A Distributed Simulation Approach to Integrate AnyLogic and Unity for Virtual Reality Applications: Case of COVID-19 Modelling and Training in a Dialysis Unit

Jalal Possik, Simon Gorecki, Ali Asgary, Adriano Solis, Grégory Zacharewicz, Mohammadali Tofighi, Mohammad Ali Shafiee, Asad Merchant, Mehdi Aarabi, Abel Guimaraes, et al.

A Distributed Simulation Approach to Integrate AnyLogic and Unity for Virtual Reality Applications: Case of COVID-19 Modelling and Training in a Dialysis Unit

Jalal Possik[a,b], Simon Gorecki[c], Ali Asgary[a,b], Adriano O. Solis[a,b], Gregory Zacharewicz[d], Mohammadali Tofighi[a], Mohammad Ali Shafiee[e], Asad A. Merchant[e], Mehdi Aarabi[e], Abel Guimaraes[a], Nazanin Nadri[a]

[a]*Advanced Disaster, Emergency and Rapid Response Simulation (ADERSIM), York University, Toronto, Ontario, M3J 1P3, Canada*

[b]*School of Administrative Studies, York University, Toronto, Ontario, M3J 1P3, Canada*

[c]*Groupe de Recherche en Economie Théorique et Appliquée (GREThA) UMR CNRS 5113, University of Bordeaux, 33608 Pessac, France*

[d]*Laboratoire des Sciences des Risques (LSR), Institut Mines-Telecom (IMT) Mines Ales, CEDEX, 30319 Alès, France*

[e]*Division of General Internal Medicine, Department of Medicine, Toronto General Hospital, Toronto, Ontario, Canada*

{jpossik, asgary, asolis, tofighim}@yorku.ca, simon.gorecki@u-bordeaux.fr, gregory.zacharewicz@mines-ales.fr, {mohammad.shafiee, asad.merchant}@uhn.ca, mehdi.aarabi@gmail.com, abelguima@gmail.com, nazaninnadri1@gmail.com

***Abstract*—Different heterogeneous simulation components can be integrated to produce a more effective complex global system. The IEEE High-Level Architecture (HLA) is an international standard that promotes interoperability and reusability for distributed simulation (DS). This paper proposes a DS system that integrates an agent-based and discrete-event simulator with a 3D game engine to build virtual reality (VR) applications that replicate real environments. In this case study, AnyLogic is used as an agent-based and discrete event simulator to simulate the process flow and COVID-19 transmission inside the University Health Network dialysis unit, Toronto, Canada. Unity game engine delivers the 3D modelling replicating the real architecture and environment of the dialysis unit. The HLA standard plays a major role in the integration of AnyLogic and Unity to produce a more effective and powerful DS system for VR applications.**

Keywords—Modelling and simulation, distributed simulation, virtual reality, AnyLogic, Unity, HLA, COVID-19, hemodialysis, dialysis unit

I. Introduction

The Modelling and Simulation (M&S) approach encompasses broad techniques and methods to replicate the behavior of real systems. The integration of M&S helps in solving complex problems and quickly anticipating scenarios at considerably reduced costs. It is an efficient way to innovate while avoiding costs or minimizing risk of accident or material damage [1]. In addition, submitting data to a digital model is much simpler than testing the implications of various data in real-life situations. M&S is becoming a commonly used approach in different domain and industries. The role of M&S in the healthcare domain has been widely established and is currently one of the commonly applied approaches to solving healthcare management problems. M&S enables pathways for program development, program and procedural expansion, and the introduction of new medical programs. This will be especially important for new hospital builds, as well as in improving hospital capacity and resource planning.

One of the most used paradigms in the last two decades is Agent-Based Modelling (ABM), which first appeared in 1990 [2]. It is used to represent real scenarios and simulate the behaviour of autonomous entities called agents, in order to study their effect on the overall system. However, according to Grigoryev [3], the first effective use of ABM was in 2000. In ABM, an agent represents an independent entity (which can be a human, a vehicle, a machine, etc.). Nowadays, agent-based systems are mostly used for simulating social, economic, political, and human behaviours [4], [5]. Therefore, several applications of ABM in the context of urban health issues and infectious disease epidemiology have already been reported in the literature [6]–[8]. Agent-based modelling and simulation (ABMS) brings several advantages compared to standard simulation techniques – e.g., system dynamics (SD), discrete-event simulation (DES). With ABMS models, each agent can have its own attributes and behaviours, which collectively give rise to the system's overall behaviour [5]. However, ABMS requires more memory and CPU capacity than a classical discrete-event simulation. [3]. Notwithstanding constantly improving, more powerful computer capacities and resources (CPU and memory evolution), the complexity of models developed to characterize agent-based systems continues to present computational challenges. This limitation brings the need for distributed technologies into the agent-based domain. The principle of Distributed Simulation (DS) is to execute a simulation on multiple computers connected to the same network. Each component can be located on a different distributed computer/CPU but is part of a single main simulation. Division of complexity can be done on several clusters, servers, computers, virtual machines, or threads and can solve complex processes with the distribution of calculation. That is why, in some case studies, DS can be a good solution to split and relocate complexity into several interconnected components.

In this article, we present a DS system developed to simulate the dialysis unit at Toronto General Hospital, one of ten medical programs/centres of the University Health Network (UHN) which are located at eight different sites in Toronto, Canada. The DS architecture is built based on the IEEE High-Level Architecture (HLA) standard. This standard ensures the communication and data exchange between heterogeneous components. The main agent-based model is implemented under AnyLogic (version 8.7.4) [3], [9] to simulate COVID-19 transmission between patients, physicians, nurses, and staff of the dialysis unit. Adding too many elements to an AnyLogic

model induces slowdowns. Moreover, 3D animation will also make the simulation execution too slow. Accordingly, we introduce Unity3D as a game engine connected and synchronized to the process flow of AnyLogic simulation model. Unity displays, in parallel and real-time 3D animation, the running process of AnyLogic. In order to ensure the communication between AnyLogic and Unity, we developed a custom extension to both applications in order to make them HLA compliant. The concept and architecture of the developed DS are discussed in the following sections.

II. State of the Art

Information and communication technologies are proving more and more essential for the healthcare domain, which is facing structural changes, a rise in technology use, and planning for improvement of patients' outcomes [10]. With the constantly increasing costs, disorganization, and quality failures experienced by healthcare providers and patients, a better understanding is needed of the knowledge sharing and reuse between various healthcare systems, applications, devices, and other technological components. Interoperability is critical for improving the standard of care and reducing healthcare costs [11].

One of the main concerns in DS relates to the need to integrate multiple heterogeneous simulators that work in different simulation environments. For example, most of the early DS projects in the defense domain were designed to train the military at a lower cost and without risk [12], [13]. A variety of simulators, such as the aircraft, ground vehicles, and ships simulators have been combined to build a complete virtual environment for real and hypothetical scenarios [14]. DS systems are becoming increasingly complex in terms of both the dynamics and the heterogeneity of components of the system. The degree of heterogeneity is at four main levels: (i) data heterogeneity, (ii) middleware heterogeneity, (iii) application heterogeneity, and (iv) non-functional heterogeneity [15].

Much research has been focused on defining methods and techniques that address the problems of reuse and interoperability between heterogeneous simulation models and the execution of these models in a distributed environment [16]–[19]. Since the late 1980s, several architectures have focused on the need to link different simulations. These efforts were mainly aimed at providing a consistent way of reusing existing simulations in some new combinations in order to avoid developing a limited and monolithic architecture that cannot meet all the goals and requirements of the simulation [19]–[21]. Currently, the HLA standard originally developed by the US Department of Defense (DoD) is considered to be one of the most widely used popular standards primarily used for solving interoperability and reusability problems in distributed simulation [16], [22], [23]. HLA is an established standard used by many researchers and engineers to solve their interoperability problems in the industry [24]–[28].

In the following sections, we introduce the methodologies and solutions that are used to solve the interoperability challenges and difficulties that engineers, particularly in the healthcare sector, face. We propose a method of connecting and exchanging data between AnyLogic and Unity. AnyLogic is the platform used for ABMS implementation and Unity for 3D development and implementation. The method and integration of the two distributed heterogeneous environments are described in detail in the next section.

III. Materials and Methods

The need for interoperability among heterogeneous simulation components has been tackled by the IEEE HLA which is a distributed modelling standard. It opens the door to multi-model development, data exchange, interoperability, reuse, and communication with external systems. This section introduces the process of synchronous simulation.

A. HLA

The first release of this standard was HLA US DoD 1.3 [29]. Then it was adopted by IEEE in 2000 and named HLA IEEE 1516. In 2010, it was revised and updated, including improvements; this latest version is called HLA Evolved. The next version of HLA (HLA 4) for M&S is currently under development. It will have new system modelling functions and new safety-related modelling functions [30, p. 4]. The HLA standard contributes to the development of DS, which works by creating simulations composed of various simulation components. These components are called “Federates”. A federation consists of multiple federates, one runtime infrastructure (RTI), and a federated object model (FOM). RTI provides a set of standardized services for information and data exchange, synchronization and joint management. FOM defines the objects and interaction classes used for communication. The main purpose of HLA is to provide modelling compatibility and reuse models. These models run on separate computers with different operating systems, implemented in different programming languages, and distributed on a LAN or a WAN. All these different components are unified in a federation. With the help of a publish/subscribe mechanism (p/s) based on FOM and HLA objects management, these various heterogeneous components (the so-called federates) can exchange data and information (Fig. 1).

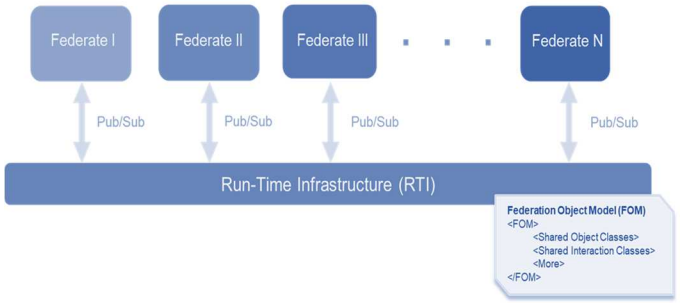

Fig. 1. HLA connected federates

An HLA component can communicate with other federates with a mechanism of p/s to an object class. An object class has a set of attributes that can be updated using the method *updateAttributeValues()* of the RTIAmbassador class. In the same way, an HLA component can p/s to ephemeral components which are interactions/parameters using *sendInteraction()* method. All HLA components subscribed to an object can receive an update of those attributes, and all federates subscribed to an interaction will receive the interaction when it is published.

In Fig. 2, we can see the Business Process Modelling and Notation (BPMN) flowchart, which shows an example of the communication between three connected federates through the HLA objects and attributes. The first federate updates an object with *updateAttributeValues()* method. This action results in an RTI callback to update all connected federates through the *reflectAttributeValues()* method. Finally, the second federate sends an interaction to the RTI ambassador which triggers an RTI call back to all connected federates through the method *receiveInteraction()*.

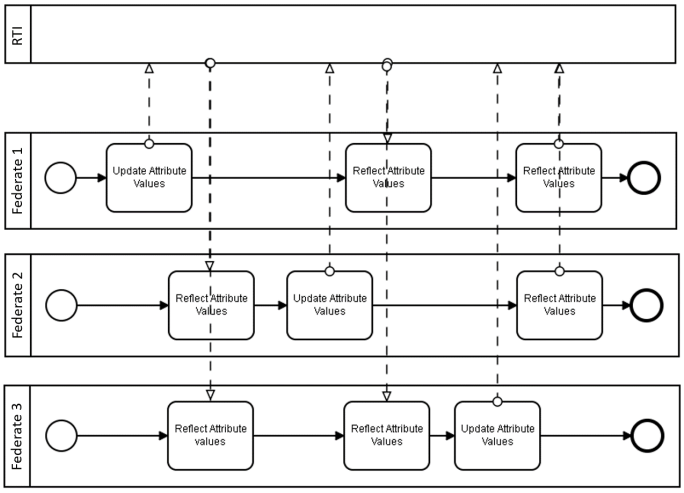

Fig. 2. Update/reflect attribute values between HLA federates

In this research study, the objective is the integration of two applications: Unity and AnyLogic. Unity is a game engine built on C++ language. Scripts and codes inside Unity are written in C#, Javascript/UnityScript and sometimes in Boo. However, AnyLogic is a simulation engine written in Java. To solve the interoperability problem, a top layer is added over both Unity and AnyLogic.

B. System Architecture

The communication between Unity and AnyLogic is ensured with the p/s mechanism of objects/interactions, provided by HLA. An FOM is a file that describes all shared objects and interactions between components. Each federate can send and receive data declared in the FOM file. In this study, the Pitch pRTI [31] Java library is used for HLA development. Each of the components (AnyLogic and Unity) has a developed HLA interface. Both of them can thus create a federate and create/join a new or existing federation. The Federation Development and Execution Process (FEDEP) allows to avoid the early phases errors and reduces the cost and time of simulation development. In this system, there is no need to implement the HLA time management mechanism; both federates are neither time-constrained nor time-regulating. Fig. 3 shows AnyLogic and Unity connected as federates to the same RTI instance. Thanks to the pRTI, the development of the HLA interface between Unity and AnyLogic becomes easier. However, the development of an HLA interface for Unity is more complex. To enable interoperability between .NET and Java, IKVM [32], a .NET implementation of a Java Virtual Machine, is used. DLL files are exported from the Java HLA code and then imported into Unity to make them HLA compatible. After this step, creating/joining a federation, publishing/subscribing data, and sending/receiving messages are enabled in C# for Unity.

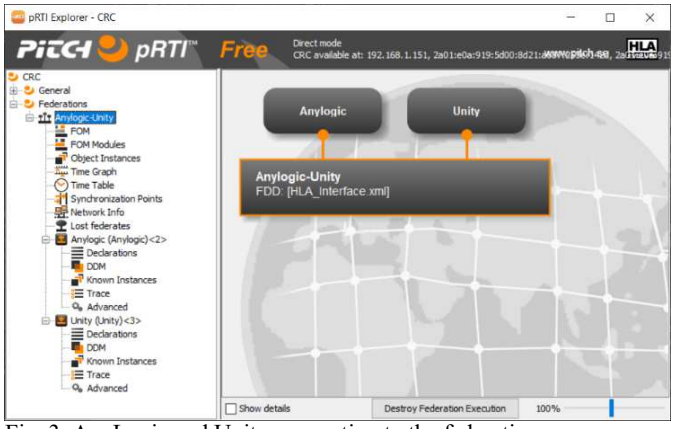

Fig. 3. AnyLogic and Unity connection to the federation

The safety and quality improvements of the dialysis unit, especially during the COVID-19 pandemic period, are the main goal of this DS system. In addition, a virtual replicated environment (process flow, COVID-19 transmission, and 3D visualization) is essential to train the medical team preparatory to improving patient safety and experience. Moreover, this DS works as a decision-aided tool to improve the flow performance inside the unit, especially during the COVID-19 pandemic period where some rooms are reserved only for patients who have tested positive for COVID-19. Thus, several goals are targeted:

1) the optimization of patient safety and flowrate,
2) the improvement of the patient experience, and
3) the increased efficiency of the unit.

C. Simulation Case Study and Results

The platform and simulation models are based on the dialysis unit of the UHN. The UHN dialysis unit has a total capacity of 55 dialysis chairs. This unit serves 308 patients, of whom 278 undergo intermittent hemodialysis three to six days a week, and 30 are nocturnal/evening dialysis patients who are dialyzed three days a week. There are three daily shifts and a nocturnal shift from Mondays to Saturdays.

The hemodialysis unit is staffed by nephrologists, nurses, technicians, pharmacists, dieticians, social workers, and other clerical and support staff. There are 3-day time shifts: 7:30 – 12:30, 13:00 to 17:00 and 17:30 to 21:30, and one nocturnal shift from 23:00 to 6:00. Patients in each shift come in a staggered fashion, with each group coming in approximately 30 minutes after the one preceding it. Patients for any given shift are brought into the unit in 3 or 4 cohorts. The first cohort of patients is brought in by the clerk who goes out to the waiting area and gives the patients their wrist band identifications. The patients walk in and, after weighing themselves, go to their assigned stations where nurses are waiting. By the time, the first cohort patients are settled onto the machine and have been assessed and their blood pressures checked, the second cohort will be allowed to come in, ushered by the clerk. Patients are being asked to line up in the corridor at 2 m distances outside the doorway.

One nurse will start dialysis for 3 or 4 patients and will call in the next patient as soon as they have finished setting up a patient. After patients are done with their dialysis, the assigned nurses will take them off the machine, decannulate their arteriovenous (AV) fistula or central venous catheter, and check their blood pressure, sitting, and standing. The process of starting and stopping dialysis for a patient takes between 15 to 20 minutes each. Nurses are given their assignments on the day of their shift. Nurses are assigned to specific dialysis stations, and not to patients. There is no consistency regarding which stations they are assigned to. They are randomly distributed. Patients are not usually assigned to the same stations, although some do have their preferred spot.

Before, or between the shifts, hemodialysis assistants (HAs) set up the dialysis machines. This includes putting on dialyzers and tubing onto the machine, ensuring the machines are cleaned and sterilized according to the protocol, and ensuring other equipment/gadgets are made available at the station to be used by the nurse and patient. There are three HAs per shift for each ward that prepare dialysis machines for the admitted patients that are dialyzed in their rooms in the hospital (referred to as off-unit). When they are not actively preparing machines, the HAs usually sit at nursing stations.

Ten different nephrologists are responsible for the 12 shifts (six on the east dialysis wing, and six on the west side). Two nephrologists are each responsible for two shifts. A nurse practitioner (NP) assists nephrologists for morning and afternoon shifts. Nephrology fellows are also assigned to conduct rounds in the unit on behalf of the most responsible physician (MRP). The MRP spends approximately 1-2 hours in the dialysis unit, once a week, while the NP and/or the nephrology fellow spend approximately 1-2 hours per shift rounding on the patients on both sides (East and West). When the NPs or fellows are not actively seeing patients, they may stay in their office area, or leave the unit.

The BPMN flowchart of Fig. 4 shows the patient flow from the arrival to the discharge process.

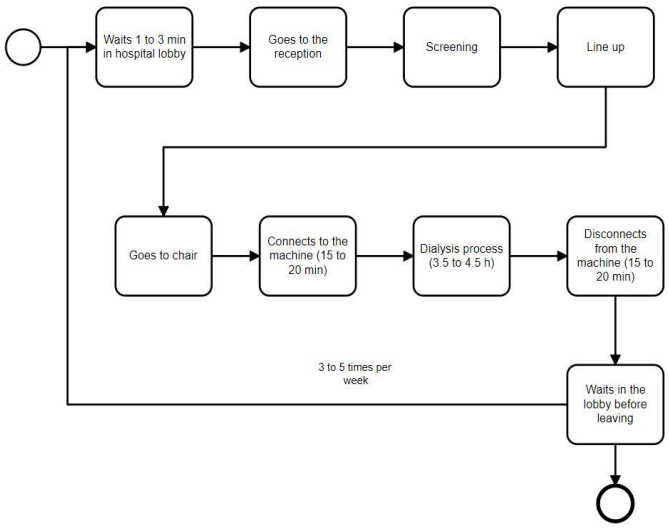

Fig. 4. Patient flow in the dialysis unit

The hemodialysis process flow is built on AnyLogic for 2D tracking and experimentation. In the 2D dashboard, the user tracks the movements of agents (patients, nurses, nephrologists, clerks, assistants, technicians, and MRPs) inside the hemodialysis unit, the number of available beds, the simulation time, and other features.

A modified version of the SEIR (Susceptible, Exposed, Infected, and Recovered) disease transmission model [33]–[35] is used in this research. Additional compartments are added in the SEIR model to include symptomatic, asymptomatic, and pre-symptomatic cases in addition to hospitalization, self-isolation, and deceased compartments. Each of the agents can be in one of the aforementioned states. Moreover, a statistics dashboard is developed to track the COVID-19 transmission inside the unit (Fig. 5).

AnyLogic broadcasts the time-stamped and encrypted agent (patient, nurse, physician, etc.) data: ID, Type, and Position (x, y, z) to the RTI every 0.1 seconds using the HLA publish/subscribe mechanism. Unity, which is configured as an HLA federate connected to the RTI and subscribed to the published data, obtains the position and based on the received type and ID, assigns it to the appropriate agent.

Fig. 5. 2D and statistics dashboard of AnyLogic

Using Unity particle systems, we visualize how a virus might spread from an infected person to nearby agents and surfaces, based on the research of Jankovic [36] (Fig. 6).

Three particle generating objects attached to the mouth of a virtual infected patient are used. Regular breathing, sneezing,

and coughing are all represented by different particle systems. For regular breathing, we imitate a person's breathing every three seconds with particles moving at a speed of 0.3 meters per second within a one-meter radius. In the case of sneezing, we use a speed of five to ten meters per second for particles having a radius of eight meters. The coughing is modelled with varied frequencies in which particles spread at a speed of one meter per second and travel within a radius of up to two meters.

Fig. 6. Unity particle systems

A Unity script, attached to objects, is developed to count the number of particles interacting with the avatar or object in order to compute the number of particles reaching each virtual object's surface or body. During the simulation run, the number of particles is transmitted from Unity to the RTI via the publish mechanism and received by AnyLogic via the subscribe mechanism of HLA.

Fig. 7 shows the 2D representation of the dialysis unit generated using AnyLogic running in parallel with the 3D model developed on Unity.

Fig. 7. Unity 3D/AnyLogic 2D representation

IV. Discussion

Scientists from all over the world have submitted papers on M&S, artificial intelligence, computational science, and other technologies to help in winning the fight against COVID-19, e.g. [37]–[40], among many others.

In this paper, we describe a DS system that was developed for a hemodialysis unit in Toronto, Canada, to provide hospital managers and physicians with predictive analytics, visual models, and VR applications that help in medical training and the discovery of solutions that improve the safety and quality of services, particularly during the COVID-19 pandemic.

AnyLogic is a leading simulation software for process modelling and simulation. Unity is a renowned tool for creating 3D projects and virtual reality applications. In this project, we integrate both, AnyLogic and Unity, based on HLA IEEE standard and architecture to build up a more powerful DS system. This DS enables AnyLogic and Unity to run simultaneously in parallel and collaborate encrypted data in both directions (AnyLogic to Unity and Unity to AnyLogic).

The hemodialysis case study has been successfully implemented and tested on this DS system. AnyLogic is used to replicate the hemodialysis process flow from admission to discharge. In addition, a dashboard is developed using AnyLogic in order for managers and decision makers to track and experiment the agents' movements inside the hemodialysis unit, the number of available beds, the simulation time, and other features. During the simulation run, AnyLogic sends encrypted data (time stamped agent's ID, type, and position) to Unity via the HLA interface. Unity, in turn, decrypts the data and updates the agents' attributes accordingly. Unity, through its developed HLA interface, also sends certain data to AnyLogic, such as the number of virus particles.

Use of the HLA standard enables the integration of two powerful and leading software applications, allowing VR applications to run in parallel with process flow simulation. Moreover, HLA opens the door for AnyLogic or/and Unity to connect with other HLA-compliant applications.

V. Conclusion

The main goal of this study is to design a DS system for the healthcare domain, which has been confronted with significant challenges due to the COVID-19 epidemic. The HLA standard is primarily utilized to address interoperability concerns between AnyLogic, as the platform used for agent-based and discrete event M&S, and Unity as a visualization engine. The final result of this DS system is a VR application with AnyLogic controlling the ABM and process flow and the Unity engine operating the graphics and 3D replicated environment in parallel.

By bringing real players into the virtual environment of the dialysis unit, we will be able to tailor their reactions and experience. Patients, nephrologists, nurses, assistants, technicians, and clerks can all be characters in the game. Thus, a reinforcement learning training setup can be implemented with the model's existing agents by integrating the behaviours and activities of real players, allowing for more efficient training.

Acknowledgment

We thank the Canadian Institutes of Health Research (CIHR), the University Health Network (UHN), and the Public Health Agency of Canada (PHAC) for their financial support of our research.

References

[1] A. O. Solis, F. Longo, L. Nicoletti, P. Caruso, and E. Fazzari, "A modelling and simulation approach to assessment of a negative binomial approximation in a multi-echelon inventory system," *Int. J. Simul. Process Model.*, vol. 9, no. 3, pp. 146–156, 2014.

[2] P. O. Siebers, C. M. Macal, J. Garnett, D. Buxton, and M. Pidd, "Discrete-event simulation is dead, long live agent-based simulation!," *J. Simul.*, vol. 4, no. 3, pp. 204–210, Sep. 2010, doi: 10.1057/jos.2010.14.

[3] I. Grigoryev, *AnyLogic 8 in Three Days*, 5th ed. 2018. [Online]. Available: https://www.anylogic.com/upload/al-in-3-days/anylogic-in-3-days.pdf

[4] C. M. Macal, "Everything you need to know about agent-based modelling and simulation," *J. Simul.*, vol. 10, no. 2, pp. 144–156, May 2016, doi: 10.1057/jos.2016.7.

[5] C. M. Macal and M. J. North, "Tutorial on agent-based modelling and simulation," *J. Simul.*, vol. 4, no. 3, pp. 151–162, Sep. 2010, doi: 10.1057/jos.2010.3.

[6] R. Bagni, R. Berchi, and P. Cariello, "A comparison of simulation models applied to epidemics," *J. Artif. Soc. Soc. Simul.*, vol. 5, no. 3, 2002.

[7] I. Mahmood *et al.*, "FACS: a geospatial agent-based simulator for analysing COVID-19 spread and public health measures on local regions," *J. Simul.*, pp. 1–19, Aug. 2020, doi: 10.1080/17477778.2020.1800422.

[8] Y. Meng, R. Davies, K. Hardy, and P. Hawkey, "An application of agent-based simulation to the management of hospital-acquired infection," *J. Simul.*, vol. 4, no. 1, pp. 60–67, Mar. 2010, doi: 10.1057/jos.2009.17.

[9] AnyLogic Company, *AnyLogic 8.7.4 University version*. 2020. [Online]. Available: https://www.anylogic.com/

[10] A. H. Harris, "Path from predictive analytics to improved patient outcomes: a framework to guide use, implementation, and evaluation of accurate surgical predictive models," *Ann. Surg.*, vol. 265, no. 3, p. 461, 2017.

[11] O. Iroju, A. Soriyan, I. Gambo, and J. Olaleke, "Interoperability in healthcare: benefits, challenges and resolutions," *Int. J. Innov. Appl. Stud.*, vol. 3, no. 1, pp. 262–270, 2013.

[12] R. M. Fujimoto, *Parallel and distributed simulation systems*, vol. 300. Citeseer, 2000.

[13] A. Tolk, *Engineering principles of combat modeling and distributed simulation*. Wiley Online Library, 2012.

[14] N. R. Council and others, *Defense modeling, simulation, and analysis: meeting the challenge*. National Academies Press, 2006.

[15] G. S. Blair, M. Paolucci, P. Grace, and N. Georgantas, "Interoperability in complex distributed systems," in *International School on Formal Methods for the Design of Computer, Communication and Software Systems*, 2011, pp. 1–26.

[16] A. Garro and A. Falcone, "On the Integration of HLA and FMI for Supporting Interoperability and Reusability in Distributed Simulation," in *Proceedings of the Symposium on Theory of Modeling & Simulation: DEVS Integrative M&S Symposium*, San Diego, CA, USA, 2015, pp. 9–16.

[17] M. Lees, B. Logan, and G. Theodoropoulos, "Distributed Simulation of Agent-Based Systems with HLA," *ACM Trans Model Comput Simul*, vol. 17, no. 3, pp. 11-es, Jul. 2007, doi: 10.1145/1243991.1243992.

[18] S. Gorecki, J. Possik, G. Zacharewicz, Y. Ducq, and N. Perry, "A Multicomponent Distributed Framework for Smart Production System Modeling and Simulation," *Sustainability*, vol. 12, no. 17, 2020, doi: 10.3390/su12176969.

[19] J. Possik, A. Amrani, and G. Zacharewicz, "WIP: Co-simulation system serving the configuration of lean tools for a manufacturing assembly line," presented at the Proceedings of the Works in Progress Symposium, WIP, 2018.

[20] K. L. Morse, M. Lightner, R. Little, B. Lutz, and R. Scrudder, "Enabling simulation interoperability," *Computer*, vol. 39, no. 1, pp. 115–117, Jan. 2006, doi: 10.1109/MC.2006.17.

[21] S. Gorecki, J. Possik, G. Zacharewicz, Y. Ducq, and N. Perry, "Business Models for Distributed-Simulation Orchestration and Risk Management," *Information*, vol. 12, no. 2, 2021, doi: 10.3390/info12020071.

[22] A. Falcone, A. Garro, S. J. E. Taylor, and A. Anagnostou, "Simplifying the development of HLA-based distributed simulations with the HLA Development Kit software framework (DKF)," in *2017 IEEE/ACM 21st International Symposium on Distributed Simulation and Real Time Applications (DS-RT)*, 2017, pp. 1–2. doi: 10.1109/DISTRA.2017.8167691.

[23] J. Huang, X. Zhao, J. Hao, and R. Ju, "Brief Introduction of KD-HLA: An Integrated Environment to Support M amp;S Based on HLA," in *2010 Second International Conference on Computer Modeling and Simulation*, 2010, vol. 2, pp. 281–283. doi: 10.1109/ICCMS.2010.498.

[24] S. Gorecki, Y. Bouanan, G. Zacharewicz, J. Ribault, and N. Perry, "Integrating HLA-Based Distributed Simulation for Management Science and BPMN," *IFAC-Pap.*, vol. 51, no. 11, pp. 655–660, 2018.

[25] J. Possik, "Contribution to a Methodology and a Co-Simulation Framework assessing the impact of Lean on Manufacturing Performance," PhD Thesis, Bordeaux, 2019.

[26] G. Zacharewicz, C. Frydman, and N. Giambiasi, "G-DEVS/HLA environment for distributed simulations of workflows," *Simulation*, vol. 84, no. 5, pp. 197–213, 2008.

[27] B. P. Zeigler, S. B. Hall, and H. S. Sarjoughian, "Exploiting HLA and DEVS to promote interoperability and reuse in Lockheed's corporate environment," *Simulation*, vol. 73, no. 5, pp. 288–295, 1999.

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

[29] F. Kuhl, R. Weatherly, and J. Dahmann, *Creating computer simulation systems: an introduction to the high level architecture*. Prentice Hall PTR, 1999.

[30] B. Möller, M. Karlsson, R. Herzog, and D. Wood, *Security in Simulation - New Authorization Opportunities in HLA 4*. 2021.

[31] Pitch Technologies, *pRTI*. Pitch Technologies, 2021. [Online]. Available: www.pitch.se

[32] J. Frijters, *IKVM*. 2014. [Online]. Available: http://www.ikvm.net/

[33] H. W. Hethcote and D. W. Tudor, "Integral equation models for endemic infectious diseases," *J. Math. Biol.*, vol. 9, no. 1, pp. 37–47, 1980.

[34] H. W. Hethcote, H. W. Stech, and P. Van Den Driessche, "Nonlinear oscillations in epidemic models," *SIAM J. Appl. Math.*, vol. 40, no. 1, pp. 1–9, 1981.

[35] H. W. Hethcote, "Measles and rubella in the United States," *Am. J. Epidemiol.*, vol. 117, no. 1, pp. 2–13, 1983.

[36] L. Jankovic, "Experiments with Self-Organised Simulation of Movement of Infectious Aerosols in Buildings," *Sustainability*, vol. 12, no. 12, p. 5204, 2020.

[37] G. Wainer, K. Hinsen, and K. Gaither, "Computational Science in the Battle Against COVID-19--Part II," *Comput. Sci. Eng.*, vol. 23, no. 01, pp. 5–6, Feb. 2021, doi: 10.1109/MCSE.2020.3048123.

[38] A. Asgary, M. G. Cojocaru, M. M. Najafabadi, and J. Wu, "Simulating preventative testing of SARS-CoV-2 in schools: policy implications," *BMC Public Health*, vol. 21, no. 1, Art. no. 1, Dec. 2021, doi: 10.1186/s12889-020-10153-1.

[39] M. Tofighi *et al.*, "Modelling COVID-19 Transmission in a Hemodialysis Centre Using Simulation Generated Contacts Matrices," Infectious Diseases (except HIV/AIDS), preprint, Jan. 2021. doi: 10.1101/2021.01.03.21249175.

[40] R. Cárdenas, K. Henares, C. Ruiz-Martín, and G. Wainer, "Cell-DEVS Models for the Spread of COVID-19," *Proceedings of the 14th International Conference on Cellular Automata for Research and Industry*, 2020, pp. 239-249.