INTEROPERATING WARGAME SIMULATIONS

Pier-Giovanni Taranti[1,2], Ricardo Choren[3], Carlos Lucena[4], Michael Luck[2], and Peter McBurney[2]
[1]CASNAV/Brazilian Navy – Rio de Janeiro/RJ, Brazil
[2]King's College London – London, UK
[3]SE/8 - IME, Praça General Tibúrcio, 80 – Rio de Janeiro/RJ, Brazil
[4]PUC-Rio, Rua M. de São Vicente 225 – Rio de Janeiro/RJ, Brazil

Resumo

Jogos de Guerra são uma ferramenta inestimável para os militares. Eles podem apoiar uma gama de atividades, tais como planejamento, treinamento, educação e avaliação de cenário. Simuladores têm sido desenvolvidos para uso em Jogos de guerra a fim de prover a capacidade de representar a dinâmica do cenário. Atualmente, os paradigmas da guerra têm se alterado devido ao crescente número de operações conjuntas envolvendo diferentes atores. Para apoiar estes novos paradigmas são necessários Jogos de Guerra conjuntos. Contudo, os simuladores utilizados pelos Jogos de Guerra normalmente não possuem a capacidade de interoperar. Existem problemas envolvendo os conceitos compartilhados, protocolo de comunicação e sobre o avanço do tempo de simulação, por exemplo. O avanço do tempo de simulação é um desafio, uma vez que os simuladores são rotineiramente projetados como scaled-time com a capacidade de acelerar a simulação a fim de manter a dinâmica do jogo e evitar que os jogadores fiquem desocupados. Devido à falta de recursos computacionais, um simulador pode não ser capaz de seguir a nova taxa de avanço, mais rápida que a anterior. Este trabalho apresenta uma arquitetura projetada para permitir o desenvolvimento de simulações de ambientes virtuais compostas por simuladores scaled-time com a habilidade de alterar a taxa de avanço da simulação em tempo de execução. A arquitetura adapta a taxa de acordo com informação de retorno recebida pelos simuladores agregados. São também apresentados uma implementação da arquitetura em uma solução multiagente aplicada a um cenário naval.

Palavras-Chaves: Jogos de Guerra; Simulações de Jogos de Guerra; Interoperabilidade de Jogos de Guerra.

Abstract

Wargames are an invaluable tool for military. They can support a range of activities, such as planning, training, education and scenario evaluation. Simulators have been developed in order to provide wargames with the ability to represent the scenario dynamics. Currently, the number of joint operations involving different actors are increasing, changing the war paradigms. In order to support these new paradigms, we need joint wargames. However, the wargames' associated simulators are frequently unable to interoperate. There are concerns about shared concepts, communication protocol and simulation-time advance, as examples. The simulation-time advance is a challenge, since these simulators are frequently designed as scaled-time with the ability of speeding-up the simulation, in order to maintain wargame dynamics and to avoid players being idle. Moreover, due to the lack of computational resources, a simulator may not be able to follow a faster rate. This paper presents an architecture designed to allow the development of virtual environment simulations composed of scaled-time simulators with the ability to change the simulation-time advancing rate. The architecture adapts the simulation rate according to feedback information provided by the joint simulators. The paper also presents the architecture implementation in a multi-agent solution and its usage in a naval scenario.

Keywords: Wargame; Wargame Simulation; Wargame Interoperability.

1. INTRODUCTION

A wargame is a warfare model or a simulation whose operation does not involve actual military forces, and whose sequence of events affects and is affected, in turn, by decisions made by players representing opposite sides [12, 28]. In a nutshell, in a wargame, we have players who will command simulated units in a mock operation. These games are an invaluable tool for the military forces, and are applied in activities such as training, education and analysis. Wargames can be classified as serious games [3], and their concepts have been applied to areas other than the military, such as politics, logistics and business [12, 14, 28, 32–34].

Warfare paradigms have changed since the last global conflict when wargames were widely applied. Nowadays, information flows quickly, and concepts such as network centric warfare (NWC) [5], power to the edge [6] and effect-based operations [35] have flourished in the past two decades, following not only technological advances, but also new menaces and actors in war. Additionally, non-military actors and asymmetric warfare are a new reality on the battlefield, which is sometimes referred to as a complex endeavour [4]. As a consequence of these moves, military forces need to interoperate in an integrated and live command and control (C2) system: for example before and during an air raid, the commander of the air force component needs to be informed in time about naval units, non-governmental actors, enemy movements, and so on. In the event of a pilot being lost, the support of the special operations component on the field could be immediately required. This integration, here exemplified at the tactical/operational level, is related to the concept of interoperability.

The interoperability of C2 systems is an active field of research, mainly centred on technical integration of existing systems. However, the integration of C2 software systems does not ensure interoperability, for which new doctrines are needed with associated procedures. It is needed to test and polish these doctrines and procedures, and to train officers on it. Furthermore, the most suitable tool to perform these activities is a wargame. During the game, it is possible to test new ideas, have insights and change decisions at a cost much fewer than in a real environment. Once the procedures are defined, the same wargame is useful to train and educate military personnel.

However, this implies performing joint wargames, with each component or actor of the operation's theatre interacting with others. Performing these joint games is not a simple task: wargames have been developed over decades and are mainly centred on one specific component. The air force, army and navy have different cultures and concepts, and these are represented in their C2 systems, and also in the wargames they use. These wargames have specificities from a higher level of abstraction, related to procedures, concepts used in the games, to a lower technical level, related to the simulator and the solution it uses to generate the simulated environment. To create such joint wargames, with the consequent shared environment, the following are necessary: a communication protocol, to coordinate the information flow between simulators; a data model able to represent the needed shared concepts; and a mechanism to handle the advance of the overall simulation or, in other words, to advance the simulation-time, which is the logical abstraction of the physical-time in the simulation [16].

The problem of advancing the simulation-time in joint wargames is a challenge, mainly because the involved simulations are frequently scaled-time [29]; that is, the simulation-time is advanced following a linear relation to the advance of the physical-time. Furthermore, wargames tend to have implemented mechanisms to speed up the simulation when possible, in order to optimize the resources employed on it. While there are solutions and industry standards to interoperate simulations; these are not able to handle the changing of the simulation-rate at execution time, for example.

Another issue that arises from this situation of joint scaled-time simulations is that all the simulators need to advance their representation of the simulated environment (simulation

models) according to the current rate. However, some simulators may not be able to follow the stated rate due to a lack of computational resources. Thus, inconsistent states could appear, with simulators showing elements at different times (i.e past, present and future) as if they were on the same temporal snapshot.

To address this problem, this paper presents an architecture that provides a central simulation clock for joint scaled-time simulations and can adapt the simulation-rate to a suitable value at execution time, using feedback provided continuously by the joint simulators. The work is presented as follows: the next section discusses the problem and its underlying technical challenges; Section 3 presents the suggested architecture and an example of implementation. Section 4 shows a conceptual example of use and in Section 5 we discuss the results. The conclusions are presented in Section 6 .

2. THE PROBLEM

We present here some general concepts about wargames and a brief discussion about the issues that interfere with interoperability. Afterwards, a short discussion about the supporting simulators and the challenges for their interoperation is also presented.

2.1. WARGAMES

Wargames, as we understand them, came from the Koenigspiel (1664 AD) and the more recent Kriegsspiel (1811 AD), and have been evolving in different ways in military forces and countries. The books from Perla [28, 12] and Sabin [33] provide an excellent overview of the theme.

There are wargames played on tables, where opponents play in turns, taking decisions and following some set of rules. These games allow a fruitful brainstorm, and are a common tool used by military staff for planning and intelligence analysis, and military history study. Some games are played on real-time simulations with the men-in-the-loop, such as flight simulators for the air force, tank and helicopter simulators to army or ships bridge simulators for navy. These simulations are mainly used for training and testing, frequently from the tactical to the operational level. Another type of game, which is the focus of our research, is played only by the military at the operational/strategic level, i.e by the commanders and their staff. All commanded military units are then simulated in a virtual environment, and the simulator takes care of evolving these units along the simulation-time, according to some plan and the interventions made by players and/or the empire - the game judge or director. These games can cover months of military operations and are played over a number of days, or even weeks, and their supporting simulation is classified as a constructive simulation by [16].

These strategic/operational wargames are highly expensive; however, the outputs that come from their execution are very important. The game infrastructure comprises technical people with a background in computer science, system modelling, military operations, logistics, communications, and so on. The examples we have verified during our research suggest that these games are conducted in a permanent building with dedicated rooms and hardware. The supporting simulator is dedicated software, developed to the specific problem. Some of these games have been evolving for more than 50 years.

The players' cost also influences the execution of wargames: these players are senior officers and advisers, and it is difficult to have such a group spared from their daily activities to run the game. Matching their agendas is a challenge and, because of this, except in exceptional situations, wargames are scheduled months in advance. The game agenda needs to be strictly respected, since the players will leave the game facility on the scheduled date.

Thus, in order to improve the cost/benefit relationship, a common procedure is to speed-up the game when no intervention is required from players. As an example, if an air raid has just been launched, and no action will be carried out before it bombs its target, the

game judge could increase the game advancement rate, avoiding the players being idle.

Wargames aim to insert the players in a situation that is possible in the real world, and military commanders are required to handle imprecise information, or even the lack of needed information, and to decide against the time in a dynamic environment. Thus, some level of imprecision is acceptable for wargames, such as delays in providing information to players. These delays are limited by the need in maintaining the feeling of reality.

2.2. SIMULATIONS FOR WARGAMES

To support this variation on the simulation-rate used for advancing the game, the associated simulator needs to be implemented as a scaled-time simulator with support for the changing of this rate at execution time. However, looking to the wargame interoperability problem, this approach prevents the use of industry standards for interoperating the underlying simulators. The most-used standards and solutions for simulation interoperability are HLA and DIS [36], and both of them were developed to interoperate real-time or scaled-time simulations at a fixed rate. Furthermore, DIS [1] is UDP based, using broadcast information over a local network, but wargame facilities are frequently placed at military training centres or at academies for command staff, rarely being close to one another.

Additionally, the simulation interoperability is affected by delays that can occur in the shared virtual environment. A possible consequence of these delays is having elements represented in the same virtual space but with different simulation times and being part of the context for each other [13]. For example, an entity representing a frigate could start engaging a fighter that flew past it minutes earlier and should now have faded from its sensors.

A number of factors can lead one simulator to be delayed in relation to others, such as: network delays; lack of computational power to face the established simulation-rate; and exogenous events (e.g interruptions to the simulator thread commanded by the operational system to answer an I/O call). A solution for interoperating in such a simulator needs to consider these delays. HLA [2], besides not supporting simulation rate variation, takes the approach of excluding from the simulation environment simulators that do not comply with the simulation-time advancing constraints. This approach is not acceptable in our problem domain, since the exclusion of one of the simulators implies in removing one military component from the joint game, such as the navy or the political component. This exclusion is likely to invalidate the wargame outputs. On the other hand, wargames can admit a relative level of delay, which adheres to the concept of virtual environment simulation (VES) presented in [16], that describes the precision to matching times when executing simulated events (or updates in time-directed simulations) as having flexibility. Furthermore, this flexibility in accepting delays is consistent with human perception of reality or to some constraint from the VES components. The VES concept is initially defined for real-time simulations, but we extend it to scaled-time simulations. In the cited work, Fujimoto also presents the concept of distributed virtual environment simulation (DVES) which comprises the creation of a shared virtual environment using a number of simulators that communicate with each other through a network. Thus, the problem could be re-stated as how to establish joint wargames as DVES.

An important information when designing a DVES is the characteristics of each of the joint simulators, since there are different approaches to represent dynamics of the simulated system, such as time-directed simulation (TDS) and discrete event simulation (DES). The same occurs when discussing the way in which the simulation-time is advanced: the most-used approaches are real-time, paced-time and as fast as possible [17, 21, 48]. It is impossible to ensure, before-hand, which approach is used by wargame simulators to be integrated.

The distribution of the current simulation-time to all DVES components is not a big challenge, since it could be achieved using existing techniques. On the other hand, ensuring that delays will remain at acceptable levels is an open problem from the real-time systems

field - see [20, 11] for more information about delays when executing scheduled actions. Furthermore, these delays are not caused at the application level, considering well-developed software, but by the inability to match the scheduled time for execution due to lack of resources.

Thus, consider the relation ***Ts = K.Tr***, where ***Ts*** is the simulation-time, ***K*** is a linear constant, i.e the simulation-time advancing rate, and ***Tr*** is the physical or real-time. A limited value for ***K*** will cause the physical-time available to perform the computation not to be sufficient, inducing tardiness due to the discussed lack of computational resources. Additionally, due to the dynamic nature of the computational environment, this limit-value for ***K*** changes over time. For example, I/O routines can interrupt the simulation process, using CPU time, and generating unexpected delays in this component of our DVES. In this way, it is not feasible to foresee an adequate maximum value for ***K*** before executing the simulation in a production environment.

The need to changing ***K***'s value at execution time to speed-up the wargame inserts more complexity into the possible solution. This is because the available time to execute some scheduled action will be shortened, which increases the likelihood of excessive delays. In contrast, when the value of ***K*** is reduced, the physical-time available to execute an action is dilated, decreasing the probability of delays.

In order to handle the problem of the simulation-time advancing in DVES applied to joint wargames, we have undertaken research in the works presented in [38–45], and that has led to the development of an architecture for implementing DVES with scaled-time and a variable simulation-rate.

3. THE PROPOSED ARCHITECTURE

In order to address the problem, we have specialized it to one instance, and then developed a solution which could be further applied to a more general case. The instance chosen was DVES developed with Multi-Agent Based Simulations (MABS). The decision was supported by the following factors.

Initially, agent technology facilitates an easy mapping between the concepts of real-system actors to software agents [15], and from the software engineering point of view, it has advantages in maintainability and evolution of the simulation model, since interventions can be executed directly in the agents, which are inherently loosely-coupled [25].

MABS also offers the possibility of modelling and instantiating the system at the micro-level, and observing the overall system behaviour afterwards, which is interesting when simulating complex systems [23, 30]. Additionally, DVES developed with MABS in which agents have control of their own execution thread presents extra complexity, since each simulator behaves internally as a parallel simulation, causing delays between its threads.

Finally, a number of mature MABS platforms are Java-based or provide support for the language [7, 22, 24, 37, 46], and Java is one of the most-used programming languages in industry. However, standard Java technology provides no guarantees about achieving schedule times by construction. Java programs run on virtual machines (VMs) and have no direct access to main memory. Furthermore, Java threads have no standard to define how they are matched to the operational system threads [26]. The mechanism that controls the memory used by each VM, called a garbage collector (GC), can suspend the Java program during the collection.

The proposed solution is composed by a multi-agent based architecture that is presented in Figure 1. The heart of the solution is a distributed simulation clock: each simulation that joins the DVES needs to comply with the simulation clock's interface. All joint simulations receive the current simulation-time and rate from this clock. Additionally, these joint simulations also need to provide feedback on their delays at an established frequency.

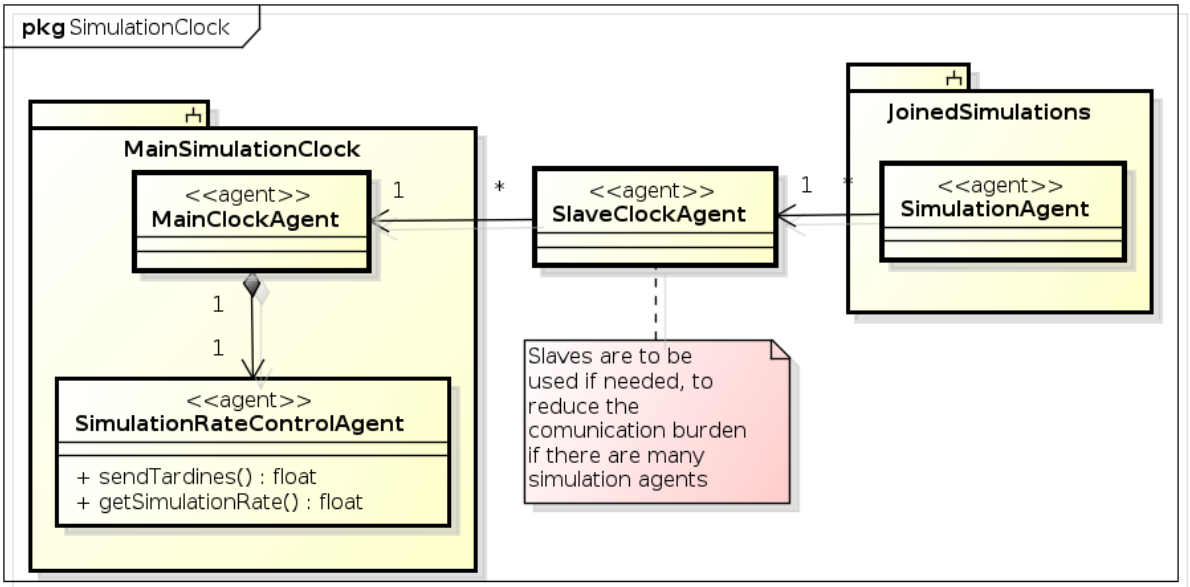

Figure 1: Architecture

The simulation clock is formed by the central clock and the slaves. The slaves are responsible for receiving the current simulation-time and rate from the central clock and for making this information available to the joint simulations, or simulation agents in a MABS. In the same way, these slave clocks collect information about delays and summarize it to the central clock.

The central simulation clock has similar features as the slaves, and, additionally, it runs the algorithm that adapts the simulation-rate to a suitable value for all simulation components. This routine is performed in frequent cycles, where the interval is a parameter established according to the simulation model, by the user/developer.

3.1. TAMING DELAYS

The reasoning behind the approach is to substitute $\boldsymbol{K}$ by $\boldsymbol{F_{()}}$ in the relation $\boldsymbol{T_s = K.T_r}$. The parameters of $\boldsymbol{F_0}$ use the concept of tardiness as a measure for delays. To calculate tardiness, we use a non-dimensional value which indicates how much the execution duration is greater than the programmed, considering the physical-time. The reference to calculate the delay is the instant $\boldsymbol{T_0}$, which is the time when the action was scheduled. The used tardiness formula is:

$$Tardines = \frac{T_0 - T_{executed}}{T_0 - T_{programed}} - 1$$

Given the tardiness concept and measure, the $\boldsymbol{F_0}$ used to substitute $\boldsymbol{K}$ is:

$$T_S = F(b, mt, nt, h).T_r$$

In the proposed relation, the variables have their values updated before running the algorithm to calculate the new ***F(b, mt, nt, h)***, whose value will be used to update the simulation-time in the next cycle. In the same period, the main clock will receive data to update the variables values for the next cycle. The variables are as follows:

- ***b*** - current bias of the tardiness;
- ***mt*** - maximum value of tardiness received during the last update cycle;
- ***nt*** - the number of tardiness measures that were above the established limit; and
- ***h*** - the number of cycles the DVES has been performing without exceeds the tardiness

limit.

The simulation-rate control algorithm implements a heuristic that has a non-linear behaviour. This heuristic was developed and refined during the research, and it responds differently according to the input. It is not a function, since it has some points of discontinuity that can be triggered by high values of tardiness. Additionally, the architecture is modularized in a way that the substitution of this algorithm can be done without affecting the other components.

Considering the control algorithm, it is likely that, for specific cases, a better solution could be achieved by substituting it by a neural network or even a support vector machine (SVM) [47]. However, these tools need to be trained for a particular DEVS in its production environment. Thus, they are inappropriate for the general case.

The architecture considers that each simulator will be wrapped by a simulation agent. These simulation agents are responsible for interfacing with the simulation clock and for collecting the tardiness information. The tardiness information is generated inside this agent, through a parallel thread that runs a cyclic behaviour and collects its tardiness at each execution.

These agents are active objects in the architecture, or, in other words, they must be implemented as a thread in order not to interfere with the clock and the legacy systems. Additionally, these simulation agents can also be extended as new simulators, as presented in our example of instantiation.

3.2. THE ARCHITECTURE INSTANTIATION

The proposed architecture was instantiated as a platform to develop DVES in which the joint simulations use the DES formalism [49, 48] and was coded using Java technology, libraries and platforms to develop multi-agent systems and multi-agent based simulations (MABS).

In order to choose the underlining platforms, we conducted our own review, supported by other works presented in [9, 31], and the final option fell on MASON [22]. MASON is a mature and well-known platform to develop MABS. It allows the execution of DES and provides a comprehensive set of libraries. However, it does not have a native support for being used in DVES.

Since MASON does not support scaled-time simulations, a new behaviour was included in the simulation agents from the architecture to control the advancing of the simulation-time in MASON. In a nutshell, the simulation agents advance the MASON simulation state to the current simulation-time at each cycle, following the timestamp of the events in the discrete-event queue of each MASON instance.

Since a communication mechanism and also a shared representation of the environment is needed to conduct the wargames, we have implemented the following solutions, in addition to the architecture:

- A communication component, inspired by the LINDA [10] architecture: agents can send information through this component to a list of destinations, and also recover messages sent to them by other agents.
- A spatial shared state component: this component receives virtual space updates from agents' position and attitude, and answers queries from agents about simulation entities in their visible space.

Both components are connected to a database with geographic information support (i.e PostGIS/PostgreSQL) and all data is stored for posterior analysis or playing-back. To use these features, a shared data model between the simulators is needed, both semantic and syntactic. The current implementation does not provide a template for shared data models; it is up to developers to choose a suitable model.

The final DEVS is composed of simulators that interact with each other and advance

by the same simulation rate over the simulation-time. The shared spatial representation and the communication components provide a point of extension where it is possible to monitor all the simulation and to interfere with the joint MASON simulators through message sending.

The platform implementation was done using the following frameworks and libraries (Fig. 2): JADE [8], MASON, Jena [18], JTS [19], and PostGIS. All communications between components that run on distinct processes or simulators were implemented using remote method invocation (RMI), a Java implementation for the CORBA specifications [27].

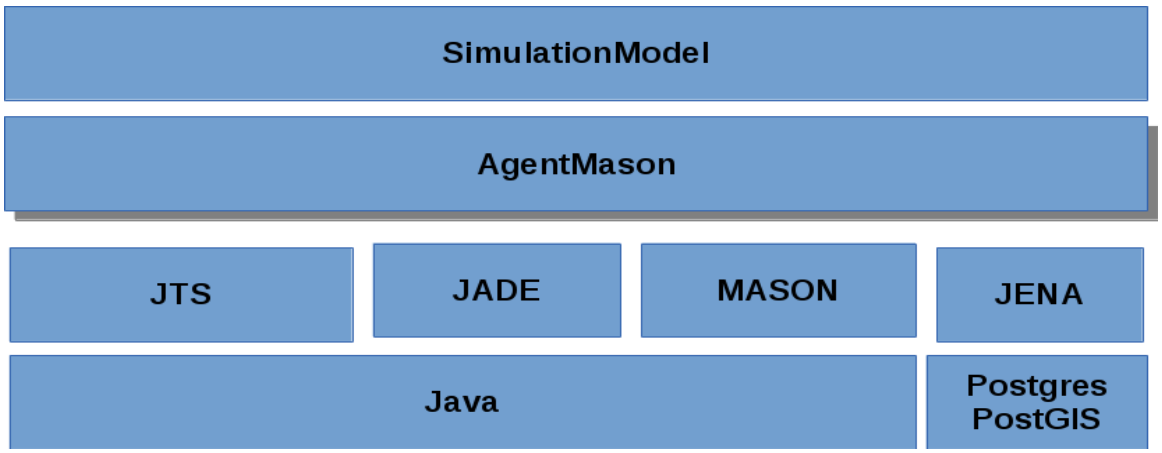

Figure 2: Implementation

4. EXAMPLE OF USE: SUBMARINE HUNTING

A scenario was implemented to test the solution, and also to exemplify its use. This scenario consists of a number of escort ships that hunt a submarine, which is placed in a patrol zone and is engaging merchant vessels. The general behaviour is simple:

- Submarines try to sink all ships detected in the torpedo range, and evade (hide) after an attack;
- Escort Ships search for the submarine, sharing its position to the other escort ships when it is detected. If they have a submarine target in torpedo range, they fire on it; and

Merchant Ships cross the area and are eventually sunk. Each ship/submarine is itself a MASON simulator able to handle its own knowledge base, updated by sensors and external inputs. These simulators implement the actions to be performed by ships/submarines as discrete-events actions extending the MASON models. These actions are triggered at the appropriated simulation-time by the simulation agents from the architecture.

The environment used to execute this DVES was formed by two computers, one of them running two instances of MABS in virtual machines and the other with only one instance. The number of MASON simulations joined on the environment varied from 1 to 120 during the experiment, and the maximum number of agents in each joined simulator was 40. Each MASON simulation represents one military unit, and each MABS instance represents one joint wargame, i.e simulator.

During the experiment, sets of MASON simulations were joined and removed at execution time, to verify the heuristic capability of adapting the simulation-rate under harsh changes.

Figure 3 presents a snapshot from the shared environment, where each point represents a MASON simulation, that is, a ship or a submarine. The simulation takes place in the west entry of the English Channel, and the picture was captured using the software QGIS 2.6 connected to a geographic database.

Finally, the graph depicted in Figure 4 shows the variation of the simulation-rate over 25 minutes of simulation, measured in wallclock-time. The wallclock time is measured in physical time, stating with the simulation [16]. During the experiment, the tardiness was controlled; however, about 8% of the tardiness measures were greater then the limit of 0.2 established for this test. The graph represents one test that was initiated with one node with 10 agents, and after 5 and 10 minutes, other nodes with were joined to the simulation. These nodes were removed from the simulation in minutes 15 and 20. The consequence of these movements can be observed in the graphs.

5. GENERAL DISCUSSION

The problem of interoperating wargames depends on a solution to interoperating its constructive simulations; which simulators have some specificities that prevent the use of standard solutions as HLA or DIS. The presented architecture has been developed to address these issues, and provides a solution to simulation-time distribution under a flexible simulation-rate. The architecture was conceived aiming at a flexible solution both to implement DVES and to integrate legacy simulators, providing components that act as interfaces.

Specific features for supporting a global representation and the information exchange between the joint simulators were not incorporated into the architecture, due to its focus on the handling-time issue. We have implemented basic support for testing purposes in the case study. However, in our view, the federation object model (FOM), which is part of the HLA standard, could be used as a base for a more comprehensive work. The FOM is well experimented and decoupled from the HLA time management.

The architecture does not aim to eliminate tardiness: it is designed to ensure that the simulation runs below a maximum tardiness level. The key is to define a level of tardiness that will not compromise the users' perception of reality. We didn't find, during our review, works with a focus on the interoperability of wargames simulators with similar characteristics as those discussed throughout this work.

The presented implementation, using a Java-based multi-agent system on distributed

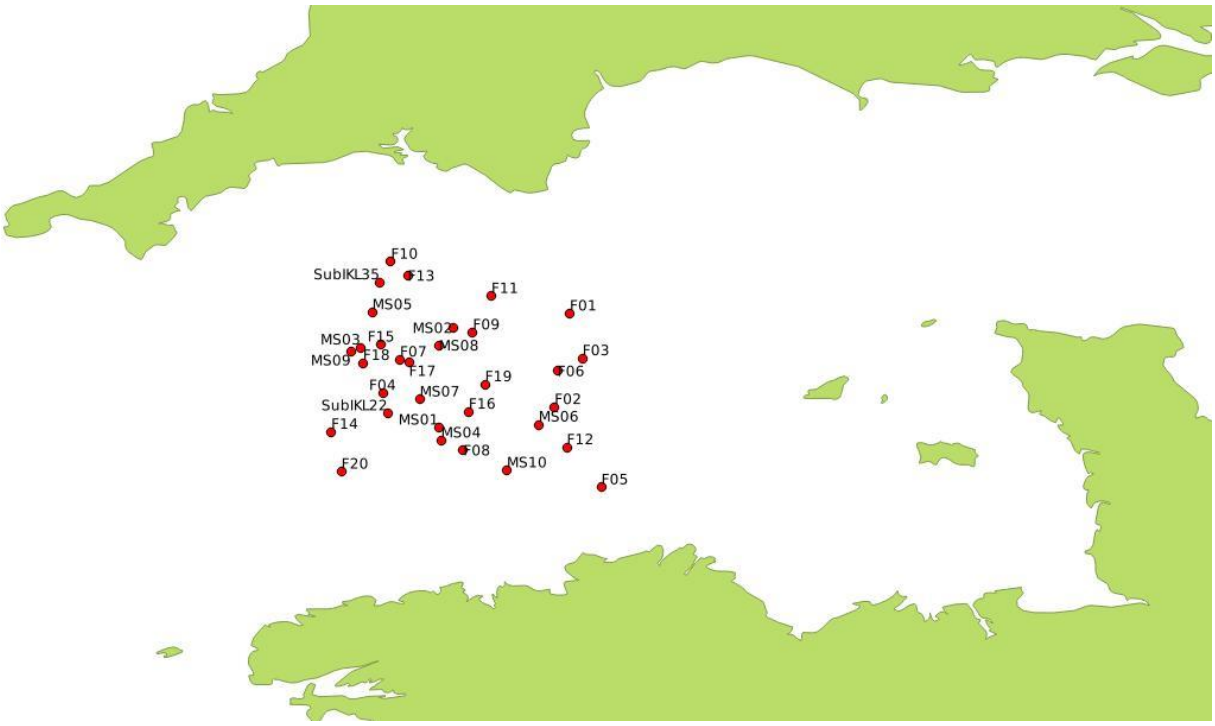

Figure 3: Sub Hunt

environment, is a complex scenario for taming tardiness. Besides the usual problems, Java-based multi-agent system are usually multi-threaded and have a number of issues relating to time-scheduling that affect performance. The example of use is simple and constrained by the resources available to conduct it. Despite this, this example was sufficient to check all platform features and specifically the tardiness control efficacy.

6. CONCLUSION

This paper presents a solution to a specific industry problem in the military domain, which is interoperability for constructive simulations used in wargames. The issue affects only a small number of simulators, but these systems are used both to conduct planning and training activities at strategic/operational levels. This means that the players are frequently highly-ranked military or civilians involved in defence affairs. A solution to this problem is pressing, since joint wargames are needed both for training and for establishing new procedures under the new paradigms of joint military operations in complex endeavours. Moreover, wargames are one of the most useful tools to exploit and analyze new scenarios.

Interoperability between the supporting simulators is a major step needed to interoperate wargames and to produce a shared virtual environment. The dynamic view of the overall military operation is important for devising interesting outputs, such as: the need for new processes to avoid mutual interference in war operations; a logistical bottleneck in a multi-modal system being tested to face a hypothetical natural catastrophe (e.g floods); or to train the people responsible for governmental agencies (e.g homeland security, police, health structure), to respond in a coordinated way to a hurricane approaching the coast. The cost of not performing the games could be unacceptable in these cases.

The solution presented is a conceptual architecture, which was further detailed and implemented for testing purposes. The results indicate a viable solution, which is not optimal, having space to be improved and refined, or even substituted in the future. However, it is a feasible solution, which was designed to be applied with minimal intervention inside wargames simulators.

As a suggestion for future work, the architecture could be extended to provide a middleware platform for message exchange.

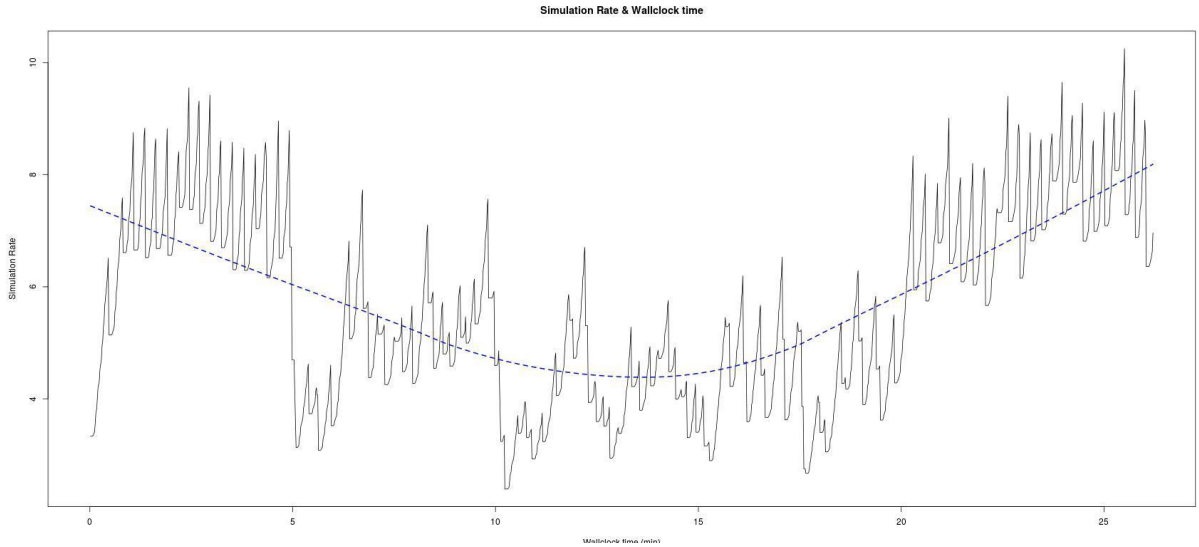

Figure 4: Simulation rate & wallclock time

7. REFERENCES

[1] Ieee 1278.1-2012 - standard for distributed interactive simulation–application protocols, https://standards.ieee.org/findstds/standard/1278.1-2012.html.

[2] Ieee 1516-2010 - standard for modeling and simulation (m&s) high level architecture (hla)– framework and rules, https://standards.ieee.org/findstds/standard/1516-2010.html

[3] Abt, C.: Serious games. University Press of America, Incorporated (1987)

[4] Alberts, D.S.: The Agility Advantage: A Survival Guide for Complex Enterprises and Endeavors. CC (2011)

[5] Alberts, D.S., Garstka, J.J., Stein, F.P.: Network Centric Warfare. CC (1999)

[6] Alberts, D.S., Hayes, R.E.: Power to the Edge: Command and Control in the Information Age. CCRP (2003)

[7] Amouroux, E., Chu, T.Q., Boucher, A., Drogoul, A.: Gama: An environment for implementing and running spatially explicit multi-agent simulations. In: Ghose, A., Governatori, G., Sadananda, R. (eds.) Agent Computing and Multi-Agent Systems, Lecture Notes in Computer Science, vol. 5044, pp. 359–371. Springer Berlin Heidelberg (2009), http://dx.doi.org/10.1007/978-3-642-01639-4_32

[8] Bellifemine, F.L., Caire, G., Greenwood, D.: Developing Multi-Agent Systems with JADE (Wiley Series in Agent Technology). John Wiley & Sons (2007)

[9] Berryman, M.: Review of software platforms for agent based models. Tech. Rep.DSTO-GD-0532, Australian Government, Department of Defence (2008)

[10] Carriero, N., Gelernter, D.: Applications experience with linda. SIGPLAN Not. 23(9), 173–187 (Jan 1988), http://doi.acm.org/10.1145/62116.62132

[11] Cedeno, W., Laplante, P.A.: An overview of real-time operating systems. Journal of the Association for Laboratory Automation (JALA) 12(1), 40–45 (February 2007)

[12] Curry, J., Perla, P.: Peter Perla's The Art of Wargaming A Guide for Professionals and Hobbyists. lulu.com (2012)

[13] Dey, A.: Understanding and using context. Personal and Ubiquitous Computing 5(1), 4–7 (2001)

[14] Dickinson, J.R., Gentry, J.W., Burns, A.C.: A seminal inventory of basic research using business simulation games. Developments in Business Simulation and Experiential Learning 31 (2014)

[15] Drogoul, A., Vanbergue, D., Meurisse, T.: Multi-agent based simulation: Where are the agents? In: Multi-Agent-Based Simulation II, pp. 43–49 (2003)

[16] Fujimoto, R.M.: Parallel and Distributed Simulation Systems. Wiley Series on Parallel and Distributed Computing, John Wiley & Sons Inc. (2000)

[17] Gordon, G.: System Simulation. Prentice Hall PTR, Upper Saddle River, NJ, USA (1977)

[18] JENA: A Semantic Web Framework. http://jena.sourceforge.net/ (2007)

[19] JTS: Java topology suite. http://www.vividsolutions.com/jts/jtshome.htm (2010)

[20] Laplante, P.A.: Real-Time Systems Design and Analysis, 3rd Edition. Wiley-IEEE Press (2004)

[21] Law, A., Kelton, D.: Simulation Modeling and Analysis 2nd. McGraw-Hill, New York, NY (1991)

[22] Luke, S., Cioffi-Revilla, C., Panait, L., Sullivan, K., Balan, G.: Mason: A multiagent simulation environment. Simulation 81(7), 517–527 (Jul 2005), http://dx.doi.org/10.1177/0037549705058073

[23] Moffat, J.: Complexity theory and network centric warfare. Information age transformation series, DoD Command and Control Research Program (2003)

[24] North, M., Collier, N., Ozik, J., Tatara, E., Macal, C., Bragen, M., Sydelko, P.:Complex adaptive systems modeling with repast simphony. Complex Adaptive Systems Modeling 1(1), 3 (2013), http://www.casmodeling.com/content/1/1/3

[25] North, M.J., Macal, C.M.: Managing business complexity : discovering strategic solutions with agent-based modeling and simulation. Oxford University Press (2007)

[26] Oaks, S., Wong, H.: Java Threads. O'Reilly Media, 3th edn. (2004)

[27] OMG: Corba specifications, http://www.omg.org/spec/index.htm

[28] Perla, P.P.: The art of wargaming: a guide for professionals and hobbyists. Naval Institute Press (1990)

[29] Perumalla, K.S.: Parallel and distributed simulation: Traditional techniques and recent advances. In: Proceedings of the 38th Conference on Winter Simulation. pp. 84–95. WSC '06, Winter Simulation Conference (2006), http://dl.acm.org/citation.cfm?id=1218112.1218132

[30] Pidd, M.: Systems Modelling: Theory and Practice. John Wiley & Sons (2004)

[31] Railsback, S.F., Lytinen, S.L., Jackson, S.K.: Agent-based simulation platforms: Review and development recommendations. SIMULATION 82(9), 609–623 (2006), http://sim.sagepub.com/content/82/9/609.abstract

[32] Robledo, L.F., Godoy, E.: Sigen: Simulation for training and management on emergency situations. In: 13th International Command and Control Research and Technology Symposium (ICCRTS): C2 for Complex Endeavors (2007)

[33] Sabin, P.A.G.: Simulating War: Studying Conflict Through Simulation Games. Continuum Publishing Corporation (2012)

[34] Siemssen, M., Kreuzeder, M.: Negotiations and Decision-Making in the European Union - Teaching and Learning through Role-Play Simulation Games. VDM Verlag Dr. Mueller e.K. (2007)

[35] Smith, E.A.: Effects-Based Operations. CC (2003)

[36] Strassburger, S., Schulze, T., Fujimoto, R.: Future trends in distributed simulation and distributed virtual environments: results of a peer study. In: Proceedings of the 40th Conference on Winter Simulation. pp. 777–785. Winter Simulation Conference (2008)

[37] SWARM: Swarm development group. http://www.swarm.org/index.php/Swarm main pag(2010)

[38] Taranti, P., Choren, R., Lucena, C.J.P.: Architecture to tame simulation time tardiness in ads. In: Proceedings of the 2010 Spring Simulation Multiconference. vol. Agent-Directed Simulation (ADS'11), pp. 37–44. Society for Modeling and Simulation International, Boston (April 2011)

[39] Taranti, P.G.: An architecture to tame time tardiness in multiagent based simulations. Ph.D. thesis, Pontificia Universidade Catolica do Rio de Janeiro (PUC-Rio) (2013)

[40] Taranti, P.G., Breitman, K.K., Lucena, C.J.P., Choren, R.: An approach to reduce the gap between conceptual and execution models in agent-directed simulations. In: Proceedings of the 2010 Spring Simulation Multiconference. pp. 4:1–4:8. SpringSim'10, ACM, New York, NY, USA (2010)

[41] Taranti, P.G., Choren, R., Bommel, P., de Lucena, C.J.P.: Virtual environment simulations (ves) with mabs: an approach to tame tardiness in java-based simulation systems. Monografias em Cincia da Computao 17, Pontifcia Universidade Catlica do Rio da Janeiro (Puc-Rio) (December, 2012 2012), iSSN: 0103-9741

[42] Taranti, P.G., Choren, R., Lucena, C.J.: A quantitative study about the hidden risk of using time-scheduler mechanisms to control execution flow in jade-based mas.In: Proceedings of I Workshop on Autonomous Software Systems (AutoSoft) at Interoperating wargame simulations 15 First Brazilian Conference on Software: Theory and Practice (CBSoft). pp. 61–70. SBC (2010)

[43] Taranti, P.G., Lucena, C.J.P., Choren, R.: Mabs com turnos centrados em agentes e implementadas em java: controlando atrasos em relacao ao tempo de simulacao. In: Anais do II Workshop sobre Sistemas de Software Autonomos (AutoSoft 2011) - CBSoft 2011. pp. 40–49. SBS, Sao Paulo (2011)

[44] Taranti, P.G., de Lucena, C.J.P., Choren, R.: An architecture to tame simulation time tardiness in ads. In: Proceedings of the 2011 Workshop on Agent-Directed Simulation at the 2011 Spring Simulation Multiconference. pp. 29–36. ADS '11, Society for Computer Simulation International, San Diego, CA, USA (2011), http://dl.acm.org/citation.cfm?id=2048355.2048359

[45] Taranti, P.G., Lucena, C.J.P.d., Choren, R.: A quantitative study about tardiness in java-based multi-agent systems. In: Agent Systems, their Environment and Applications (WESAAC), 2011 Workshop and School of. pp. 37 –44. Curitiba, Parana Brazil (april 2011), print ISBN: 978-1-4673-0735-2

[46] Tisue, S., Wilensky, U.: Netlogo: Design and implementation of a multi-agent modeling environment. In: Proceedings of agent. vol. 2004, pp. 7–9 (2004)

[47] Witten, I.H., Frank, E.: Data Mining: Practical machine learning tools and techniques. Morgan Kaufmann, 3rd edn. (2011)

[48] Zeigler, B.P.: Toward a formal theory of modeling and simulation: Structure preserving morphisms. Journal of the ACM (JACM) 19(4), 742–764 (1972)

[49] Zeigler, B.P., Praehofer, H., Kim, T.G.: Theory of modeling and design – integrating discrete event and continous complex dynamic systems (2. ed.). Academic Press, 2nd edn. (1999)

AUTHOR BIOGRAPHIES

Pier-Giovanni Taranti is an Officer and researcher in Software Engineering at Centro de Analises de Sistemas Navais (Brazilian Navy). His is interest in multi-agent simulations and multi-agent architecture and their application in the industry.

Ricardo Choren is an Associated Professor at Instituto Militar de Engenharia (Brazil). He has experience on Computer Science, and interested in Software Engineering, Software Agents, and Software Design.

Carlos Lucena is Full-Professor of Computer Science and Software Engineering at Pontificia Universidade Catolica -Rio de Janeiro (Brazil). Hi is interested in the following areas: Software Engineering and Formal Methods, Software Engineering Environments and Formal Specification Methods, and Applications of Information Technology.

Michael Luck is Professor of Computer Science and Dean of the Faculty of Natural and Mathematical Sciences at King's College London (UK). He was Head of the Department of Informatics from 2011 to 2013, where he also works in the Agents and Intelligent Systems group, undertaking research into agent technologies and intelligent systems. He is Scientific Advisor to the Board for Aerogility.

Peter McBurney is Professor of Computer Science and Head of the Department of Informatics at King's College London (UK). His undertakes research in agent communications languages and protocols, and in agent-based modelling (ABM). He is particularly interested in applications of ABM to complex adaptive systems domains comprising intelligent interacting entities, such as financial and economic markets, and domains of cyberwar and cyber conflict.