",
    )

    _ = execute_command_set(
        cases=cases,
        command_set="post",
    )


def post_processing(cases):
    mapping: Dict[str, Dict[str, Any]] = {
        "Desired_Position": {  # column name you want to map a variable to
            "key": "reference_model|Desired_Position[1]:stdev",  # variable in FMU
            "unit": 1,  # usually 1, unless you want to apply a scaling factor
        },
        "Vessel_Position": {  # column name you want to map a variable to
            "key": "vessel_model|Vessel_Position[1]:stdev",  # variable in FMU
            "unit": 1,  # usually 1, unless you want to apply a scaling factor
```

```
        }
    }

    # column names
    names: List[str] = [name for name in mapping if not re.search("(^_|COMMENT)",
        name)]

    series: Dict[str, Series] = {
        "path": Series(data=None, dtype=dtype(str), name="path"),
    }

    for index in range(len(cases)):

        case = cases[index]
        case_folder: Path = case.path
        result_folder: Path = case_folder / "results"
        result_dict_file: Path = result_folder / "watchDict-OspSystemStructure-
            resultDict"  # Output of watchCosim

        series["path"].loc[index] = str(relative_path(Path.cwd(), case_folder))

        result_dict = DictReader.read(result_dict_file, includes=False, comments=
            False)

        for name in names:
            value: Any = None
            value_eval_string = "result_dict['" + "']['".join(mapping[name]["key"].
                split(":")) + "']"
            try:
                value = eval(value_eval_string)
            except Exception:
                logger.warning(f'"{value_eval_string}" not in {result_dict_file}')
                continue

            if name not in series:
                series[name] = Series(data=None, dtype=dtype(type(value)), name=
                    name)

            if value is not None:
                series[name].loc[index] = value

    df: DataFrame = DataFrame(data=series)

    # Using stdev as metric
    result = norm(df['Desired_Position'] - df['Vessel_Position'])
    return result
```

Listing 2: "utils.py"

```
import os
import unittest
from stgem.algorithm.ogan.algorithm import OGAN
from stgem.algorithm.ogan.model import OGAN_Model
from stgem.algorithm.random.algorithm import Random
from stgem.algorithm.random.model import Uniform
from stgem.generator import STGEM, Search
from stgem.objective import Minimize
from stgem.objective_selector import ObjectiveSelectorMAB
from sut import OSPSUT


class TestOSP(unittest.TestCase):
    def test_models(self):
        generator = STGEM(
            description="test_models",
            sut=OSPSUT(),
            objectives=[Minimize(selected=[0], scale=True)
                        ],
            objective_selector=ObjectiveSelectorMAB(warm_up=10),
            steps=[
```

```
                Search(budget_threshold={"executions": 50},
                       algorithm=Random(model=Uniform(parameters={"min_distance":
                           0.2}))),
                Search(budget_threshold={"executions": 50},
                       algorithm=OGAN(model=OGAN_Model()))
            ]
        )
        r = generator.run()
        file_name = "osp_results.pickle"
        r.dump_to_file(file_name)
        os.remove(file_name)
```

Listing 3: "test_osp.py"

```
/*--------------------------------*- C++ -*----------------------------------*\
filetype dictionary; coding utf-8; version 0.1; local --; purpose --;
\*---------------------------------------------------------------------------*/
current_velocity0             0.3406075794892933;
current_velocity1             4.081830084889943;
_case
{
    case                      test_0;
    layer                     '';
    level                     0;
    index                     0;
    path                      cases\test_0;
    is_leaf                   true;
    no_of_samples             0;
    condition
    {
    }
    parameters
    {
        current_velocity0         0.3406075794892933;
        current_velocity1         4.081830084889943;
    }
    commands
    {
        prepare
        (
            'copy ..\..\gunnerus-dp\template\caseDict .'     'dictParser caseDict'
                'ospCaseBuilder parsed.caseDict'
        );
        run
        (
            'cosim run OspSystemStructure.xml -b 0 -d 3600'
        );
        post
        (
            'watchCosim -d -p watchDict'
        );
    }
    status                    CaseStatus.NONE;
}
```

Listing 4: "Parameters Dictionary File - paramDict"