Enhancing Simulation Capabilities Through the Integration of Discrete Event Simulation and Virtual Reality

Joseph Jabbour, Jalal Possik, Charles Yaacoub, Sina Namaki Araghi, Simon Gorecki, Grégory Zacharewicz, Adriano Solis

23rd International Conference on Modelling and Applied Simulation, 018
21st International Multidisciplinary Modeling & Simulation Multiconference

2724-0037 © 2024 The Authors.
doi: 10.46354/i3m.2024.mas.018

Enhancing Simulation Capabilities Through the Integration of Discrete Event Simulation and Virtual Reality

Joseph Jabbour[1,2,*], Jalal Possik[1], Charles Yaacoub[1], Sina Namaki Araghi[3], Simon Gorecki[4], Gregory Zacharewicz[2] and Adriano O. Solis[5]

[1]ICL, Junia, Université Catholique de Lille, LITL, F-59000 Lille, France
[2]Laboratoire des Sciences des Risques, IMT Mines Alès, Alès, 30100, France
[3]Tarbes University of Technology (UTTOP), Production Engineering Laboratory (LGP), Tarbes, France
[4]University of Bordeaux, CNRS, IMS, UMR 5218, 33405 Talence, France
[5]School of Administrative Studies, York University, Toronto, Ontario, M3J 1P3, Canada

[*]Corresponding author. Email address: joseph.jabbour@univ-catholille.fr

Abstract

This paper explores the integration of Discrete Event Simulation (DES) and Virtual Reality (VR) technologies, focusing on JaamSim and Unity. By combining the strengths of both, we aim to enhance simulation capabilities across various domains. The integration leverages the High-Level Architecture (HLA) IEEE standard to address interoperability and synchronization challenges. This approach mitigates resource consumption issues seen in traditional methods that rely on extensive data transfers, which are dependent on network and server performance. Our integrated system provides an immersive and interactive environment that improves user engagement and decision-making. It allows trainees to better understand complex processes through hands-on VR experiences and enables decision-makers to visualize the impacts of different strategies in real-time. This approach maintains a constant rate of messages per client cycle, irrespective of the simulation length, unlike previous methods that heavily depended on the process duration.

Keywords: Discrete Event Simulation (DES), Virtual Reality (VR), JaamSim, Unity, High-Level Architecture (HLA), Distributed Simulation, Interoperability, Immersive environments

1. Introduction

The rapid advancement of technology has ushered in an era where innovative tools are increasingly being utilized across various domains to enhance efficiency, understanding, and performance. Among these tools, Virtual Reality (VR) and Discrete Event Simulation (DES) stand out due to their transformative potential. VR, with its ability to create immersive and interactive environments, has found applications in sectors ranging from healthcare to education, offering significant improvements in training, therapy, and learning outcomes. DES, on the other hand, excels in modeling and analyzing complex systems, providing valuable insights that aid in optimizing processes across logistics, manufacturing, and healthcare.

This paper explores the current state of VR and DES technologies, their individual applications, and the existing challenges in their integration. It aims to provide a comprehensive overview of the benefits that

such integration could bring to various sectors, alongside a detailed examination of the barriers that need to be overcome. By addressing these challenges, the paper seeks to contribute to the ongoing efforts in developing robust and scalable solutions that leverage the strengths of both VR and DES, ultimately paving the way for more advanced and effective simulation environments. In this paper, we specifically address the integration of JaamSim, a DES tool, with Unity, a leading platform for VR development. JaamSim is an open-source simulation software known for its flexibility and comprehensive set of features for modeling complex systems (JaamSim, 2024). Unity, on the other hand, is a widely used game development engine that offers extensive capabilities for creating immersive VR experiences (Unity, 2024). The integration of these two platforms presents significant challenges, particularly in terms of data exchange and synchronization. To overcome the interoperability hurdles inherent in combining these distinct technologies, the High-Level Architecture (HLA) IEEE standard is utilized. It is designed to facilitate interoperability and reusability of simulation components across diverse platforms (IEEE, 2010). HLA provides a framework that allows simulations to communicate and operate together within a common environment, making it an ideal choice for integrating JaamSim and Unity.

This paper is organized as follows: Section 2 provides a comprehensive literature review on the individual applications and benefits of VR and DES technologies, as well as the barriers to integrating both technologies. Section 3 outlines the materials and methods used in integrating JaamSim with Unity, detailing the tools, frameworks, and methodologies employed. Section 4 presents the results and discusses the advantages of the integrated system, highlighting its effectiveness in enhancing performance, user engagement, and decision-making. Section 5 covers the conclusion and perspectives, including future research directions such as testing with real case studies, incorporating reinforcement learning, adding behaviors to agents, and enhancing user experience in VR environments. Finally, Section 6 acknowledges the contributions and funding sources that supported this research.

2. State of the art

VR has been increasingly integrated into various domains, demonstrating significant benefits across multiple sectors. For instance, in healthcare, VR provides immersive training environments, enhancing the realism and effectiveness of medical training (Halbig et al., 2022). Similarly, VR applications in psychology enable the creation of three-dimensional facsimiles of real objects, facilitating better psychological studies and treatments (Wilson & Soranzo, 2015). In the realm of education, VR's immersive capabilities have been leveraged to create engaging and effective learning environments, improving students' understanding and retention of complex subjects (Radianti et al., 2020). The technological advancements in VR have also been beneficial in fields like manufacturing and logistics, where VR aids in better planning and operational efficiency (LinkedIn, 2024). The aerospace and defense industries have adopted VR to simulate complex scenarios and training exercises, enhancing preparedness and performance (MathWorks, 2024). Moreover, public health and wellness sectors utilize VR for therapeutic purposes, offering new ways to support mental health and physical rehabilitation (Hamad & Jia, 2022).

On the other hand, DES has proven to be a versatile tool in modeling and analyzing complex systems across various domains. Its applications in logistics and manufacturing have enabled better planning and optimization of processes (LinkedIn, 2024). In aerospace and defense, DES helps in modeling intricate systems and evaluating different operational strategies (MathWorks, 2024). The healthcare sector benefits from DES by using it to model patient flows and optimize resource allocation, improving overall efficiency (Vázquez-Serrano et al., 2021). Additionally, the theoretical foundations and practical applications of DES are well-documented, providing a robust framework for researchers and practitioners (Wainer & Mosterman, 2016; Abu-Taieh & El-Sheikh, 2010). These applications underscore the versatility and effectiveness of DES in addressing complex and dynamic problems across different sectors.

Despite the promising applications of both DES and VR individually, integrating these technologies has not yet been fully successful. The combination of DES and VR could potentially offer enhanced simulation capabilities by providing immersive and interactive environments for analyzing complex systems. However, several challenges hinder this integration. One significant barrier is the technological limitations of current VR systems, which lack the necessary features for seamless integration with DES, thus limiting their combined potential (Balin et al., 2023). The interoperability issues between DES software and VR platforms pose substantial challenges, as most VR systems are designed primarily for gaming and entertainment, not for the detailed analytical processes required in DES (Cook et al., 2019). Interoperability refers to the ability of different systems, devices, or applications to work together within and across organizational boundaries. In the context of DES and VR, interoperability issues arise due to differences in data formats, software platforms, and communication protocols. These issues can significantly hinder the seamless integration and effective utilization of combined DES

and VR systems. A significant interoperability issue is the compatibility of data between DES and VR systems. DES models often produce detailed and complex data sets that describe the behavior and interactions of various system components over time. Translating this data into a VR environment requires a consistent and compatible data format that both systems can interpret and use effectively. However, current VR platforms may not support the intricate data structures produced by DES, leading to difficulties in creating accurate and realistic VR simulations (Turner et al., 2016). Another major challenge is the integration of software platforms used for DES and VR. DES typically relies on specialized simulation software, such as Arena, Simul8, or AnyLogic, which have their unique interfaces, data handling methods, and computational processes. On the other hand, VR development often involves platforms like Unity or Unreal Engine, which are designed primarily for creating interactive and immersive environments. Bridging the gap between these disparate software systems requires the development of middleware or integration frameworks that can facilitate communication and data exchange between DES and VR platforms (Balin et al., 2023). The integration of DES and VR also necessitates the use of robust communication protocols to ensure smooth and real-time data exchange. DES models often run complex simulations that can generate large volumes of data at high speeds. Transferring this data to a VR environment in real-time without significant latency or data loss is a considerable technical challenge. Existing communication protocols may not be optimized for such high-performance requirements, leading to delays, synchronization issues, and reduced simulation accuracy (Webster & Kourkoulakou, 2022).

To address interoperability issues between heterogeneous components, standardization efforts are essential and a shift towards Distributed Simulation (DS) has been recognized as essential (Gorecki et al., 2020, 2021). For instance, initiatives like the IEEE HLA standard for modeling and simulation provide guidelines for creating interoperable simulation systems, which can be extended to include VR integration (IEEE Xplore, 2010). The adoption of HLA standard facilitates time synchronization and communication between heterogeneous components, ensuring coherent parallel operation (Possik, Zacharewicz, et al., 2023; Possik et al., 2018, 2019).

Practical examples of integrated DES and VR systems highlight both the potential benefits and interoperability challenges. For example, in the healthcare sector, integrating DES models of patient flow with VR-based training simulations can provide a more comprehensive understanding of hospital operations. However, these projects often encounter issues related to data translation and real-time synchronization, underscoring the need for robust interoperability solutions (Tokgöz et al., 2022). In addition, previous literature highlights that the integration of DES and VR is highly resource-demanding, particularly when both systems fully reproduce a real environment and operate in parallel, including architecture, resources, agents, and process flows in both environments(Possik, Asgary, et al., 2023; Possik et al., 2021, 2022). This approach requires extremely powerful servers to handle the simulations effectively. Furthermore, users often encounter significant slowdowns when using VR headsets, which hampers the overall performance and usability of the system. Jabbour et al. (2023) presented a novel approach to integration aimed at addressing the aforementioned challenges. The integration strategies detailed in their work significantly enhance the feasibility and practicality of such systems, ensuring improved performance and optimized resource management. This paper builds upon the conceptualization and approach presented in Possik et al. (2022) by implementing HLA layers for each of the DES and VR components to ensure HLA compatibility and resolve interoperability issues. It presents a simple process flow to demonstrate the feasibility of the proposed approach, accompanied by a performance comparison to showcase the effectiveness of the DS system. The following section outlines the systematic approach adopted, detailing the tools, frameworks, and methodologies leveraged to achieve seamless integration, thereby ensuring a robust and scalable solution for simulation in VR environments.

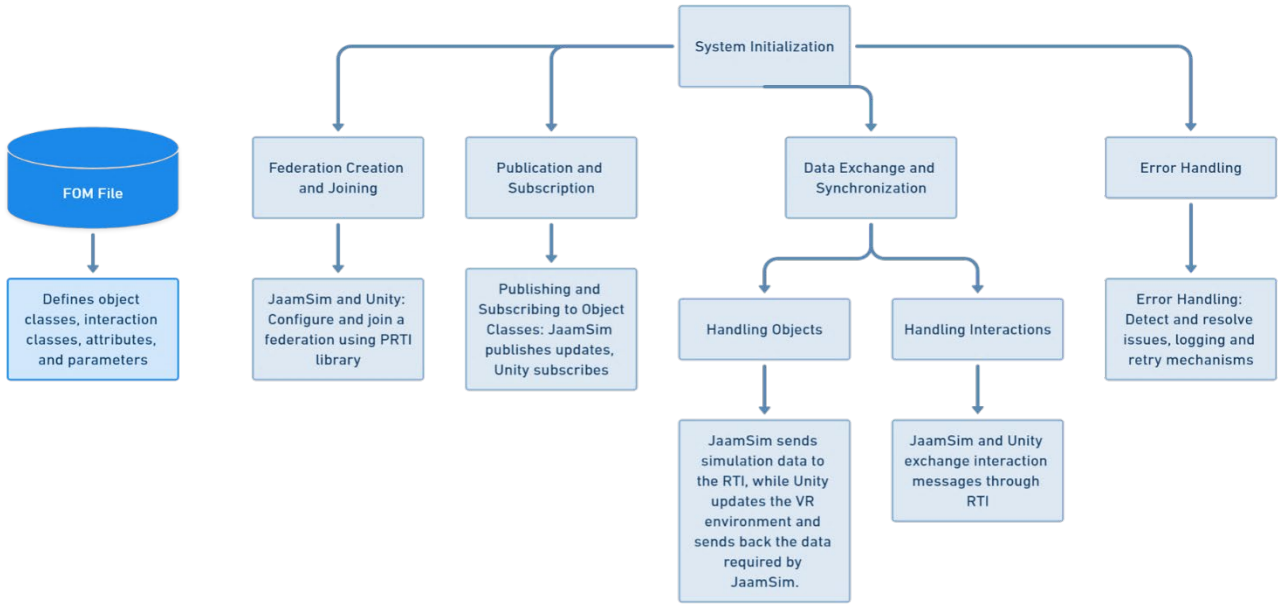

Figure 1. HLA standard: communication and interaction between federates

3. Materials and Methods

The need for interoperability among heterogeneous simulation components is addressed by the IEEE High-Level Architecture, a distributed simulation standard. HLA enables multi-model development, data exchange, interoperability, reuse, and communication with external systems. This section introduces the process of synchronous simulation using HLA with JaamSim and Unity.

The first version of HLA was HLA US DoD 1.3 developed within the department of defense of the united states, later adopted by IEEE in 2000 as HLA IEEE 1516. It was revised and updated in 2010, leading to HLA Evolved. The upcoming version, HLA 4, soon to be released, will introduce new system modeling and safety-related functions. HLA supports the development of DS by creating simulations composed of various components, called "federates." A federation includes multiple federates, a runtime infrastructure (RTI), and a federated object model (FOM). The RTI provides standardized services for data exchange, synchronization, and management, while the FOM defines the objects and interaction classes used for communication.

3.1. HLA mechanism

HLA facilitates modeling compatibility and model reuse. Models can run on separate computers with different operating systems and programming languages, distributed across a LAN or WAN. These components are unified in a federation, where a publish/subscribe (p/s) mechanism based on FOM and HLA objects management enables data exchange among federates.

In this study, we utilized JaamSim 2024-04, Unity 2022.3, and the HLA Evolved version to develop a robust system architecture that integrates DES with VR technologies. The architecture aims to seamlessly connect JaamSim, known for its flexibility in modeling complex systems, with Unity, a leading VR development platform, to create an advanced simulation environment. The PRTI library of the High-Level Architecture (HLA) standard was employed to address the challenges of interoperability and data synchronization. By implementing custom adapters and interfaces, we enabled effective communication between JaamSim and Unity, ensuring that simulation data and user interactions were accurately synchronized. This integration facilitates a cohesive federation execution model managed by the PRTI library, providing a scalable and reliable framework for complex simulations. Through this approach, we were able to leverage the strengths of both JaamSim and Unity, paving the way for more immersive and effective simulation applications.

Upon execution, JaamSim operates as a black-box simulator, encapsulating its internal mechanisms while providing outputs based solely on its input parameters. To integrate JaamSim into the HLA framework, we developed a specialized interface that ensures compatibility with HLA standards. This interface functions by wrapping JaamSim's native execution environment, facilitating the translation of simulation data into HLA-compliant formats. It handles the registration of JaamSim objects and interactions with the Runtime Infrastructure (RTI) using the PRTI library. The interface also manages data exchange, subscribing to and publishing attribute updates and interactions as specified in the Federation Object Model (FOM). Through this approach, JaamSim

can seamlessly communicate and synchronize with other federates within the HLA federation, thereby enhancing interoperability and ensuring coordinated simulation operations across diverse platforms.

The integration of JaamSim and Unity using the PRTI library of the HLA involves several technical components and steps. Here, we provide a detailed technical explanation of how the PRTI library facilitates this integration.

Figure 1 illustrates the flowchart of the system initialization process, encompassing federation creation, publication and subscription, data exchange, interactions, and error handling. Each of these processes is detailed below.

3.1.1. Federation creation and joining

Both JaamSim and Unity federates are configured to initialize and join a federation, as shown in Figure 2.

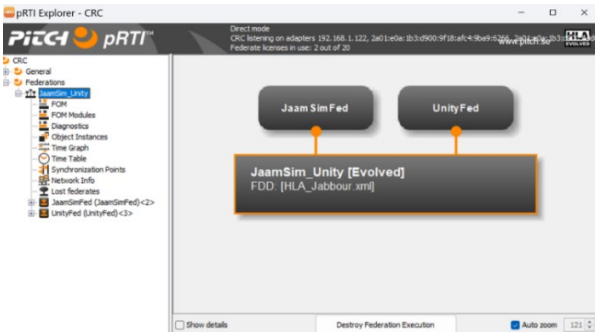

Figure 2. HLA pRTI Federation

This involves connecting to the RTI (implemented using the Java PRTI library), creating a federation execution environment, and joining the federation as federates. Figure 3 outlines the procedure for connecting federates to an HLA federation. Each federate (JaamSim and Unity) follows these steps to join the federation, specifying the FOM, an XML file that defines the structure of the federation.

```
Input: federateName, federationName
Output: federateHandle
Function
FederateJoinFederation (federateName, federationName):
    federateHandle = PRTI_Connect()
    PRTI_JoinFederation (federateHandle, federateName,
federationName)
    return federateHandle
```

Figure 3. Algorithm 1: Federate Join Federation

3.1.2. Federation object model

The FOM includes definitions for object classes, interaction classes, attributes, and parameters that federates will use to exchange information. As shown in Figure 4, to enable the movement of agents within the VR environment, the simulation environment is configured to send entities as classes. Each entity class includes attributes such as type, index, next destination, and action. Conversely, the VR environment sends interaction messages to the simulation environment. These messages include ActionDone, sent when an action is completed in the VR environment, and ArrivedDestination, sent when an agent reaches its destination, allowing the simulation to continue.

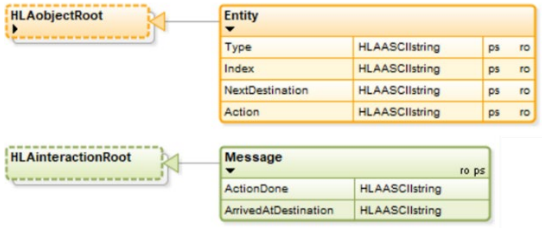

Figure 4. HLA FOM configuration

3.1.3. Publication and subscription

All the aforementioned attributes and parameters are configured as publish/subscribe (ps) attributes and parameters. For the entities, the simulation publishes the attributes (Type, Index, NestDestination, and Action), and the VR environment subscribes to these attributes to receive updates. For the messages, the VR environment publishes the parameters (ActionDone and ArrivedAtDestination), and the simulation subscribes to them to adapt accordingly.

```
Input: federateHandle, objectClassName, attributes
Output: None
Function    PublishSubscribeObjectClass(federateHandle,
objectClassName, attributes):
    PRTI_PublishObjectClass(federateHandle,
objectClassName, attributes)
    PRTI_SubscribeObjectClass(federateHandle,
objectClassName, attributes)
```

Figure 5. Algorithm 2: PublishSubscribeObjectClass

Figure 5 shows the function that explains how to assign the objects and attributes to each federate handle, specifying whether those attributes are to be published by this federate or subscribed to. The same algorithm should be followed to define the publish/subscribe configurations for interactions and parameters.

3.1.4. Data exchange and synchronization

In this DS system, JaamSim acts as the simulation engine, creating and managing the simulation data. The data is structured as objects with specific attributes or interactions with defined parameters, which are then communicated to the RTI. The RTI serves as a middleware that facilitates data exchange between JaamSim and Unity. Unity, functioning as the 3D visualization engine, receives the simulation data from the RTI and updates the VR environment based on the information provided by the RTI. This process ensures that the virtual environment accurately reflects the

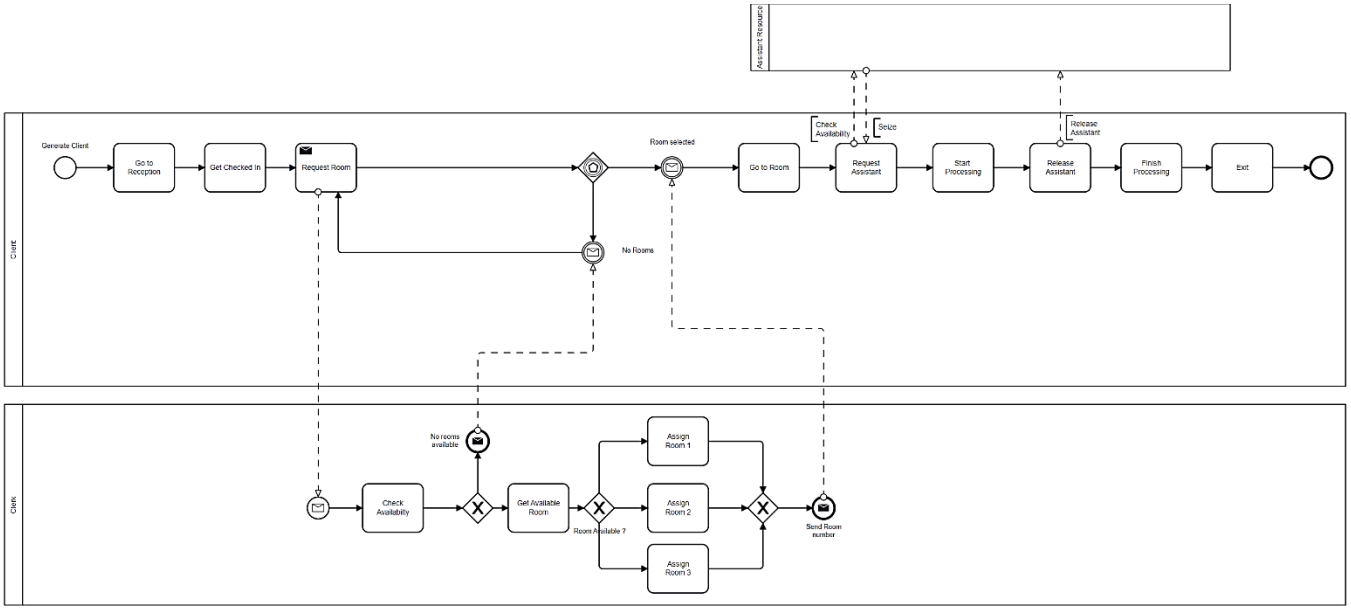

Figure 6. BPMN Process flow of the DES

current state of the simulation. Moreover, Unity collects and sends back relevant data to the RTI, which JaamSim requires to continue its simulation processes. This feedback loop includes information such as the status of actions performed within the VR environment. The time management mechanism of HLA has been disabled in this configuration. Consequently, the RTI does not manage the synchronization of events over simulated time. Instead, data published to the RTI is instantly disseminated to all federates that have subscribed to receive it, ensuring real-time updates across the system.

Figure 7 ensures that when an object's attributes are updated by one federate, those changes are reflected to other federates that are subscribed to that object's updates. This maintains consistency and synchronization of object states across the simulation. Figure 8 enables federates to send interactions to one another, facilitating communication and coordination within the simulation. For instance, one federate might send an interaction indicating an event, such as "*action done*", which other federates then receive and act upon.

The function *UpdateAttributes* calls *PRTI_UpdateAttributes(federateHandle, objectInstanceHandle, attributes)*, which updates the attributes of the specified object instance in the PRTI. The function *ReflectAttributes* calls *PRTI_GetAttributeUpdates(federateHandle)*, which retrieves the updated attributes for the specified federate from the PRTI and returns *reflectedAttributes*, which contain the updated attributes that have been received.

Similarly to the previous process, the function *SendInteraction* calls *PRTI_SendInteraction(federateHandle, interactionClass, parameters)*, which sends the specified interaction and its parameters to the PRTI. The function *ReceiveInteraction* calls *PRTI_GetInteractionParameters(federateHandle)*, which retrieves any interaction parameters intended for the specified federate from the PRTI and returns *receivedParameters*, which contain the parameters of the received interaction.

```
Input: federateHandle, objectInstanceHandle, attributes
Output: reflectedAttributes
Function                UpdateAttributes(federateHandle,
objectInstanceHandle, attributes):
    PRTI_UpdateAttributes(federateHandle,
objectInstanceHandle, attributes)

Function ReflectAttributes(federateHandle):
    reflectedAttributes                                    =
PRTI_GetAttributeUpdates(federateHandle)
    return reflectedAttributes
```

Figure 7. Algorithm 3: UpdateReflectAttributes

```
Input: federateHandle, interactionClass, parameters
Output: receivedParameters
Function                SendInteraction(federateHandle,
interactionClass, parameters):
    PRTI_SendInteraction(federateHandle,
interactionClass, parameters)
Function ReceiveInteraction(federateHandle):
    receivedParameters                                     =
PRTI_GetInteractionParameters(federateHandle)
    return receivedParameters
```

Figure 8. Algorithm 4: SendReceiveInteraction

3.1.5. *Error handling and scalability*

Both JaamSim and Unity implement mechanisms to detect and resolve synchronization issues, data inconsistencies, and communication failures. Robustness is ensured through logging and retry mechanisms, as shown in the function *HandleError* of Figure 9.

```
Input: operation
Output: success
Function HandleError(operation):
Try:
     operation()
     success = True
Except Exception as e:
     LogError(e)
     success = False
    return success
```

Figure 9. Algorithm 5: ErrorHandling

3.2. DES and VR environments

In this section, the functionalities of the DS platform are discussed through a simple process flow to demonstrate the feasibility and effectiveness of this architecture and integration. Figure 10 shows the VR environment (Unity 3D) where agents move according to the DES (JaamSim) process flow. The example involves an entry point, a reception area, and three processing rooms. The process represented in Figure 6 begins with the generation of a client, who is assigned an index and directed to the reception area. Upon arrival, the client undergoes a check-in process. Following this, the client is randomly assigned to one of the three processing rooms. An assistant is then assigned to guide the client to the designated room and initiate the processing phase. After escorting the client and starting the processing, the assistant is released and returns to the reception to assist other clients. The client continues the processing independently. Once the processing is completed, the client exits the system.

Figure 10. VR environment

In JaamSim, clients are simulation entities designed to represent individuals generated by entity generators and processed within the simulation. Each client is assigned a list of key attributes that define their identity and behavior: Type, Index, NextDestination, and Action.

- The index assigned to a client is a unique identifier that remains constant and unchangeable throughout the simulation. This helps in tracking and referencing the client.
- Unlike the index, the next destination and action for a client are dynamic and updated at different stages based on the model's logic. The next destination could be another location or process the client needs to move to, while the action specifies what the client will do upon reaching the destination.

The Seize component manages the allocation of resources to entities as they progress through the simulation. After clients receive their attributes, they often need to acquire specific resources (assistants) to proceed with their tasks. The Seize component allocates an assistant from the assistant's resource pool. When a client reaches the Seize component, it requests an assistant. If an assistant is available, the client seizes the resource and continues to the next step. If no assistants are available, the client must wait until one becomes free. Clients perform their tasks assisted by the allocated assistant resources. Upon task completion, clients release the resources back to the assistant's resource pool using the Release component, making them available for other clients.

Subsequently, clients arrive at a decision point managed by the Branch component. At this point clients are routed to one of three possible destinations, such as Room1, Room2, or Room3. The routing decision is based on a discrete probability distribution that can be adjusted based on the simulation needs. Finally, entities reach the end of their journey in the simulation at the EntitySink endpoint component. These EntitySink components represent the conclusion of the entities' lifecycle within the simulation, where they are removed from the system.

Table 1. HLA attributes

Attributes		
Type	1.	Clients
	2.	Assistants
Index	Assigned automatically by the system	
Next Destination	0.	Stay in place
	1.	Reception
	2.	Room1
	3.	Room2
	4.	Room3
	5.	Exit
Actions	0.	No action
	1.	Get checked in
	2.	Processing Room 1
	3.	Processing Room 2
	4.	Processing Room 3
	5.	Assist Client
	6.	Exit

Table 1 lists the attributes used to construct this simulation case. These attributes play distinct roles in guiding the simulation entities (clients) through their workflows in both simulation and VR environments.

Table 2. Communication process between the DES and VR environments

AGENT ARRAY					
Type	Index	Next Destination	Action	SIMULATION	VIRTUAL REALITY
1	0	0	0	1: Client; 0: No Index; 0: No Next Destination; 0: No Action	*Client*: Generate Entity Type 1
				Client: Generate Entity Type 1 (Client)	
1	1	1	0	1: Client; 1: Index; 1: Reception; 0: No Action	
				Client: Assign index: 1 (Client 1) *Client 1: Assign next destination: Reception*	
					Client 1: Go to Reception Send Arrived at Destination message
1	1	0	1	1: Client; 1: Index; 0: No Next Destination; 1: Get Checked in	
				Client 1: Assign Action: "Get Checked in"	
					Client 1: Get checked in *Client 1*: Send Action Done message
2	1	3	0	2: Assistant; 1: Index; 3: Room 2; 0: No Action	
				(Seize Assistant) *Assistant 1: Assign next destination: "Room 2"*	
1	1	3	0	1: Client; 1: Index; 3: Room 2; 0: Get Checked in	
				Client 1: Assign next destination: "Room 2"	
					Client 1: Go to "Room 2" *Assistant* 1: Go to "Room 2" *Client 1*: Send Arrived at Destination message *Assistant* 1: Send Arrived at Destination message
1	1	0	2	1: Client; 1: Index; 0: No Next Destination; 2: Processing Room 2	
				Client 1: Assign Action: Processing "Room 2"	
2	1	0	5	2: Assistant; 1: Index; 0: No Next Destination; 5: Assist Client	
				Assistant 1: Assign Action: Assist Client	
					Client 1: Processing in "Room 2" *Assistant 1*: Assist Client
					Assistant 1: Send Action Done message
2	1	1	0	2: Assistant; 1: Index; 1: Reception; 0: No Action	
				Assistant 1: Assign next destination: Reception *(Release Assistant)*	
					Assistant 1: Go back to reception
					Client 1: Send Action Done message
1	1	5	0	1: Client; 1: Index; 5: Exit; 0: No Action	
				Client 1: Assign next destination: Exit	
					Client 1: Go to Exit *Client 1*: Send Arrived at Destination message
1	1	0	6	1: Client; 1: Index; 0: No Next Destination; 6: Exit	
				Client 1: Assign Action: Exit	
					Client 1: Exit *Client 1*: Send Action Done message

Type attribute identifies the kind of entity involved, such as a client or assistant. For instance, "1: Client" signifies that the entity is a client. *Index* provides a unique identifier for each entity, ensuring that each client or assistant can be individually tracked throughout the process. For example, "1: Index" refers to the first client. *Next Destination* specifies the upcoming location where the entity needs to go. It directs the entity to the appropriate area, such as "1: Reception" indicating the client should go to the reception, or "3: Room 2" directing them to Room 2. *Action* defines the specific task the entity must perform at its current or next location. For instance, "1: Get Checked in" means the client needs to complete the check-in process, and "2: Processing Room 2" indicates that the client should begin processing in Room 2.

The process presented in Table 2 begins by generating a client (Type 1) with no initial index, destination, or action. The client is then assigned an index (1) to be identified during the simulation, directed to the reception (Next Destination: 1), and instructed to go there.

Upon arrival, the client sends an "Arrived at Destination" message. At the reception, the client is assigned the action "Get Checked in" (Action: 1). Once checked in, the client sends an "Action Done" message. The client is then directed to Room 2 (Next Destination: 3) along with an assistant. Both the client and the assistant proceed to Room 2 and send arrival messages. In Room 2, the client begins processing (Action: 2), while the assistant assists the client (Action: 5). Once their tasks are completed, the assistant sends an

"Action Done" message and returns to the reception. The client finishes processing, sends an "Action Done" message, and is directed to exit the system (Next Destination: 5). Finally, the client exits and sends a completion message.

This detailed process flow highlights the seamless interaction between the DES model and the VR environment, showcasing the strengths of the integrated system. The next section will delve into the results and discussion, providing insights into the advantages of this approach and integration.

4. Results and Discussion

The integration of JaamSim and Unity, as demonstrated in Figure 11, showcases the synergy between DES and VR environments. Figure 11 illustrates the JaamSim simulation on the left, where the detailed simulation logic and process flow are defined, and the Unity 3D environment on the right, where agents are visualized performing their respective tasks. JaamSim handles the detailed event-based modeling, where each client (agent) is generated, assigned tasks, and guided through various stages such as check-in and processing. The Unity 3D environment brings the simulation to life by providing a visual and immersive representation of the JaamSim model. Figure 11 shows two agents (clients) moving through rooms and performing tasks as directed by the DES, enhancing user engagement and understanding by visualizing the simulation in a realistic setting. This VR environment makes the simulation more intuitive and accessible, allowing users to see the agents interact with their surroundings, thereby improving comprehension and engagement.

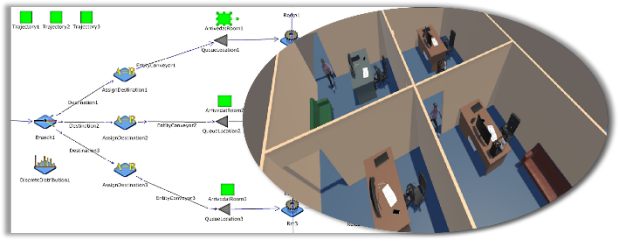

Figure 11. JaamSim/Unity 3D

In the context of exchanged messages between DES and VR, the approach proposed in this paper achieves a constant rate of 18 messages per client cycle, which starkly contrasts with the previous methodologies that required 24 messages per second per agent throughout the process. This discrepancy means that, within the initial second, the previous approaches already surpass the 18 messages required by the new methodology. Moreover, the new approach decouples the message volume from the process duration, which can vary widely depending on the case study; from minutes to days or even weeks. This independence implies that, for two individuals (a client and an assistant) moving within the simulation environment, the number of messages per cycle remains unchanged. Quantitatively, within the first ten minutes of the simulation, the previous methodologies exceed 14,000 messages, while the new approach maintains a much lower volume of approximately 160 messages as shown in Figure 12.

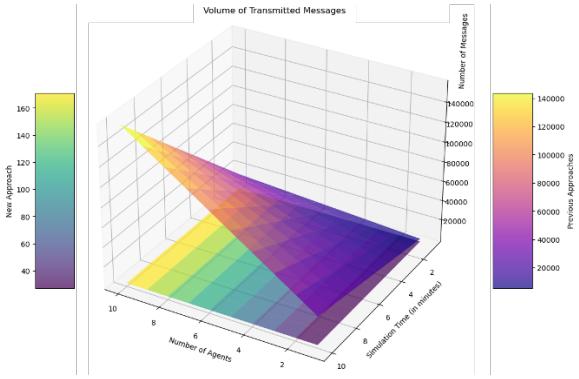

Figure 12. Comparison of the volume of transmitted messages

The integrated system serves as an effective training tool. Trainees can immerse themselves in the VR environment, gaining a deeper understanding of the simulation process through hands-on, interactive experiences. This immersive approach significantly enhances retention and comprehension. Additionally, decision-makers can leverage the VR environment to test various scenarios, observe outcomes, and make well-informed decisions. The real-time visualization of different strategies' impacts substantially improves decision-making capabilities.

Traditional integration methods often require extensive data transfers to replicate environments, which can be resource-intensive and heavily reliant on network performance. Our integration approach, coupled with efficient resource management, minimizes unnecessary data exchanges and reduces dependency on network and server performance, resulting in a more robust and scalable system.

5. Conclusions

In conclusion, the integration of DES and VR technologies presents a significant opportunity to enhance simulation capabilities across various domains. By leveraging the strengths of both technologies, we can create more immersive and interactive simulation environments that offer deeper insights and improved outcomes. This paper has demonstrated the feasibility and effectiveness of integrating JaamSim, a robust DES tool, with Unity, a leading VR development platform, using the HLA standard to address interoperability and synchronization challenges. Our integration framework overcomes the traditional issues of resource consumption seen in older integrations, which relied heavily on sending numerous messages to replicate an exact environment in both DES and VR. This previous method was highly dependent on

network and server performance, leading to inefficiencies. By utilizing our approach of integration and the HLA standard, we have mitigated these issues, ensuring a more efficient and scalable solution.

Looking forward, several key areas require further research and development to fully realize the potential of integrated DES and VR systems. Testing our platform with real case studies in sectors such as manufacturing, healthcare, and others will provide practical insights and help identify domain-specific challenges. Enhanced testing with VR headsets and controllers in larger, more complex case studies will assess the system's performance and ensure its scalability and reliability. Incorporating complex behaviors and decision-making capabilities into the VR agents is crucial for creating more realistic and nuanced simulations. Customizing integrated DES and VR systems to meet the unique requirements of different industries, such as healthcare, manufacturing, and logistics, will enhance their effectiveness and adoption. Finally, enhancing the user experience in VR environments by improving interactivity and realism is vital for training and operational simulations, providing users with more engaging and effective tools. By addressing these areas, we can advance the integration of DES and VR technologies, paving the way for more advanced, immersive, and effective simulation environments. These developments have the potential to transform how we approach complex problem-solving and training across multiple sectors, ultimately leading to better outcomes and more efficient processes.

Funding

This research was supported by the Agence Nationale de la Recherche (ANR) under the DemoES - PEIA project (ANR-21-DMES-0014). We would like to extend our sincere gratitude to the ANR for their financial support. Their funding has been instrumental in the successful execution of this project.

References

Abu-Taieh, E. and El-Sheikh, A. (2010). *Handbook of Research on Discrete Event Simulation Environments: Technologies and Applications*. IGI Global. https://doi.org/10.4018/978-1-60566-774-4

Balin, S., Bolognesi, C. M., and Borin, P. (2023). Integration of Immersive Approaches for Collaborative Processes with Building Information Modeling (BIM) Methodology for the AEC Industry: An Analysis of the Current State and Future Challenges. *Virtual Worlds*, 2(4), Article 4. https://doi.org/10.3390/virtualworlds2040022

Cook, M., Lischer-Katz, Z., Hall, N., Hardesty, J., Johnson, J., McDonald, R., and Carlisle, T. (2019). Challenges and Strategies for Educational Virtual Reality. *Information Technology and Libraries*, 38: 25–48. https://doi.org/10.6017/ital.v38i4.11075

Gorecki, S., Possik, J., Zacharewicz, G., Ducq, Y., and Perry, N. (2020). A Multicomponent Distributed Framework for Smart Production System Modeling and Simulation. *Sustainability*, 12(17), Article 6969. https://doi.org/10.3390/su12176969

Gorecki, S., Possik, J., Zacharewicz, G., Ducq, Y., and Perry, N. (2021). Business Models for Distributed-Simulation Orchestration and Risk Management. *Information*, 12(2), Article 71. https://doi.org/10.3390/info12020071

Halbig, A., Babu, S. K., Gatter, S., Latoschik, M. E., Brukamp, K., and von Mammen, S. (2022). Opportunities and Challenges of Virtual Reality in Healthcare – A Domain Experts Inquiry. *Frontiers in Virtual Reality*, 3, Article 837616. https://doi.org/10.3389/frvir.2022.837616

Hamad, A. and Jia, B. (2022). How Virtual Reality Technology Has Changed Our Lives: An Overview of the Current and Potential Applications and Limitations. *International Journal of Environmental Research and Public Health*, 19(18), Article 11278. https://doi.org/10.3390/ijerph191811278

IEEE Xplore (2010). IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA)—Framework and Rules. Retrieved from https://ieeexplore.ieee.org/document/STD96061

JaamSim (2024). JaamSim - Discrete-Event Simulation Software. Retrieved from https://www.jaamsim.com/

Jabbour, J., Possik, J., Yaacoub, C., Solis, A. O., Kieken, D., Sobanski, T., and Zacharewicz, G. (2023). Synergistic Fusion of Simulation and Virtual Reality: A Proposed New Approach for Collaborative Integration. *Proceedings of the 22nd International Conference on Modeling and Applied Simulation* (MAS 2023), Athens, Greece, September 18-20, 2023, Paper 023:1-8. https://doi.org/10.46354/i3m.2023.mas.023

LinkedIn (2024). How can you use discrete-event simulation to plan logistics better? Retrieved from https://www.linkedin.com/advice/1/how-can-you-use-discrete-event-simulation-1e

MathWorks (2024). Applications of Discrete Event Simulation in the Aerospace and Defense Industry. Retrieved from https://www.mathworks.com/videos/applications-of-discrete-event-simulation-in-the-aerospace-and-defense-industry-81569.html

Possik, J., Amrani, A., and Zacharewicz, G. (2018). Development of a co-simulation system as a decision-aid in Lean tools implementation. *Summer Simulation Multiconference. https://doi.org/10.22360/SUMMERSIM.2018.SCSC.037*

Possik, J., Asgary, A., Solis, A. O., Zacharewicz, G., Shafiee, M. A., Najafabadi, M. M., Nadri, N., Guimaraes, A., Iranfar, H., Ma, P., Lee, C. M.,

Tofighi, M., Aarabi, M., Gorecki, S., and Wu, J. (2023). An Agent-Based Modeling and Virtual Reality Application Using Distributed Simulation: Case of a COVID-19 Intensive Care Unit. *IEEE Transactions on Engineering Management*, 70(8): 2931–2943. https://doi.org/10.1109/TEM.2022.3195813

Possik, J., Azar, D., Solis, A. O., Asgary, A., Zacharewicz, G., Karami, A., Tofighi, M., Najafabadi, M., Shafiee, M. A., Merchant, A. A., Aarabi, M., and Wu, J. (2022). A distributed digital twin implementation of a hemodialysis unit aimed at helping prevent the spread of the Omicron COVID-19 variant. *Proceedings of the IEEE/ACM 26th International Symposium on Distributed Simulation and Real Time Applications (DS-RT 2022)*, 168–174. https://doi.org/10.1109/DS-RT55542.2022.9932047

Possik, J., D'Ambrogio, A., Zacharewicz, G., Amrani, A., and Vallespir, B. (2019). A BPMN/HLA-Based Methodology for Collaborative Distributed DES. *Proceedings of the IEEE 28th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE 2019)*, 118–123. https://doi.org/10.1109/WETICE.2019.00033

Possik, J., Gorecki, S., Asgary, A., Solis, A. O., Zacharewicz, G., Tofighi, M., Shafiee, M. A., Merchant, A. A., Aarabi, M., Guimaraes, A., and Nadri, N. (2021). A Distributed Simulation Approach to Integrate AnyLogic and Unity for Virtual Reality Applications: Case of COVID-19 Modelling and Training in a Dialysis Unit. *Proceedings of the IEEE/ACM 25th International Symposium on Distributed Simulation and Real Time Applications (DS-RT 2021)*, virtual conference, September 2021, 1–7. https://doi.org/10.1109/DS-RT52167.2021.9576149

Possik, J., Zacharewicz, G., Zouggar, A., and Vallespir, B. (2023). HLA-based time management and synchronization framework for lean manufacturing tools evaluation. *Simulation*, 99(4): 347–362. https://doi.org/10.1177/00375497221132577

Radianti, J., Majchrzak, T. A., Fromm, J., and Wohlgenannt, I. (2020). A systematic review of immersive virtual reality applications for higher education: Design elements, lessons learned, and research agenda. *Computers & Education*, 147, Article 103778. https://doi.org/10.1016/j.compedu.2019.103778

Tokgöz, P., Stampa, S., Wähnert, D., Vordemvenne, T., and Dockweiler, C. (2022). Virtual Reality in the Rehabilitation of Patients with Injuries and Diseases of Upper Extremities. *Healthcare*, 10(6), Article 1124. https://doi.org/10.3390/healthcare10061124

Turner, C. J., Hutabarat, W., Oyekan, J., and Tiwari, A. (2016). Discrete Event Simulation and Virtual Reality Use in Industry: New Opportunities and Future Trends. *IEEE Transactions on Human-Machine Systems*, 46(6): 882–894. https://doi.org/10.1109/THMS.2016.2596099

Unity (2024). Unity Real-Time Development Platform | 3D, 2D, VR & AR Engine. Retrieved from https://unity.com/

Vázquez-Serrano, J. I., Peimbert-García, R. E., and Cárdenas-Barrón, L. E. (2021). Discrete-Event Simulation Modeling in Healthcare: A Comprehensive Review. *International Journal of Environmental Research and Public Health*, 18(22), Article 12262. https://doi.org/10.3390/ijerph182212262

Wainer, G. and Mosterman, P. (2016). Discrete-Event Modeling and Simulation: Theory and Applications. In *Discrete-Event Modeling and Simulation: Theory and Applications* (p. 493). https://doi.org/10.1201/9781315218731

Webster, C. and Kourkoulakou, S. (2022). Composing in virtual immersion: Avatar and representation. *Hybrid. Revue des arts et médiations humaines*, 9. https://doi.org/10.4000/hybrid.2968

Wilson, C. J. and Soranzo, A. (2015). The Use of Virtual Reality in Psychology: A Case Study in Visual Perception. *Computational and Mathematical Methods in Medicine*, 2015, Article 151702. https://doi.org/10.1155/2015/151702