

Design of an Online Decision-Making Support System for Joint Training Simulation

Jiung-yao Huang

Dept. of Computer Science
and Information Engineering
National Taipei University
San Shia, Taipei, 237 Taiwan
jyhuang@mail.ntpu.edu.tw

This paper describes the design of an online decision-making support system for a distributed operational training environment. The goal of this research is to allow the user to interactively actuate an analytical simulation system to perform online decision assessment while he is playing in the operational training simulation. The interoperability between these two heterogeneous simulation systems is an important issue for this research. This paper presents an approach using High Level Architecture (HLA) standards to achieve this interoperation. This paper explores the rationale of interoperating an event-driven analytical system and a time-stepped operational training system. From this investigation, this paper identifies the interoperation interval, called the *assessment era*, between these two heterogeneous systems that can execute asynchronously while interacting cooperatively. Furthermore, the process of computing such an assessment era is fully discussed with a demonstration scenario. Our experimental exercise shows that the presented method is a more feasible and efficient way to improve the training effect than a traditional joint training environment.

Keywords: Online decision-making system, HLA, federate gateway, federate agent

1. Introduction

The legacy analytical wargame system described in this paper is an event-driven simulation. After the initial conditions are given, the system is executed with pre-scripted events in multiple rounds until the result is converged to a stable value. To achieve this goal, we use a faster-than-real-time time advancing strategy to perform the simulation. Furthermore, the initial conditions for the analysis are given based upon hypothetical assumptions independent of the opposing force situation and the dynamics of the battlespace. For real-world warfare, the scenario may change rapidly along with the enemy's situations. Hence, for the analytical wargame system to perform a practical assessment, it must be able to clone the dynamic scenario before performing analysis. In other words, when the analytical system is capable of interoperating with the operational training environment, it can perform course of action (COA) analysis of the enemy's scenario [1-4]. This paper presents a method of using HLA standards to integrate

an analytical system with a joint training simulation environment so that the analytical system can perform online decision-making support for a user.

Joint training simulation is the application of interoperating multiple time-stepped operational training systems to form a consistent synthetic battlespace [5]. This paradigm emerges from the distributed simulation community, which designs different interoperating protocols to interconnect geographically distributed nodes of simulation [6]. To achieve this goal, the simulation community places certain restrictions on the realization of individual simulations with the aim of creating a consistent synthetic simulation environment over a network. This approach views the entire synthetic virtual world as a black box that is composed of thousands of geographically distributed nodes of simulators. Simulation nodes within this black box are interconnected by an interoperating protocol to achieve the goal of concurrent processing. Well-known protocols for this paradigm include Distributed Interactive Simulation (DIS) [7], Aggregate Level Simulation Protocol (ALSP) [8], and the High Level Architecture (HLA) [9]. The HLA is an ameliorative technique that combines the

operational concepts of DIS and ALSP, as well as other distributed simulation issues, into a standardized infrastructure. The technical innovation of HLA allows a radical upward scaling of the number of distributed simulation nodes able to participate in a single scenario.

The HLA is a project initiated by the U.S. Department of Defense to support interoperability among geographically distributed simulators. It later became an international standard, IEEE 1516, in 2001 [9]. The HLA defines an infrastructure to support the reusability and interoperability among heterogeneous simulations. To achieve this goal, the HLA defines a *time management* service for a simulation system to coordinate its execution pace with others. The time management service defines the synchronization mechanism to ensure causal order consistency of the event sequence among distributed simulation nodes. In HLA terminology, each distributed node of a simulation is called a *federate*. Furthermore, a federation is a simulation environment that consists of a set of federates. Depending upon the simulating model, a federate can use the time-stepped or event-driven time advancing technique. For example, an analytical federate is a non-interactive simulation that uses the event-driven approach, whereas the operational training simulation, which requires a user's input to perform the simulation, adopts the time-stepped mechanism.

Traditionally, the analytical simulation and the operational simulation are disjointed simulation domains. As demanded by the complexity of contemporary warfare, the heterogeneous joint training environment becomes increasingly important to military readiness [10]. While the design of the heterogeneous joint simulation environment is a complex technical challenge, the HLA time management service defines a method to synchronize different types of federates within a federation execution or a federation communication.

This paper presents an approach to transform a legacy analytical system into an online decision-making support system for the user of a joint operational training simulation. Although the HLA specification enables time-stepped and event-driven federates to execute in a single federation, questions of why and how to integrate an analytical federation with a time-stepped federation cause diverse disputes. In the following sections, we first present the classification of interoperability according to the HLA. Next, we address the technical issue of joining the analytical system with a time-stepped operational federate. Finally, we present the performance study of the described approach.

2. Methodology of Joint Simulation of Heterogeneous Systems

Similar to the Internet Protocol layers, interoperability among distributed simulations can also be classified into four layers: application interoperability, model interoperability, service interoperability, and communication interoperability [11]; see Figure 1.

In this paper, the heterogeneous joint simulation refers to the joint training simulation environment composed of different types of simulation systems, including different levels of resolutions or different time-advancing mechanisms such as time-stepped or event-driven. We can categorize the heterogeneous joint simulations according to the layers of interoperability [11]. The application layer of interoperability is one of the goals that HLA specification attempts to achieve. Simulations of different types functionally interoperate through interactions with their runtime infrastructures (RTIs), assuming they are using the same RTI. Their operational discrepancies may include the time-advancing mechanisms employed or design purposes with distinct simulating models. The model level of heterogeneous interoperability refers to distinct object models used by interoperated distinct federates. This level of interoperation may include federates using different federation object models (FOMs) or the same FOM but with different resolutions of object representation. For example, the tank simulator and the battalion wargame system have entities with different resolutions, and their interoperability can be classified as the model level of interoperation. For the service level of interoperability, simulations with different communication service implementations share information with each other. This category of interoperation may include federations using RTIs from different vendors or simulation environments using different communication protocols. Finally, the communication level of interoperability attempts to achieve the service-level interoperability between components of the same RTI or those

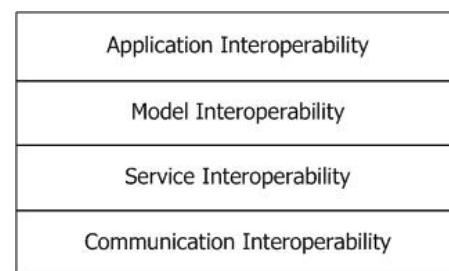


Figure 1. Levels of interoperability

from different RTI manufacturers. For example, different manufacturers building their RTIs with distinct communication protocols, such as TCP and SCRAMNet, cannot interoperate directly.

This paper focuses on the issue of heterogeneous interoperability of time-stepped and event-driven simulations. Restricted by the experimental environment, our study also includes upgrading non-HLA simulation systems so that they can interoperate by using the HLA specification. In the following subsections, the approaches to achieve interoperation among multiple federations are given. Next, we describe the methods to connect an HLA simulation system with a non-HLA simulation system, such as a DIS simulator.

2.1 Interoperation of Multiple Federations

Heterogeneous integration of distributed simulations links distinct federations into a federation community [12]. This linking can be classified into four types of architectures: 1) homogeneous FOM & homogeneous RTI, 2) heterogeneous FOM & homogeneous RTI, 3) homogeneous FOM & heterogeneous RTI, and 4) heterogeneous FOM & heterogeneous RTI [11]. The original design principle of the HLA specification is the interoperation of multiple federates that use the same FOM and communicate through a single RTI implementation. That is, the specification of the HLA only defines the first type of architecture: homogeneous FOM & homogeneous RTI. To solve the needs of building the federation community, several methods have been proposed over the years and can be divided into four types [11, 12]: federate gateway, federate proxy, RTI broker, and RTI interoperability protocol. Their differences depend upon whether they are designed to resolve the interoperability at the application level, model level, service level, or communication level.

2.1.1 Federate Gateway

A *federate gateway* is a translation device that enables interconnection among two or more discrete federation executions in a federation community [11]. Two types of gateways have been discussed: the federate gateway, which is used to interoperate heterogeneous federations, and the HLA gateway, which translates messages between federation and legacy simulation systems [13]. The security guard federate [14] is an example of a federate gateway. The main goal of this federate is to filter or guard secure information among integrated federations to produce a secure simulation environment. According to Myjak and colleagues [11], the federate gateway

does not directly support HLA interface. Although the security guard federate directly interacts with two RTIs—which does not fit the original definition of a federate gateway—Myjak and Sharp [12] list it in the federate gateway category. The same issue also applies to the HLA gateway [13]; in order to perform protocol translation, the HLA gateway directly interfaces with an RTI at one end and a DIS protocol at the other. The cluster gateway by Chen and colleagues [15] is another example of an HLA gateway. The cluster gateway acts as a special federate inside a cluster of federates to provide inter-cluster interoperation with other clusters and to develop a cluster-based HLA architecture.

In essence, the main idea of the federate gateway is to provide a connection point between two or more heterogeneous federations or between a federation and other legacy systems. It provides an “out-of-band” communication path between different federation executions. Based upon the above gateway examples, the federate gateway belongs to both the application level and model level of interoperability. The federation community connected by the federate gateway belongs to either the architecture of heterogeneous FOM & homogeneous RTI, as the security guard federate does [14], or homogeneous FOM & heterogeneous RTI, as the HLA gateway does [13].

2.1.2 Federate Proxy

Unlike the federate gateway, the *federate proxy* is a federate that joins more than one federation execution. This approach allows multi-federation interoperation by using only the services defined in the Federate Interface Specification Application Programmer’s Interface (API). Hence, this type of federate was previously called a “federate bridge” or “bridging” of federations [16]. Such a federate acts as a mediator; it helps a local federate update its status to a remote federate on a different federation and vice versa. Through the help of the federate proxy, federates of different federations with different FOMs can transparently communicate [12]. The federate proxy can also “glue” federations that use distinct RTIs into a federation community [17]. Hence, by definition, the federate proxy also belongs to both the application level and model level of interoperability. Also like the federate gateway, the architecture of the federation community connected by the federate proxy is either heterogeneous FOM & homogeneous RTI [12] or homogeneous FOM & heterogeneous RTI [17].

2.1.3 RTI Broker

The RTI broker is similar to a federate proxy that lies between two federations. However, unlike the federate proxy, the RTI broker acts as a mediator that interchanges messages between two RTIs, possibly manufactured by different vendors, at the service level. That is, it directly interacts with the RTI to provide interoperation. Hence, it not only interoperates federations but also synchronizes RTIs designed with different techniques. The main drawback of this approach is that it requires a yet-to-be-defined RTI-to-RTI API. Hence, the RTI broker functions at the service level of interoperability and constructs a federation community architecture of homogeneous FOM & heterogeneous RTI or heterogeneous FOM & heterogeneous RTI.

2.1.4 RTI Interoperability Protocol

A more efficient method to provide interoperability among distinct RTIs, including those with differences of programming languages, algorithms, and design, is the *RTI interoperability protocol*. It uses an “on the wire” approach that provides service-level interactions between components within an RTI or between similar components by different RTI manufacturers. Compared to the RTI broker, which interchanges messages through APIs of different RTIs, this approach allows RTIs to directly communicate their internal states as well as federate data. The RTI interoperability protocol provides a communication level of interoperability within a federation community. In other words, its community architecture can support homogeneous FOM & heterogeneous RTI or heterogeneous FOM & heterogeneous RTI.

2.2 Federating Non-HLA Simulation Systems

The HLA specification only defines the protocol to interoperate simulations built with the HLA standards in mind. It says nothing about how to communicate with legacy simulation systems. Aside from reengineering a legacy system to an HLA-compatible system, Wood and Petty [13] suggest a gateway approach to make a legacy system HLA interoperable. As discussed in the previous section, a federate gateway is an extra mechanism to convey messages between HLA and non-HLA simulation systems. The HLA Gateway, designed by the Florida Institute for Simulation and Training (IST), is well known in interoperating DIS simulators with HLA simulation systems [13]. With this approach, no modification of the non-HLA simulators is required,

and the HLA Gateway is responsible for converting messages between the two different protocols. When the HLA Gateway receives a PDU (protocol data unit) from a DIS simulator, it translates this message into the corresponding RTI service for the HLA federates. Conversely, when the HLA Gateway receives a callback from other federates, it then packs a PDU according to a *real-time platform reference-federation object model* (RPR-FOM) and forwards this PDU to the DIS simulators.

In addition to externally using an HLA Gateway to federate non-HLA simulators, Paterson and colleagues [18] suggest a middleware approach, which only modifies the communication module of the legacy system. Compared to the approach of reengineering a legacy system into an HLA system and the gateway approach, the middleware approach is a balanced alternative between the previous two methods. Essentially, the middleware plays the role of an exclusive “internal gateway” for the non-HLA system when it interoperates with the federation execution. Compared to the other options, middleware costs less to implement than the reengineering approach and performs more efficiently than the federate gateway approach.

3. Using an Analytical System as an Online Decision-Making Support System

3.1 Rationale

As the HLA is more widely adopted by different countries to interoperate different systems, such as the one described in this paper, the Taiwanese military is also actively acquiring and developing various HLA-based training systems. However, questions of why and how to interoperate training systems with the HLA standard are consistently raised as it becomes the policy of Taiwan’s future training systems. To answer these questions, various preliminary experiments have been conducted. This paper describes such an effort. The objective of this research is to answer the questions of why and how to interconnect and interoperate the time-stepped operational training system with the event-driven analytical wargaming system.

The legacy event-driven analytical wargaming system is designed to repeatedly simulate a pre-scripted sequence of events. Once the analytical simulation is executed, it will not stop until the specified number of iterations or a predefined condition is reached. Moreover, the pre-scripted sequence of events is based upon hypothetical assumptions, independent of the dynamics of the battlespace. For a time-stepped theater-level

operational training simulation, the scenario changes rapidly along with the ongoing campaign. Since analytical wargaming software and the operational training simulations use different modeling techniques and time advancing strategies, this integration raises the following questions:

- *Why:* Are there any specific benefits to interoperating the event-driven analytical system with the time-stepped operational simulation?
- *When:* Since the event-driven analytical system and the time-stepped operational training simulation were designed with different time advancing strategies for distinct purposes, should they be interoperated at all times during the federation execution? If so, the event-driven system would have to be degraded into the time-stepped simulation in order to coordinate their time advancing techniques. Can the analytical wargaming system be interactively actuated by the user of the operational training simulation? If so, when?
- *How:* Can these two diverse simulation systems be interconnected? What is the architecture to construct such a heterogeneous environment?

One of the purposes of interoperating an analytical wargaming software with an operational training system is to allow the former to perform course of action (COA) analysis for the latter. The study of COA analysis to assist decision makers in ongoing campaigns is the trend of modern wargaming research [1-4]. Gilmour et al. [3] further explore technologies to achieve real-time COA analysis to assist the decision maker at the decision-making point. Their research focuses on the high performance computing (HPC) environment. Our research, however, attempts to explore the technique of real-time COA analysis in a distributed training environment. To be more specific, our research explores the technique that integrates an analytical system into a distributed theater-level operational training environment so that the analytical system can interactively perform COA analysis for the user as quickly as possible.

In order to integrate the analytical wargaming software into the distributed operational training simulation to perform online COA analysis, the analytical software must be able to clone the dynamic scenario immediately before it is executed. In other words, it must clone the dynamic scenario at the decision-making point for it to interactively perform the online COA simulation analysis. Before integrating the analytical system with the operational training system, however, the three integrating questions, i.e., why, when, and how, need to be explored.

3.2 Why: The Objective of Interoperability

The event-driven analytical system and the time-stepped operational training simulation are two distinct types of simulation systems. Their differences range from architecture and time advancing mechanisms to application domains. Hence, the interoperability issues between these two dissimilar simulating systems include not only technical problems but also the objective of interoperability.

From the perspective of authenticity, analytical wargaming systems often derive more accurate damage assessment results than operational training simulations. In order to meet the deadline of each time step, a time-stepped operational training simulation often uses fast but simplified assessment models to analyze the simulated combat. On the other hand, since an event-driven analytical wargaming system attempts to compute simulation results as quickly as possible, its assessment model is often more precise and its outcome is more credible. Furthermore, since a time-stepped training simulation is a tactical operational training system, its scenario is dynamic with an adversarial environment. On the other hand, the scenario for the analytical system is often restricted to a specific topic with a pre-scripted sequence of events. Due to these innate differences, we categorize the reasons for the interoperation between the analytical system and operational training simulation as follows:

- 1) The analytical system boosts the simulation accuracy of the operational training simulation. Since an operational training simulation aims to exercise the tactical skills of the commander, the accuracy of the assessment model is not an important subject. On the contrary, the course of the simulation is the important issue in an operational training environment. However, for certain training scenarios, the precision of the damage assessment model may profoundly affect the results of training. Hence, we can use the outcome of the analytical system to boost the engagement accuracy of the operational training simulation. For example, operational training simulations often use tabular data to estimate the results of engagement while analytical systems use a more accurate damage assessment model, such as the Monte Carlo method, for the engagement.
- 2) The analytical system acts as an online decision-making support system for the operational training simulation. During an operational training simulation, the commander (user) often faces the dilemma of making a "right" decision in an adversarial environment. Based upon the time advancing mechanism of the operational

training simulation, the user often has to decide how to react in a hostile situation in a short period of time with limited information. Hence, within this short time period, we can use an analytical wargaming system to analyze the possible outcome of the user's intention. In other words, the analytical system plays the role of online decision-making support system at that critical decision-making point.

We now focus on the second objective of interoperability over the distributed simulation environment. That is, we attempt to study the technique and mechanism to interoperate the analytical system with the theater-level operational training simulation, so that an event-driven analytical wargaming system can be the online decision-making support system of an operational training environment.

3.3 When: A Model of Interaction

After concluding that the purpose of interoperability is to use an analytical wargaming system as an online decision-making support system, the next question is when the analytical system should be executed during the operational simulation. In addition, how much time can the analytical system have to interactively analyze the user's intention? If the analytical system takes too much time to perform its analysis, the result may be too late to be useful to the user. Hence, from the standpoint of an operational simulation, the time to execute the analytical system, say t_1 , and to receive its analysis result, say t_2 , are two crucial factors for the analytical system as an effective online decision-making support system. That is, two important interactions between the analytical system and the operational training simulation take place at time t_1 , when the user of the operational simulation actuates the analytical system, and time t_2 , when the analytical system returns the analysis result. Furthermore, the interval $[t_1, t_2]$ becomes the critical moment for the analytical system to clone the dynamic scenario, receive the execution command, perform the analysis, and return the analysis result. We call the interval $[t_1, t_2]$ the *assessment era*. If the interval $[t_1, t_2]$ is too small, the analytical system will not have enough time to assess the user's intention and return the analysis result. Hence, the minimum value of the assessment era becomes the decisive factor in determining whether an event-driven analytical system can play the role of an online decision-making support system.

To find the answer, the timing requirement of the analytical system must be studied first. Define T_T as the time required for an operational simulation system to send an execution command to an analytical

system to perform an assessment and T_R as the time for an analytical system to return an analysis result to an operational simulation. Furthermore, let T_A be the average simulation time for an analytical system to perform an assessment per iteration. Hence, the *minimum value of the assessment era* T_{\min} is

$$T_{\min} = T_T + T_R + n * T_A \quad (1)$$

where n is the minimum number of iterations for an analytical system to deduce an acceptable simulation result. Note that times T_T and T_R are dependent on network latency and operating system overhead.

Furthermore, the time a hostile event occurs in the operational simulation will determine if there is enough time for an analytical system to perform analysis. Let t_s be the time of a hostile event e when it is initially detected and let t_d be the latest time at which the user must react to that event e . Hence, the interval $[t_s, t_d]$ is the maximum time that the operational simulation allows the analytical system to perform the online decision-making support. We call the interval $[t_s, t_d]$ the *event period*. Note that time t_d is the function of event e , $t_d = f(e)$, which depends upon the application domain. If the event period is larger than the minimum assessment era, i.e., $|t_d - t_s| > T_{\min}$, then the user can use the analytical system to perform COA analysis. Let t_c , where $t_c \in [t_s, t_d]$, be the time at which the user sends out the execution command to the analytical system, and let the interval $[t_c, t_d]$ be the available time for the analytical system to analyze the user's intention. Note that the available time must also be greater than the minimum assessment era, i.e., $|t_d - t_c| > T_{\min}$, for the analytical system to perform an effective assessment. In other words, the operational training simulation sends an execution command to the analytical system at time t_c and the analytical system must return the assessment before the deadline, t_d .

Note that if the event period is much larger than the minimum assessment era, i.e., $|t_d - t_s| \gg T_{\min}$, the system may allow the user to perform multiple COA analyses, similar to the approach by Gilmour et al. [3], by dividing this interval into several subintervals with each subinterval equal to T_{\min} . The user can interactively actuate the analytical system within any of these subintervals and demand that the assessment be completed before that specific subinterval time expires.

3.4 How: A Joint Simulation Using HLA

The most important architecture issue in integrating an event-driven analytical system into a time-stepped operational training simulation environment is to

synchronize their distinct time management policies. This is a key problem due to the differences in the time advancing mechanisms employed by the event-driven analytical system and the time-stepped operational training simulation; they must execute asynchronously to prevent the analytical one from being scaled down as a time-stepped simulation.

To address this issue we need to further explore the essence of the two systems. For time advancing techniques, the time-stepped operational training simulation uses a scaled time-step of the simulation; the event-driven analytical system is driven by the next executable event. Furthermore, the execution of the operational simulation is manipulated by the user, while the analytical system is dominated by a statistic model with a pre-scripted sequence of events. Hence, in order to integrate these two systems into a single synthetic battlespace without obstructing their innate features, the execution of the analytical system must be dominated by the operational system. According to HLA specifications, the *time management* service of an operational system must be set as a time-regulating federate, while that of an analytical system must be set as a time-constrained federate. Hence, during a simulation, the operational simulation is executed at its own pace while the analytical system is actuated by the event of the operational simulation. When an event occurs and the user decides to execute the analytical system, the user can trigger the operational simulation to send an interaction event with time stamp t_d to the analytical system. Upon receiving this interaction event, the analytical system uses equation (1) to compute the number of iterations that are available for it to perform the analysis.

Furthermore, the analytical system must return the assessment to the operational system before t_d and then issue a *NextEventRequest()* function call. Note that the *NextEventRequest()* function call is used as a mechanism for the analytical system to inform the operational simulation about the completion of analysis.

The interoperability level of this type of integrated simulation belongs to the application level or the model level. According to Section 2, either the federate gateway or federate proxy is a feasible approach to integrate them into a single federation. Moreover, since the legacy analytical system was often designed as a stand-alone simulating system, when the analytical system is joined with the HLA federates, an agent federate is often designed for it. Since the operational simulation for our study is a proprietary theater-level operational training environment, a federate gateway was designed as illustrated in Figure 2.

This federate gateway is a type of HLA gateway and is used to translate messages between proprietary protocols and RTI service calls. On one end, this federate gateway accesses data from the SQL server and synchronizes with other components of the operational training environment through the SQL server. On the other end, the federate gateway uses RTI service calls to interoperate with the analytical system. Before the federate gateway invokes an interaction service call to actuate the analytical system, the analytical system *passively* receives the up-to-date scenario of the operational training simulation. In this way, when the analytical system is executed, the overhead for the analytical system to clone the dynamic scenario of the operational simulation is minimized.

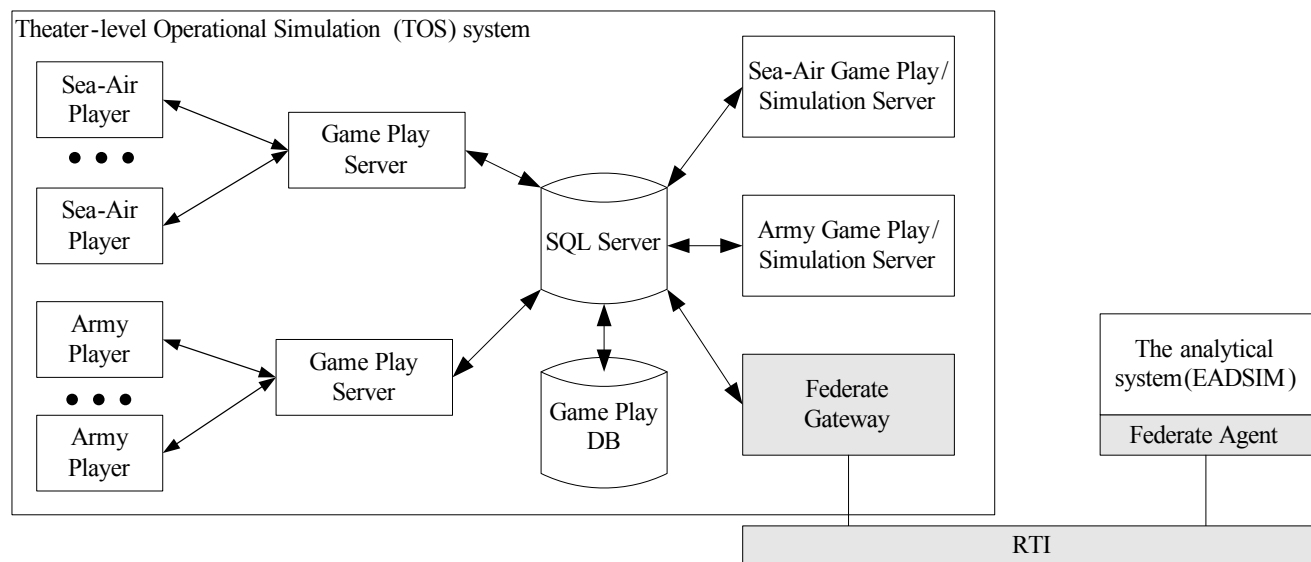


Figure 2. The architecture of the heterogeneous joint simulation environment

4. The Demonstration Scenario

From the above section, we learn that the key issue in successfully using an analytical system as an online decision-making support system is how to promptly calculate the event period of an ongoing event. The event period is dominated by the deadline, t_d , which, in turn, depends upon the type of event, e , that is simulated. Furthermore, this deadline is the latest time that the analytical system should return the analysis result to the operational simulation. Since time t_d is application dependent, a scenario is then designed to demonstrate the process of computing the event period discussed in the previous session. To simplify our study, this scenario is that the red force launches a cruise missile to attack a military base of the blue force. The blue force tries to intercept this attack with anti-aircraft missiles and artillery. Hence, the blue user faces the dilemma of shooting an appropriate number of anti-aircraft missiles to intercept this cruise missile attack. Figure 3 depicts the timing diagram of this scenario.

Assume that the blue force radar detects this cruise missile attack at time t_1 . That is, t_1 is the time t_s of the event period $[t_s, t_d]$ of this hostile event. According to the precomputed aerodynamic model of the cruise missile, we assume the cruise missile will hit the blue force base at time t_5 if the interception fails. Furthermore, based upon the models of the cruise missile and the anti-aircraft missile, we assume the closest intercepting distance for the blue force base is r_1 and, hence, the corresponding penetration time is time t_4 if the interception fails. By summing up the time required to lock and shoot the anti-aircraft missile, the shortest engagement distance for the blue force is r_2 , which gives us the latest time, t_3 ,

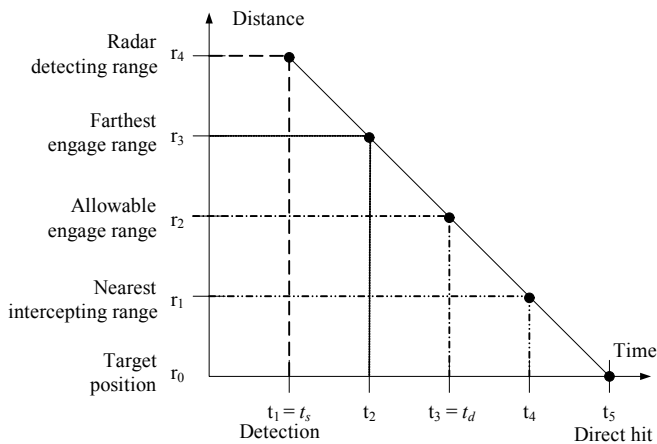


Figure 3. Distance-time relation of the scenario

at which to actuate the analytical system. In other words, time t_3 is the last chance for the blue force user to launch an anti-aircraft missile to intercept this cruise missile. Hence, time t_3 is the time t_d of the event period $[t_s, t_d]$, which gives us the interval $[t_1, t_3]$ as the event period for the analytical system to analyze the user's intention. Since the anti-aircraft missile has a maximum range of trajectory, we can further assume that the farthest intercepting distance is r_3 , and its corresponding time is t_2 . Hence, we can set time t_2 as the first decision-making point and t_3 as the second decision-making point at which the user can decide how to intercept the attack. In other words, $[t_1, t_2]$ and $[t_2, t_3]$ are two possible assessment eras during which the blue force can actuate the analytical system to analyze his intercepting intention.

Based upon this simplified scenario, the cruise missile, its carrier, and the anti-aircraft missile are the three objects that are interesting to both the operational training simulation and the analytical system. That is, the analytical system needs to clone the status of the cruise missile and the anti-aircraft missile from the operational training simulation before the assessment is executed. Figure 4 depicts the object class hierarchy of this scenario with the attributes of each object class listed under the class name. Note, as illustrated in Figure 5, that there are only two interactions between the operational training simulation and the analytical

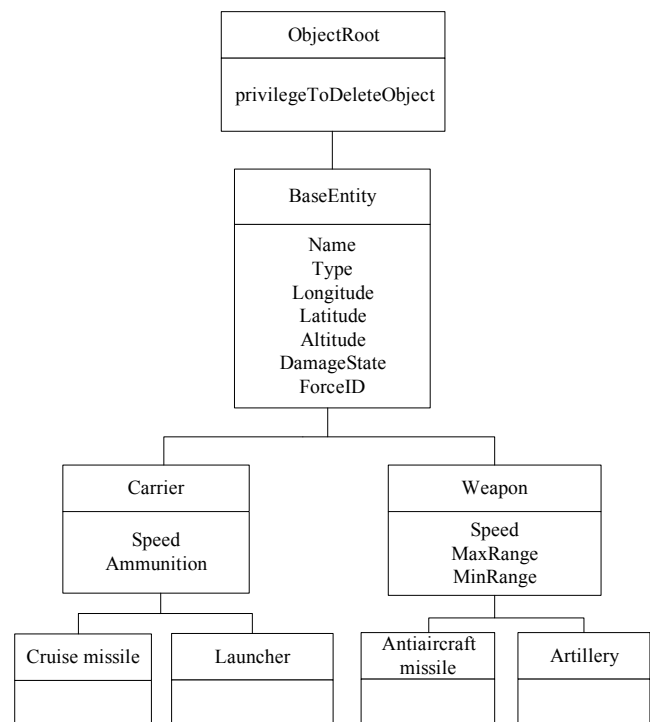


Figure 4. The object hierarchy of FOM

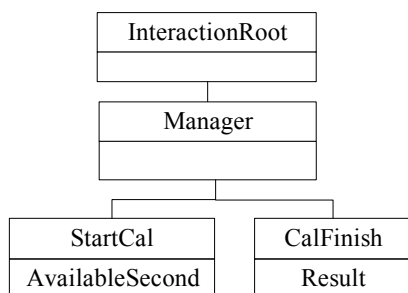


Figure 5. The interaction hierarchy of FOM

system. The first interaction, *StartCal*, is sent out when the blue user presses a button to actuate the analytical system; the second interaction, *CalFinish*, is initiated by the analytical system to return the analysis result to the operational training simulation. Note also that the parameter, *AvailableSecond*, for the interaction class of *StartCal*, is the length of the assessment era, i.e., the absolute value of $[t_1, t_2]$ or $[t_2, t_3]$. In addition, the parameter, *Result*, for the interaction of *CalFinish*, is a value set of the analysis result that needs to be returned to the blue user.

Finally, Figure 6 is a snapshot on the blue user's computer after the analytical system returns the analysis result. The top left-hand box shows that this is the first trial of analysis and the top-right box displays the number of iterations for this analysis. The second box on the right demonstrates that only one antiaircraft missile was used to intercept the cruise missile, and the bottom right-hand box displays that the interception rate for one antiaircraft missile is 57.7%, with time deviation 0 shown on the bottom box in Figure 6.



Figure 6. Snapshot of the analysis result

5. The Performance Study

To further study the relationship between the event period and the assessment era, an experiment was designed and conducted. The time-stepped operational training environment for this experiment was the domestic Theater-level Operational training Simulation (TOS) system, and the analytical system was the Extended Air Defense Simulation (EADSIM) system [19]. The TOS system is an interactive and multisided wargaming training system that models a joint training warfare environment of air, land, and sea. As illustrated on the left-hand side of Figure 2, its architecture is a proprietary client/server architecture with an SQL server to manage and control the dynamic scenario. Hence, in order to integrate the TOS system with the EADSIM system, a federate gateway was designed in the server side. The federate gateway was implemented on a notebook with Celeron 2.4 GHz CPU and 512 MB of memory. For the TOS system, this federate gateway acts like another server that will regularly query the SQL server for up-to-date information on the scenario. On the other end, the EADSIM system plays the role of an online decision-making support system. The EADSIM is an analytical model of air and missile warfare used for scenarios ranging from few-on-few to many-on-many. It allows each platform (such as a fighter aircraft) to be individually modeled, as well as the interaction among the platforms [19]. Although EADSIM supposedly provides an HLA interoperation capability, this function was unavailable to us during the experiment. Hence, a federate agent [20] was designed for the EADSIM system to interoperate with the federate gateway. This federate agent was used to represent and control the EADSIM within the federation execution. The federate agent ran together with the EADSIM on a Celeron 2.4GHz computer with 512 MB memory. The RTI version used in our experiment was DMSO RTI 1.3 NG V6.

The main concern in integrating the time-stepped simulation with the event-driven simulation is to prevent the event-driven simulation from being scaled down into a time-stepped simulation. When an event-driven analytical system is adopted to perform COA analysis, it must execute as quickly as possible to analyze the user's intention. If the analytical system is scaled down to a time-stepped simulation, it cannot perform such an assessment for the user in time. The time-regulating and time-constrained mechanisms in HLA time management [21] services provide mechanisms to solve this problem. When an analytical system is used as an online decision-making support system, it passively receives messages from the time-stepped operational simulation, and it should not

stall the ongoing exercise. In other words, during the federation execution, the logical time advancement of the federate agent for EADSIM should be dominated by the federate gateway of the TOS system. Hence, the federate agent is set as a time-regulating federate whereas the federate gateway is set to be a time-constrained federate.

There are three factors that will affect performance in integrating an analytical system as an online decision-making support system of an operational training environment. The first one is the time required for the federate gateway and the federate agent to clone the dynamic scenario from the operational training system to the analytical system. The second factor is the performance of the federate gateway's interaction with the federate agent in actuating the analytical system. Note that the first and the second factors together constitute time T_T in equation (1). The last factor is the performance of the federate agent, which corresponds to time T_R in equation (1).

The first factor is related to the time interval during which the analytical system clones the dynamic scenario from the operational training environment to perform the analysis. If this cloning process is executed immediately after the user activates the course-of-action analysis, the cloning time may take too much of the event period. That is, T_T may become a significant element of equation (1), which leaves little time to $n * T_A$. To solve this problem, we allow the federate agent to request up-to-date information from the federate gateway periodically. Since the federate agent is an event-driven federate, it can issue a *NextEventRequest()* service every 30 time units before the analytical system is actuated. In this way, it not only receives timely information from the operational training simulation but also synchronizes with the federate gateway every 30 time units. The benefits of this approach are twofold. Firstly, the event-driven analytical system will not be scaled down into a time-stepped simulation. Secondly, the cloning time is reduced to the performance of the federate gateway and the federate agent to update and reflect the attribute values. Our experiments show that the federate gateway took around 0.0015 to 0.0020 seconds per frame to access the SQL database and to issue an *updateAttributeValues()* service. For the federate agent, it took around 0.0015 to 0.0017 seconds per frame to process the *reflectAttributeValues()* callback service. This service will write the received data into the scenario file of EADSIM. After updating the scenario file, the federate agent then invokes the *NextEventRequest()* service call again for the next update. Since the time-step for the federate gateway is 5 seconds, our experiments show that the federate

gateway and federate agent will not cause any significant overhead to the entire simulation. In other words, times T_T and T_R in equation (1) are negligible, and the minimum assessment area is dominated by $n * T_A$.

6. Conclusion

This paper explores the rationale and presents an approach to integrate an analytical system into a time-stepped federation. The purpose of this joint simulation is to study the method of using an event-driven analytical system as an online decision-making support system for a time-stepped operational training simulation. The different rationales of integrating the analytical system with the operational training simulation are fully discussed. Based upon the purpose of our research, we called the interoperation period between these two systems the *assessment era*. During this assessment era, the time-stepped operational simulation and the analytical system can execute at their respective paces asynchronously while cooperating with each other. Our research is based upon the HLA (IEEE 1516) standards. The implemented structure and the time management services for such an integrated simulation are given at the end of this paper. The result of our field test shows that this approach can effectively improve the tactical skill of a user so that he can use the analytical system to evaluate his intention online before actually making his move.

The only complexity in this approach is the calculation of the deadline t_d , and, hence, the event period, since it is application dependent. The event-related parameters have to be considered when t_d is computed. However, since those parameters can be pre-estimated, online calculation of t_d during the operational simulation is still possible. Since it is application-domain specific, a further study to design a knowledge-base system on military applications to compute t_d is currently under investigation.

Finally, our research shows a promising approach of integrating an analytical wargame system with a threat-level operational training simulation environment. Restricted by the available resources, this study only focuses on a domestic proprietary client/server operational training system and the EADSIM system. In addition, since this research is simply a preliminary study of the technique for integrating different types of simulating systems, further design of experiments to investigate the usage of this technique to improve the tactical skill of the trainee is the needed. This subject is currently under investigation but is not presented herein, as it is not within the scope of this paper.

7. References

- [1] Guleyuponglu, S., and H. Ng. "Linking Real-Time and Faster-Than-Real-Time Federations," paper no. 01S-SIW-131. In *Proceedings of the Spring Simulation Interoperability Workshop*, 2001.
- [2] Hanna, J., J. Reaper, T. Cox, and M. Walter. "Course of Action Simulation Analysis." In *Proceedings of the 10th International Command and Control Research and Technology Symposium*. (Accessed July 6, 2006), 2005. Available from: <http://www.dodccrp.org/events/2005/10th/CD/papers/113.pdf>
- [3] Gilmour, D. A., J. P. Hanna, W. E. McKeever, Jr., and M. J. Walter. "Real-Time Course of Action Analysis." In *Proceedings of the 10th International Command and Control Research and Technology Symposium*. (Accessed July 6, 2006), 2005. Available from: <http://www.dodccrp.org/events/2005/10th/CD/papers/073.pdf>
- [4] Matthews, K. "Development of a Course of Action Simulation Capability." Scientific & Technical Report, Australian Government Department of Defence, Defence Science and Technology Organisation. (Accessed July 6, 2006). Available from: <http://www.dsto.defence.gov.au/publications/3437/DSTO-CR-0376.pdf>
- [5] Smith, R. "Simulation: The Engine Behind The Virtual World," *Simulation 2000 Series*, Vol. 1. (Accessed August 5, 2006), 2000. Available from: <http://www.modelbenders.com/papers/sim2000/SimulationEngine.PDF>
- [6] Smith, R. "Strategic Directions in Distributed Simulation," *Simulation 2000 Series*, Vol. 2. (Accessed August 5, 2006), 2000. Available from: <http://www.modelbenders.com/papers/sim2000/SimulationDirections.PDF>
- [7] IEEE Std 1278.1, "IEEE Standard for Distributed Interactive Simulation – Application Protocols." New York: Institute of Electrical and Electronics Engineers, Inc., 1995.
- [8] Wilson, A. L., and R. M. Weatherly. "The Aggregate Level Simulation Protocol: An Evolving System," 781–787. In *Proceedings of the Winter Simulation Conference*, 1994.
- [9] IEEE Std. 1516.1, "IEEE Standard for Modeling and Simulation (M&S), High Level Architecture (HLA) – Federate Interface Specification," i-467, 2000.
- [10] Browsers, A., and A. L. Prochnow. "JTLS-JCATS Federation Support of Emergency Response Training." In *Proceedings of the 2003 Winter Simulation Conference*. (Accessed August 5, 2006), 2003. Available from: <http://www.informs-cs.org/wsc03papers/130.pdf>
- [11] Myjak, M. D., D. Clark, and T. Lake. "RTI Interoperability Study Group Final Report," paper no. 99F-SIW-001. In *Proceedings of the 1999 Fall Simulation Interoperability Workshop*, Orlando, FL, USA, 1999.
- [12] Myjak, M. D., and S. T. Sharp. "Implementations of Hierarchical Federations," paper no. 99F-SIW-180. In *Proceedings of the Fall Simulation Interoperability Workshop*, Orlando, FL, USA, 1999.
- [13] Wood, D. D., and M. D. Petty. "HLA Gateway 1999," paper no. 99S-SIW-051. In *Proceedings of the Spring Simulation Interoperability Workshop*, 1999.
- [14] Filsinger, J. "HLA Security Guard Federate," 1015–1023. In *Proceedings of the Spring Simulation Interoperability Workshop*, Orlando, FL, USA, 1997.
- [15] Chen, D., B. S. Lee, W. Cai, and S. J. Turner. "Design and Development of a Cluster Gateway for Cluster-based HLA Distributed Virtual Simulation Environments," 193–200. In *Proceedings of the 36th Annual Simulation Symposium (ANSS'03)*, 2003.
- [16] Braudaway, W., and R. Little. "The High Level Architecture's Bridge Federate," paper no. 97F-SIW-078. In *Proceedings of the Fall Simulation Interoperability Workshop*, Orlando, FL, USA, 1997.
- [17] Dingel, J., D. Garlan, and C. Damon. "Bridging the HLA: Problems and Solutions," 33–42. In *Proceedings of the Sixth IEEE International Workshop on Distributed Simulation and Real-Time Applications*, 2002.
- [18] Paterson, D. J., E. S. Houglund, and J. J. Sanmiguel. "A Gateway/Middleware HLA Implementation and the Extra Services that can be Provided to the Simulation," paper no. 00F-SIW-007. In *Proceedings of the Fall Simulation Interoperability Workshop*, 2000.
- [19] "Extended Air Defense Simulation (EADSIM)." (Accessed July 27, 2006). Available from: <http://www.asc.hpc.mil/software/info/eadsim/>
- [20] Simon, R., W. S. Chang, and J. M. Pullen. "An Agent Architecture for Network Support of Distributed Simulation Systems," 68–75. In *Proceedings of the Symposium of the Seventh IEEE Distributed Simulation and Real-Time Applications*, 2003.
- [21] Fujimoto, R. M. "Time Management in the High Level Architecture," *Simulation* 71 (6, 1998) 388–400.

Acknowledgements

This research was sponsored by the Joint Training Simulation Center, Minister of National Defense, Taiwan. The authors would like to acknowledge all the helpful military information from the sponsor. The authors also would like to thank Mr. Chao for his programming work for the experiment.

Author Biography

Jiung-yao Huang is an Associate Professor in the Department of Communication Science and Information Engineering at National Taipei University, Taiwan. His research interests include pervasive computing, computer graphics, networked virtual reality, distributed systems, and multimedia systems. He has been a member of IEEE and ACM since 1990 and joined the Society for Modeling and Simulation in 1996. His web address is <<http://web.ntpu.edu.tw/~jyhuang/>>.