Hybrid Simulation Framework: Advanced and Secure Training

Lucas S. dos Santos[1,2], **Luiz C. G. Maria**[2],
Erisvaldo F. M. Junior[2] **e Raphael C. S. Machado**[1]

[1]Institute of Computing – Fluminense Federal University (UFF)
Niterói – RJ – Brazil

[2]SENAI Inovation Institute of Virtual Production Systems
(ISI SVP) – Benfica, RJ – Brazil

`{lseveriano,raphaelmachado}@ic.uff.br, {lgmaria,efjunior}@firjan.com.br`

Abstract. *Simulation has been employed as a teaching, training, and procedure validation tool in several fields, such as medicine, education, and the military. Although the use of Simulation presents advantages compared to traditional methodologies, several challenges exist in designing, developing, and implementing these simulation systems. These challenges become even more complex when the project involves the development of hybrid simulators. These simulators integrate physical and virtual systems. This integration makes these simulators more immersive, enhancing the simulation experience. The present work describes a framework for developing hybrid simulator systems. The framework was applied in the "Tupi" Class Submarines retrofit simulator project. The results show that the framework is a feasible solution for hybrid simulation systems, validated through collaboration with the Brazilian Navy's technical team (CIAMA).*

1. Introduction

The advances in Information and Communication Technology (ICT) have enabled the development of devices and systems capable of simulating real environments. Simulation is a method or technique for reproducing real scenarios, systems, and procedures in controlled and guided environments. Simulation involves generating an artificial log of the system, and observing and analyzing it allows the user to make inferences about the operational characteristics of the represented real system. Thus, simulation can capture the essential characteristics of the simulated environment to teach new skills, experiment with different approaches, and explore the implications of decision-making in a risk-free environment [Mentzelopoulos et al. 2011, Moorthy et al. 2005]. In other words, simulation reduces cost, increases safety, and allows visibility of events and reproducibility of operations. Given these features, simulation can be applied in various scenarios with different approaches, such as network protocol algorithm evaluation [Mohapatra and Kanungo 2012], Computational Fluid Dynamics (CFD) [Milne-Thomson 1973], Car Aerodynamics Design [Kamal et al. 2021], and Crash Tests [Ambati et al. 2012].

In the military context, simulation techniques have played a crucial role in training tactical operations and instructing technical actions involving critical environments. In this perspective, simulators can act in several military fields, such as flights through unknown sites, submarine navigation under critical ocean conditions, and driving of armed armored vehicles [SENAI 2022, Jr 2022]. Thus, simulation supports developing operational and tactical experiences, reaction time, focus, attention, and motor skills in equipment handling. Although simulation presents advantages compared to traditional methodologies, several inherent challenges exist in designing, developing, and implementing these systems. These challenges become even more complex when the project applies the development of hybrid simulators. A Hybrid Simulator contains analog and virtual elements while using real instruments and equipment from the simulated object. It includes the emulation of procedures in a virtual manner [Eldabi et al. 2018].

Among the challenges, we highlight the complexity of integrating physical and virtual systems, efficient network communication techniques with a minimum response time, time synchronization, testability, and maintenance. Therefore, the lack of technical documentation that describes the framework for developing hybrid simulators, combined with the complexity of development and implementation, makes the development of hybrid simulators a complex and decisive task in terms of project execution and fulfillment.

This work presents a framework for developing hybrid simulators. The framework describes the architecture for implementing a simulation system that integrates virtual and physical systems. The framework has two main systems: the Instructor Operating Station (IOS) and the Simulation Module (SM). The Instructor Operating Station provides all the necessary infrastructure to support the simulation. This infrastructure consists of classes and interfaces that contain the responsibilities and manage the flow of calls to the Simulation Module. On the other hand, the Simulation Module contains all the business rules of the simulation, the mathematical model of the simulation dynamics, interface management, and communication with the physical systems. For the validation purpose, we implemented the framework in a real-world case study: retrofit project of a Submarines Simulator for "Tupi" Class. The project was conducted by the SENAI Institute of Innovation of Virtual System Production in partnership with the SENAI Institute of Technology of Industrial Automation for the Brazilian Navy, Admiral Áttila Monteiro Aché Instruction and Training Center (CIAMA). The results demonstrate that framework is a viable solution for developing and implementing hybrid simulation systems.

We organize this paper as follows. After this introduction, Section 2 presents the related works. Section 3 brings the background information that is necessary for the proper understanding of this work. Section 4 proposes our framework and explain how its architecture. We implemented the framework at the Section 5. We present the results of the implementation at the Section 6. Finally, Section 7 presents our conclusions and gives directions about open challenges and future works.

2. Related Works

There are several studies in the literature that presents the use of simulation techniques. Yeong-cheol's [Kim and Ahn 2010] presents the design and implementation of an operator procedure training simulator for Korean vertical launch anti-submarine missile systems. The training simulator provides interaction between simulated and real equipment (Fire Control System). The authors proposed a development framework that grouped the components into physical and logical packages to achieve this goal. The simulator contains modules as a computer graphics system, a virtual reality visualization system, an operating table, a weapons control system, and an audio system.

Zheng [Zheng et al. 2009] describes the development process of a flight simulator and proposes a software and hardware architecture. The proposed software architecture includes the Hardware Module, the Behavior Module, and the Software Decision Module. The Hardware Module provides an interface, such as hardware devices, to initialize and run the software. The Behavior Module works as an intermediary agent between subsystems and simulates the main function of the aircraft system. Finally, the Software Decision Module contains the implementation of the simulated aircraft aerodynamics. Lin [Lin et al. 1998] presents the Submarine Voyage Training Simulator. The simulator has six degrees of freedom motion on a flexible platform and generates navigation information during the simulation. The paper describes the system architecture and the motion and navigation system modeling.

Nan [Nan and Liang 2018] presents the SubSafe system as a game-based submarine safety training system developed by the UK, which provides end users with an interactive and real-time 3D model of a Trafalgar Class SSN submarine. The paper describes the history and main functions implemented in SubSafe, as well as an evaluation focused on the effectiveness of the training.

Sol Ha [Ha et al. 2012] introduces a High-Level Architecture (HLA) interface designed to integrate a combined discrete event and discrete time simulation model into a distributed simulation environment. The combined model, comprising discrete event and discrete time components, is considered a standard formalism in modeling and simulation. The HLA interface consists of an “Interface Model,”a “Model Translator”, and an “HLA/RTI Translator”. The Interface Model is a connection point for combined models without requiring modifications. The Model Translator facilitates translating data from the Interface Model into HLA/RTI functions, while the HLA/RTI Translator performs the reverse translation. A distributed simulation of a diving submarine validates the HLA interface, demonstrating consistent results.

3. Simulation Definition

Simulation is the process of designing and reproducing a model of a real system, coupled with conducting experiments on that model, to understand the system’s behavior and evaluate its operational procedures. Simulation is an effective tool for understanding complex operations and systems. Therefore, the model must be designed to replicate the behavior of the real system and the events that may occur throughout the simulation [Shannon 1992]. Simulation involves generating a log of system operations and events, which enables the analysis and study of that simulation. Additionally, according to Pritsker [Banks 1999], simulation can be considered as a process of constructing a mathematical model of a system under study to conduct experiments on that model. Simulation is employed in various contexts, such as technology for performance or optimization, safety engineering, testing, training, education, and games.

In the context of this work, we extend the simulation definition into four categories, described bellow:

- **Virtual simulation:** this simulation’s category involves using graphical computational models to simulate the operation of real systems. It is widely used in various fields, such as engineering, science, personnel training, and entertainment. It can be a 2D or 3D graphical system, and Virtual Reality. For example, aircraft flight, Medical, and Construction are examples of virtual simulation.

- **Hybrid simulation:** combines elements of virtual simulation with hardware-based or real system-based simulation. It is used when a closer interaction between a virtual system and a real system is required. For example, in the automotive industry, hybrid simulation may involve combining a computational model of a vehicle with a physical prototype to test control and safety systems.
- **Analytical simulation:** it is a computational technique used to study and analyze complex systems by creating mathematical models. These mathematical models are designed to represent the behavior and interactions of various components or variables within the system. The primary goal of analytical simulation is to provide a deep understanding of the system's behavior, performance, and characteristics, as well as to predict how it will respond to different inputs or scenarios.
- **Process simulation:** it is used to study and optimize operations in production systems, such as factories, industrial facilities, supply chain, and processes in general. It involves modeling all the process steps, from raw material input to the final product output. This representation typically includes various elements, such as equipment, materials, procedures, and environmental factors, to mimic the entire sequence of events within the process. In summary, process simulation improves the efficiency, safety, and quality of operations.

4. Framework Proposal

In this section, we introduce a framework for developing Hybrid Simulators. This framework describes a crucial structure for developing, implementing, and deploying a solution for a hybrid simulator. In the following, we present the key elements of the framework.

4.1. Instructor Operating Station

The IOS is the module of the Framework responsible for providing the foundation of the software that the simulator will contain. This module groups the resources necessary for managing the simulation software operations. Thus, basic management operations such as system initialization, user management, report generation, simulation training control, database connection, and graphical interface projection. The Instructor Operating Station, primarily calls the Simulation Module, which contains all the business rules related to simulation operations and dynamics. The software structure was developed based on layered architecture. In this architectural model, the software is decomposed into groups of modules, where each group integrates a responsibility and a level of abstraction [Belle et al. 2013]. Figure 1 shows, at a high level of abstraction, the software architecture designed for the scope of the Software System module.

The *Presentation Layer* is located at the top of the hierarchy, and its main function is to provide an interface that translates the tasks the user requires into a visually understandable response. The *Application Layer* is an intermediate layer that coordinates all the business logic that the application performs. This business logic is formed by a set of rules obtained through the organization's requirements and serves as guidelines for the operation of the entire application. The *Service Layer* is responsible for receiving actions from the Application Layer, processing them, and returning their respective results. This layer

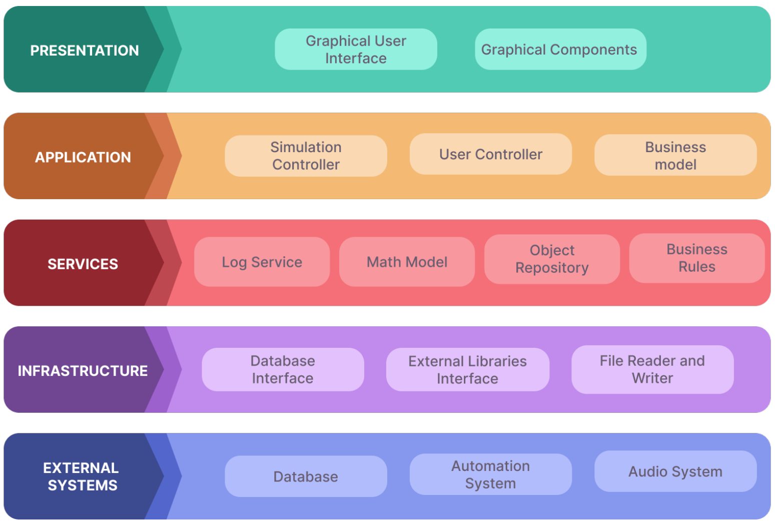

Figura 1. Software System Architecture

contains the domain rules, so it is responsible for executing the specific actions of each service. The *Infrastructure Layer* allows the Instructor Operating Station to interact with hardware and software elements external to the system's scope. This interaction occurs through receiving, storing, and providing data when requested. These elements can refer to communication with external libraries, File Reader and Writer Operations, or even the Database System. Thus, the purpose of this layer is to encapsulate all external components and group them for access by the Service Layer. The *External Layer* represents the resources external to the context of the framework.

4.2. Simulation Module

The Simulation Module consists of a self-contained project that supports all responsibilities and business rules related to the simulation process. These rules include implementing mathematical modeling, communication with the Automation System, and managing the Audio System. The entire Simulation can be reproduced independently of the Instructor Operating Station. This characteristic allows simulator designers and developers to Verify, Validate, and Test (VV&T) issues related to simulation time optimization, communication response time with the physical implementation of the simulator (TCP/IP), and implementation of mathematical modeling dynamics, for example. The main advantage of conducting tests in an isolated manner lies in reducing the overhead caused by the Instructor Operating Station module, which includes other technologies that consume computational resources (network bandwidth, processor, and memory consumption, for instance).

The Simulation Module provides a software infrastructure for mathematical modeling, communication with the Automation System, and audio management. Figure 2 shows the proposed architecture for the Simulation Module. The Math Model component contains all the business rules and implementation of the mathematical modeling related to the simulation's dynamics. In this sense, all the logic of the modeling process, which involves the code structure, can be reused regardless of the business rule. This component receives information (inputs) from the External System (Client) and the Automation System (through read operations). The Math Model gives such information through the

DataInput component. In other words, as the Simulation runs, all updated information is passed through a data structure implemented in DataOutput. In this sense, the DataInput and DataOutput components are defined as a fundamental data structure that reflects the state of the Simulation. Moreover, these structures are constantly updated to function as an Application Programming Interface (API) that allows external systems to consume information and make appropriate requests to the Simulation Module.

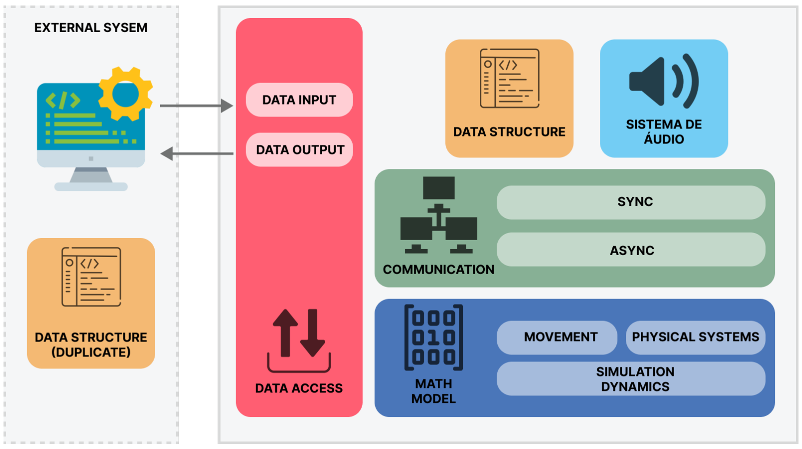

Figura 2. Simulation Module Architecture

The Automation System is an external system that provides complete support for the configuration, programming, commissioning, and diagnosis of the physical equipment contained in the simulator. Communication between the physical equipment typically occurs using the Ethernet/IP communication protocol. At the end of the simulation step (iteration), after the entire process of simulation dynamics, the Simulation Module sends the updated states to the outputs of the Automation System. This information includes motor positions, visual and sound signals, for example. Communication between the Simulation Module and the Automation System is performed through the Automation Communication component. Communication in this module can be implemented in two ways: *Synchronous* and *Asynchronous*. In *Synchronous* Communication, data flows between the Simulation Module and the Automation System at each iteration of the simulation loop, meaning communication occurs in real time. On the other hand, Asynchronous Communication occurs on demand. In other words, data transmission occurs only when there is an update in the states of the instruments in the Automation System.

4.3. The interaction between the Instructor Station and the Simulation Module

The diagram (Figure 3) shows the components of the Instructor Operating Station and how they interact with internal and external systems. It contains a graphical interface available to the user in this communication scheme. In this context, controllers propagate actions to the services that encapsulate the External Systems to fulfill the requests. The Instructor Operating Station contains all the infrastructure for managing the Database System, which allows CRUD[1] operations of relevant system information such as users, logs, and exercise data. The System is responsible for managing the requests to the Simulation Module. These requests are made in a repeating cycle defined by the service layer.

[1]Create, Read, Update and Delete operations.

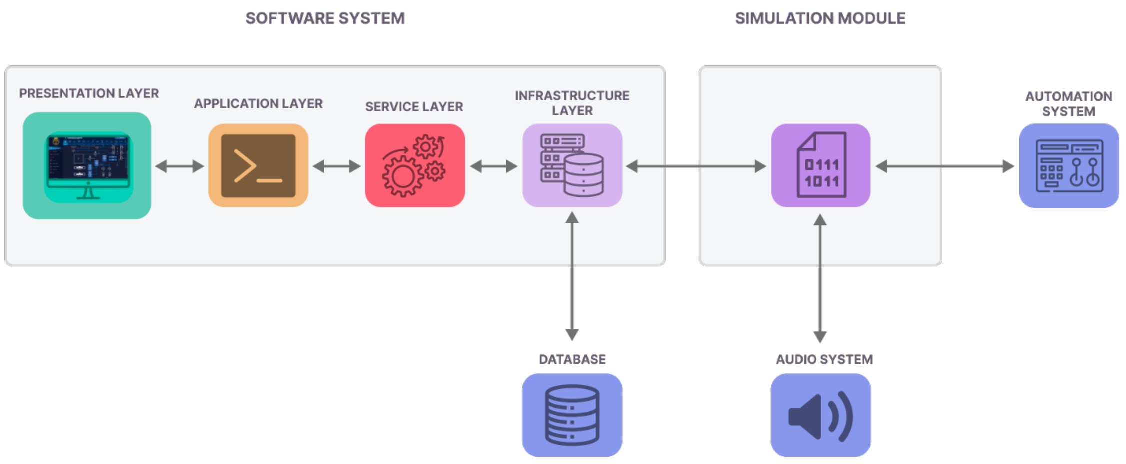

Figura 3. Interaction between System Modules

5. Implementation

This section describes the implementation of the framework as a solution for the submarine simulator project for the Brazilian Navy.

5.1. Project summary

In 2020, both of Firjan SENAI Institute's, IST AI and ISI SVP, partnered with the Admiral Áttila Monteiro Aché Instruction and Training Center (CIAMA) to develop the retrofit project for the Immersion Trainer Simulator for "Tupi" class Submarines (IT). The main objective of the project was to modernize the entire electronic and mechanical infrastructure of the simulator and develop a new operating system for Immersion Training, resulting in a more reliable and realistic simulator [Barros 2022].

The IT is a submarine simulator developed in the 90s for CIAMA to train crew members. The simulator went through development by various industry companies. Over time, relevant information, such as the system architecture, software design, and documentation, became obsolete. Further, the physical infrastructure of the simulator, which includes electrical, electronic, and mechanical components, has become antiquated.

The project presented several challenges related to physical infrastructure and software architecture. Regarding the physical infrastructure, it was necessary to remodel the simulator and implement, through methods such as 3D printing, parts that were no longer available for purchase in the market. Regarding the virtual System, new software for the instructor station was required. Additionally, the primary challenge lies in developing the mathematical model that encompasses the entire dynamic reproduction of the submarine.

5.2. Instructor Operating Station

The Instructor Operating Station was developed following the principles described in Section 4. For the project implementation, the C# programming language and the Windows Presentation Foundation (WPF) Framework were selected. The C# language offers various features that make the development process faster, including a statically typed and easily understandable language and, most importantly, a vast library with high-level functionalities. Additionally, C# is an object-oriented programming (OOP) language, which makes it highly efficient, flexible, scalable, and easy to maintain. The WPF framework provides a range of features for building the graphical interface. Moreover, it offers an architecture that separates the code handling the visual style of the interface (XAML) from the code implementing the behavior of the interface. These characteristics made WPF efficient for implementing the graphical interface of the simulator. The simulator architecture has a dedicated computer to run the Instructor Operating Station and the Simulation Module. The computer is connected to the simulator network, which remains connected to the Automation System and the Audio System.

Figure 5 shows the Instructor Operatiing Station (IOS) software installed on the panel in the instructor's cabin.

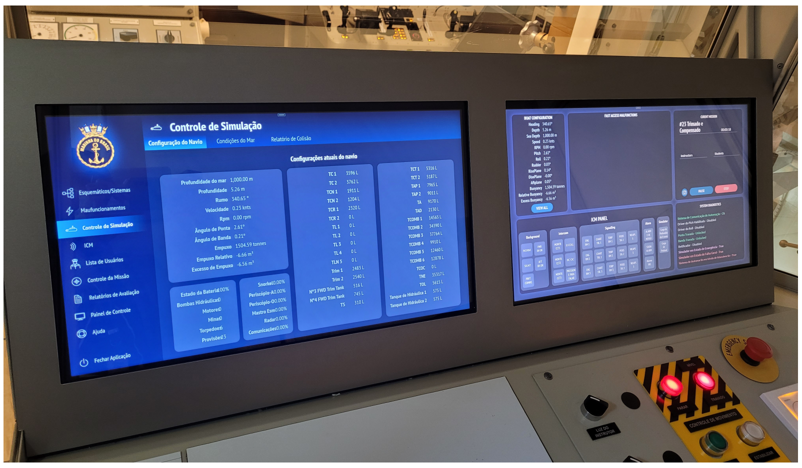

Figura 4. Instructor Operating Station

5.3. Simulation Module

The Simulation Module, as described in Section 4 is responsible for running all simulation-related dynamics, including reproducing the mathematical modeling of the submarine and carrying out all communication management with the Automation System and the management of the Audio System.

The C++ programming language was selected to develop the Simulation Module project. The C++ programming language performs better, that is, fewer computational cycles and, consequently, better energy consumption [Pereira et al. 2017]. C++ also presents greater efficiency related to the treatment of numerical problems. Regarding development, the selected language presents flexibility in implementation due to its multiparadigm character (functional or object-oriented). In addition, the language offers the ability to integrate with C#-based systems through dynamic-link libraries (DLLs).

5.4. Communication with the Automation System

The communication between the Simulation Module and the Automation System is carried out through OPC Classic protocol (Open Platform Communications) above the Ethernet/IP protocol. The OPC protocol defines a series of specifications and standards for communication between Computers, HMIs (Human Machine Interface), and Programmable Logic Controllers, the PLCs.

The OPC protocol utilizes the Distributed Component Object Model (DCOM) for communication between distributed components on the network. Data exchange occurs via OPC Data Access (OPC DA), allowing time synchronization and sending/receiving values from PLC tags. An OPC tag is a symbolic name that can be visible to OPC clients. The tag is a single data point and can be read or written. In this project, tags are used to represent both reader instruments (such as joysticks and valves) and writer instruments (such as smoke machines and light panels). In the specification of this project, tags are classified into analog and digital categories. Analog tags represent instruments/sensors with float values, such as tank volume, rudder, and valve positions. Digital tags represent discrete values (boolean/int), such as panels' on/off status.

Communication with the Automation System was implemented both synchronously and asynchronously, as specified in Section 4. Synchronous communication is responsible for writing tags (digital and analog). The instructions that send data via the network are blocking; that is, at each instruction, there is an interruption in the execution flow of the simulation. This factor generates a negative impact on the simulation time. A parallelism mechanism (thread) was implemented to separate the process of sending information from the main flow in order to reduce the negative impact. It was used for reading tags (digital and analog) for synchronous communication.

6. Evaluation and Results

The proposal framework was evaluated based on the simulator requirements and experter trainers' perspectives. In this sense, the evaluation aiming to validate its applicability in a real environment. In the following, we describe the requirements and their analysis.

Figura 5. Simulator.

6.1. Requirements Test cases

FR01 - Start the system in an automated manner: The system was designed to allow the automated startup of the components involving the Instructor Station software and the Simulation Model. Tests performed in a production environment (simulator) alongside the key actors (Trainers). The tests indicated that the system's startup process is functioning correctly when various system components are involved. Therefore, the test execution showed that the system satisfactorily met the requirement. Additionally, the operators did not show difficulties regarding operations related to system startup.

FR02 - Information about simulator state: The IOS software and SM were developed to provide real-time information about the state of the simulator. This information is processed and communicated to the automation system, updating the visualization of equipment and instruments, through simulation parameters, such as the submarine's pitch and roll angles, which influence the simulator's movement. The information presented in this panel provides the necessary parameters, including safety ones, to start the simulation with all the functionalities of the Instructor Operating Station and the Automation System. Tests indicated that the system can display real-time information from the simulator. In this sense, as changes occur in the physical state of the simulator (Automation System), the information in the "Diagnostic Recording Interface"panel is updated for the user.

FR03 - Simulation Management: The tests showed that the operators could create new simulation exercises by configuring required information such as sea state, type of seabed, malfunctions, and instructors and students assigned to the exercise. The operators were capable of making edits and deletions to the existing exercises. Users could easily handle the Instructor Station software. During exercises, trainers were able to modify the simulation by introducing malfunctions, changing sea conditions, and altering tank volumes for example. In addition to the manual and training provided to the CIAMA team, the system presented a user-friendly and easily understandable interface.

FR04 - Communication system: During the tests, the system demonstrated a high capacity for faithfully reproducing the real sounds that occur within an actual submarine. The audio validation was performed alongside trainers who were experts in submarine operations. The tests also included the validation and adjustment of audio volume and position. The audio system adopted by the Retrofit project utilizes a Surround 5.1 audio system, allowing audio to be played in different positions. The tests showed that the system could manage audio playback in specific channels. The ability to dynamically reproduce and adjust audio was also confirmed. The tests revealed that operators did not encounter difficulties performing audio-related operations, which confirmed the fidelity of the audio events in the simulation.

6.2. Comparison with the previous version

The software system update brought the implementation of a modern and modular architecture. This architecture separated the system into two main modules: the Instructor Operating Station and the Simulation Module. This separation allows for better project maintainability since they can be managed isolated. Furthermore, the technologies used in the project development provide an extended lifespan. It can be achieved by using robust technologies available in the marketplace.

The system implementation shifted the design from a distributed layout to a centralized one. This new computer organization makes the project's implementation easier to understand and maintain.

The study of the previous version of the project was thoroughly examined and analyzed, serving as a foundation for the new implementation. This in-depth study led to the incorporation of previously missing features, such as some malfunctions. Several malfunctions were not available for the instructor in the previous version.

6.3. Feedback from expert trainers

In the following, we list the collected feedback from the trainers to improve the quality of the simulation in the future works:

- Improve the response time between the Control Console System (CONGOP) and the Simulation System. There is a delay in operation that involves interaction between hardware and software. The delay negatively impacted trainer experience despite using async methods and parallel communication due to the network technology and techniques being used. It is necessary to use a lightweight communication protocol due to the numerous variables involved in the simulation process.
- Implement a media player system capable of adding audio in a modular way. This way, the trainer can add or change audio files as needed.

7. Conclusions and Future Works

We presented a framework to guide the development of systems for hybrid simulators. It describes the fundamental architecture to implement a hybrid simulation system. The framework comprises two main modules: the Instructor Operating Station and the Simulation Module. For the validation, a real-world case study was applied: the retrofit project of the Immersion Trainer for "Tupi"Class Submarines. The simulator was developed by the SENAI Institute of Innovation of Virtual Production System in partnership with the SENAI Institute of Technology of Industrial Automation for the Brazilian Navy (CIAMA). The simulator takes input from the instructor's panel data and automation system instruments, allowing the instructor to create exercises with configurable and parameterized operations. The developed system simulates the physical behavior of the submarine based on the model and scales extracted from the real system. Additionally, the instructor has complete control over the simulation through an intuitive graphical control panel. The results demonstrate that the framework is a viable solution for developing and implementing hybrid simulation systems. These results were validated based on the functional requirements established in collaboration with the technical team at Brazilian Navy (CIAMA). In future work, we intend to evaluate the framework's performance regarding simulation time and network response time with the Automation System.

Referências

Ambati, T., Srikanth, K., and Veeraraju, P. (2012). Simulation of vehicular frontal crashtest. *International Journal of Applied Research in Mechanical Engineering (IJARME)*, 2(1):37–42.

Banks, J. (1999). Introduction to simulation. In *Proceedings of the 31st conference on Winter simulation: Simulation—a bridge to the future-Volume 1*, pages 7–13.

Barros, M. (2022). Marinha/ firjan senai concluem o retrofit do treinador de imersão. Last accessed 03 november 2022.

Belle, A. B., El-Boussaidi, G., Desrosiers, C., and Mili, H. (2013). The layered architecture revisited: Is it an optimization problem? In *SEKE*, pages 344–349.

Eldabi, T., Brailsford, S., Djanatliev, A., Kunc, M., Mustafee, N., and Osorio, A. F. (2018). Hybrid simulation challenges and opportunities: a life-cycle approach. In *2018 Winter Simulation Conference (WSC)*, pages 1500–1514. IEEE.

Ha, S., Cha, J.-H., Roh, M.-I., and Lee, K.-Y. (2012). Implementation of the submarine diving simulation in a distributed environment. *International Journal of Naval Architecture and Ocean Engineering*, 4(3):211–227.

Jr, P. R. B. (2022). Exército e firjan vão desenvolver simulador para o guarani. Last accessed 30 december 2022.

Kamal, M. N. F., Ishak, I. A., Darlis, N., Maji, D. S. B., Sukiman, S. L., Abd Rashid, R., and Azizul, M. A. (2021). A review of aerodynamics influence on various car model geometry through cfd techniques. *Journal of Advanced Research in Fluid Mechanics and Thermal Sciences*, 88(1):109–125.

Kim, Y.-c. and Ahn, C. (2010). Development of hybrid anti-submarine weapon training simulator using component-based development methodology. *SISO*.

Lin, Z., Feng, S., and Ying, L. (1998). The design of a submarine voyage training simulator. In *SMC'98 Conference Proceedings. 1998 IEEE International Conference on Systems, Man, and Cybernetics (Cat. No. 98CH36218)*, volume 4, pages 3720–3724. IEEE.

Mentzelopoulos, M., Tanasa, M., Protopsaltis, A., and Economou, D. (2011). Explosive ordinance disposal: motion sensor simulator in nintendo wii. In *Proceedings of the 29th ACM international conference on Design of communication*, pages 227–234.

Milne-Thomson, L. M. (1973). *Theoretical aerodynamics*. Courier Corporation.

Mohapatra, S. and Kanungo, P. (2012). Performance analysis of aodv, dsr, olsr and dsdv routing protocols using ns2 simulator. *Procedia Engineering*, 30:69–76.

Moorthy, K., Vincent, C., and Darzi, A. (2005). Simulation based training.

Nan, Q. and Liang, M. (2018). Subsafe–a game-based training system for submarine safety. In *2018 Joint International Advanced Engineering and Technology Research Conference (JIAET 2018)*, pages 161–166. Atlantis Press.

Pereira, R., Couto, M., Ribeiro, F., Rua, R., Cunha, J., Fernandes, J. P., and Saraiva, J. (2017). Energy efficiency across programming languages: how do energy, time, and memory relate? In *Proceedings of the 10th ACM SIGPLAN International Conference on Software Language Engineering*, pages 256–267.

SENAI (2022). Firjan senai vai desenvolver simuladores para o exército, fortalecendo a base industrial de defesa do brasil. Last accessed 30 december 2022.

Shannon, R. E. (1992). Introduction to simulation. In *Proceedings of the 24th conference on Winter simulation*, pages 65–73.

Zheng, S., Huang, Q., Jin, J., and Han, J. (2009). Flight simulator architecture development and implementation. In *2009 International Conference on Measuring Technology and Mechatronics Automation*, volume 2, pages 230–233. IEEE.