Military serious game federation development and execution process based on interoperation between game application and constructive simulators

Changbeom Choi*

School of Creative Convergence Education,
558, Handong-ro, Buk-gu, Pohang-si,
Gyeongsangbuk-do 791-708, Republic of Korea
Email: cbchoi@handong.edu
*Corresponding author

Moon-Gi Seok, Seon Han Choi and Tag Gon Kim

Department of Electrical Engineering,
KAIST,
291 Daehak-ro, Yuseong-gu,
Daejeon 305-701, Republic of Korea
Email: mgseok@smslab.kaist.ac.kr
Email: shchoi@smslab.kaist.ac.kr
Email: tkim@ee.kaist.ac.kr

Soohan Kim

Video Display Biz. Technical Planning Group,
Samsung Electronic HQ,
129 Samsung-ro, Yeongtong-gu,
Suwon-si 443-742, Republic of Korea
Email: ksoohan@samsung.com

Abstract: This paper proposes a development and execution process for military serious game federation, the military serious game federation development and execution process (MSGFDEP). The MSGFDEP utilises interoperation between an existing game application and constructive simulators to extend the serious game. In order to achieve the interoperation between the game application and constructive simulators, we use a high-level architecture (HLA). By interoperating a constructive simulator with an existing game application, a serious game developer can save efforts by extending a serious game application, rather than building a serious game from scratch. The proposed methodology is comprised of two specified process: federation development and federation execution. When the developer wants to build a serious game from scratch, the proposed methodology supports three specified processes: game loop analysis, game agent design, and development. On the other hand, when the developer wants to organise the federation with existing HLA-compliant serious game and constructive simulators, the methodology provides federation synthesis. Finally, the methodology defines the federation execution process to help trainees to obtain more experiences that are realistic.

Keywords: serious game; interoperation; constructive simulator; development methodology; system of system entity structure; federate base.

Reference to this paper should be made as follows: Choi, C., Seok, M-G., Choi, S.H., Kim, T.G. and Kim, S. (2015) 'Military serious game federation development and execution process based on interoperation between game application and constructive simulators', *Int. J. Simulation and Process Modelling*, Vol. 10, No. 2, pp.103–116.

Biographical notes: Changbeom Choi received his PhD from the Korea Advanced Institute of Science and Technology (KAIST) in 2014. His research interests include discrete event systems modelling/simulation, verification, validation, and accreditation and agent-based simulation.

Moon-Gi Seok received his BS in Electrical Engineering from Korea University and MS in Computer Science from the KAIST in 2009 and 2011, respectively. He is currently a PhD candidate at the Department of Electrical Engineering, at KAIST. His research interests include discrete event systems M&S, and parallel and distributed simulation.

Seon Han Choi received his BS and MS in Electrical Engineering from KAIST, Daejeon, Korea, in 2012 and 2014. He is currently a PhD candidate at the Department of Electrical Engineering, at KAIST. His research interests include M&S of multi-fidelity models, cloud system, and big data processing.

Tag Gon Kim received his PhD in Computer Engineering with specialisation in Systems Modelling and Simulation from University of Arizona, Tucson, AZ, 1988. He was an Assistant Professor at Electrical and Computer Engineering, University of Kansas, Lawrence, Kansas, USA from 1989 to 1991. He joined Electrical Engineering Department, KAIST, Daejeon, Korea in Fall 1991 as has been a Full Professor at EECS Department since Fall 1998.

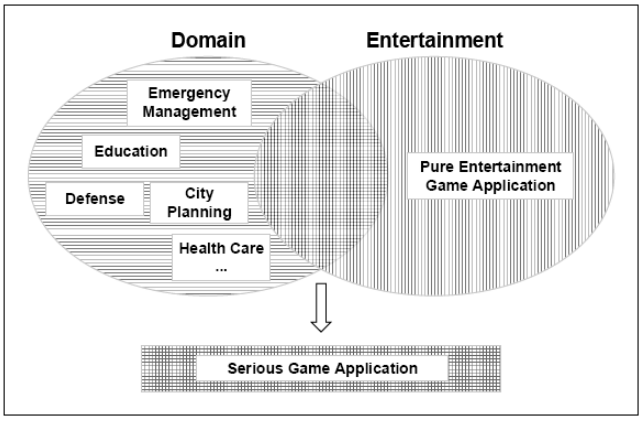
Soohan Kim is the Principal Engineer of Samsung Electronics Co. HQ in South Korea. He is in charge of technical planning of media player and B2B products of Visual Display Division. He joined Samsung in 1993 as Software Development Manager.

This paper is a revised and expanded version of a paper entitled 'Serious game development methodology via interoperation between a constructive simulator and a game application using HLA/RTI' presented at the DHSS 2013, Athens, 25–27 September 2013.

1 Introduction

Over the past few decades, the games industry has grown tremendously in a wide variety of genres. Among various genres and categories of the game, serious games have emerged to educate and train learners, rather than provide entertainment. As shown in Figure 1, a serious game application consists of the domain knowledge part and the entertainment part. The domain knowledge part of a serious game application is for training or educating a player to achieve a desired progress by the player in a certain domain. On the other hand, the pure entertainment part of the game is to attract players into the game, so that the players can attain the domain knowledge. Especially, serious games allow learners to experience situations that are impossible in the real world owing to safety, cost, and/or time (Hudson and Degast-Kennedy, 2009). For this reason, the games industry has developed various types of serious games, including games for military, manufacturing, and medical purposes.

Figure 1 Relation between domain knowledge and game application in serious game



In the military field, several commercial serious games, such as Virtual Battle Space 3 (Bohemia Interactive Simulations, 2014), Military Open Simulator Enterprise Strategy (United States Army Simulation and Training Technology Center, 2014), and Delta3D (Delta3D, 2014) are already available in the market. The main purpose of these games is to train the players by simulating various military situations. In order to provide various military situations, these games provide functionalities to edit terrain and scenarios to create specific war environments. Nevertheless, a problem with these serious games is the limited accessibility granted to their operating systems; thus, these games only enable war scenarios and terrain within a limited scope (Fong, 2006). In addition, these games are based on game engines that lack functionalities to develop new tactical scenarios, modelling the combat entities, and reuse of such entities (Park et al., 2010). Such limitations restrict more detailed and expandable representations of military simulation.

To overcome the precedent limitations, we propose a methodology to develop or enhance military serious games, which utilises existing constructive simulators as dynamic scenario generator. The constructive simulation refers to a simulation that involves simulated people operating in simulated systems. In constructive simulation, real people make inputs to such simulators but are not involved in determining the outcomes (US Department of Defense, 1998). In other words, multiple simulation models are mutually interacts in the synthetic combat environment, so that the constructive simulator can generate combat situations dynamically based on the simulation progress (Tremori et al., 2013). Since the scenario interpreter of existing serious game applications generates training situations based on the given scenarios, the player of a serious game can easily accustom to the given scenario. However, constructive simulators can help the players to experience dynamic military situation based on the interactions between simulation models. Therefore, we have paid attention to separating military serious game from static scenario interpreter.

In our approach, we utilise constructive simulators as a scenario interpreter of the serious game, so that the simulators can generate dynamic situations based on the users' requests or actions. Specifically, trainers can generate

war scenarios through a constructive simulator and the simulator sends situation of battlefield to the game application based on the trainee's interaction. Therefore, the point of this study is to propose a methodology to build or extend a serious game for military training by reusing pre-existing simulators and game applications. In this paper, we propose the military serious game federation development and execution process (MSGFDEP). The MSGFDEP is a development process based on the IEEE standard 1516.3 and it supports a user to extend a serious game, not in ad hoc approach but in formal standardised approach. The primary purpose of the MSGFDEP is to provide not only a process, but also facilities that assist the existing game application in utilising expandable war scenarios.

In our empirical study, we achieved time synchronisation and data conversion based on the high-level architecture (HLA)/RTI. By interoperating the constructive simulator with the existing serious game application, serious game developers can save effort by extending a serious game application, rather than building a new serious game from scratch; in addition, trainees can acquire more experience that is realistic. As a case study, we built and developed a military training scenario for a nuclear/bio-chemical (NBC) situation. The outcomes of the case study will show the usefulness of the proposed work, such as how the flexibility and reconfiguration of the war game scenario improve, as well as how effectively the users can train within the scenario. The successful execution of this study can offer an immediate application for military training, and is particularly suited to war scenarios based in the Korean Peninsula.

The remainder of this paper is structured as follows: Section 2 introduces existing serious game with constructive simulators and its development methodology; Section 3 describes theoretical backgrounds of proposed methodology; Section 4 explains the proposed game federation development methodology for military domain, MSGFDEP; Section 5 illustrates empirical experience of the MSGFDEP, Section 6 we briefly share what we learned during the empirical study and finally, Section 7 concludes this study.

2 Related works

In this section, we will first introduce the existing serious games with constructive simulators. Then, we introduce the development methodology for serious game.

2.1 Serious game with constructive simulators

In recent years, various researches have been proposed to utilise constructive simulators in the serious game. Among them, we selected four recent representative studies and compared them and Table 1 summaries the characteristics of the previous researches.

Möller et al. (2005) presented a serious game for flight defence based on interoperation between a flight simulator game and an air logistic simulator using the IEEE 1516 standard. In order to support interoperation, Möller et al. have proposed FS2HLA architecture, which utilises the RPR-FOM (Shanks, 1997) and developed HLA bridge to utilise serious game as a part of the HLA-federation. The interoperation method of Möller et al. (2005) is similar to our proposed methodology, however, our proposed methodology distinguishes the development process of a serious game federation and propose architecture of serious game agent (SGA) in order to interoperate multiple constructive simulators in the serious game federation.

Table 1 Comparisons of previous researches

Research	*Application field*	*Method*	*Description*
Möller et al. (2005)	Military	Interoperation	Interoperation with the IEEE 1516
Ryan et al. (2005)	Emergency management	Interoperation	Interoperation using database
Metello et al. (2008)	Emergency management	Integration	Integration with suggested simulation engine in a serious game
Massei et al. (2013)	Marine Port Training	Interoperation	Interoperation with the IEEE 1516
Garro et al. (2013)	Disaster management	Integration	Integration in the proposed serious game architecture

Ryan et al. (2005) suggested a serious game for emergency situation management using the unreal game engine and a simulator, which simulates patient vital signs. In order to interoperate the serious game and constructive simulator, Ryan et al. (2005) proposed an interface between the engine and the simulator, called the Gamebots (Adobbati et al., 2001). The Gamebots is a package for the unreal tournament engine that provides an interface to control the characters inside the game. In order to interoperate the serious game and a patient simulator, a simulator generates the medical data and stores them to the MySQL database. Then, the GameBots utilises the data to visualise the patient's behaviours. The restriction of the method is that there is no formal development process to develop a serious game federation in other domains. Moreover, the GameBots uses the database to visualise the patient simulator's results, so there is no interaction between the serious game and the constructive simulator.

Metello et al. (2008) presented a simulation engine integrated in a serious game of emergency management to execute simulation models during the game play. Its simulation engine supported time synchronisation and data transmission, but the simulation engine should support the simulation models. Therefore, if a user wants to modify the simulation model or wants to extend the given serious game, the user may redesign and implement the serious game.

Moreover, if a user wants to apply the system to the other domain, the user may develop entire game system since the simulation engine and the serious game are tightly integrated.

Massei et al. (2013) introduced a real-time distributed simulation environment for training based on the HLA/RTI. In the environment, they proposed a federate called simulation time virtual port simulator (ST-VP). The ST-VP is fully containerised real-time distributed HLA simulator, which reproduces operation in the marine port. The simulator supports a user to tailor the scenarios and reconfigure the participating training simulators. Especially, various simulation entities share the same virtual environment information using HLA/RTI. As a result, it is possible to apply distributed simulation to various simulation entities and may provide low-cost training sessions to the trainees.

Finally, Garro et al. (2013) proposed an architecture of a serious game for disaster management. In the architecture, the authors have proposed a serious game architecture for disaster management. The architecture provides a simulation-based game system for training, and the simulators have integrated into the serious game, in order to give immersive experience to the users. In their researches, they provided an immersive simulation-based game system for disaster management rather than providing serious game development methodology.

To sum up, all four previous researches have focused on practical issues and solved the data exchange of serious game and constructive simulators using ad hoc approaches. To the authors' best knowledge, there has been no specified methodology of development serious game with constructive simulators. Especially, development based on integration such as Metello et al. (2008), Adobbati et al. (2001) needs a lot of time to upgrade or maintain of the given simulators.

2.2 Serious game development based on the model driven engineering

Game-based learning utilising serious game can be effective to attract the interest of a new generation of digital native learners. However, developing a serious game from scratch hinders domain experts to develop or customise serious games, owing to the limited availability of serious games and high-level design tools. In order support the domain experts, the model-driven engineering (MDE) has been applied to the serious game development. MDE is a software development approach that utilises the model concept to represent features of software and it supports automatic transformation to generate refined software artefacts. This approach is known as one of the software development approaches that reduces the platform complexity and expresses domain concepts effectively (Schmidt, 2006).

Specially, Tang has proposed a concept of model driven game development framework (Tang and Hanneghan, 2010). The framework utilises the game contents model (GCM), game technology model (GTM), and game software model (GSM) in order to apply MDE in serious game development (Tang and Hanneghan, 2011; Tang et al., 2013). The GCM represents the logical design specification of a serious game, such as game structure, game presentation, game simulation, and game scenario. GTM represents serious games in a manner that is computationally independent of any operating platform. Finally, GSM represents the transformed model of the serious game specific to a chosen technology platform. In Tang's framework, modules load the GCM and represent the game as a software model. The GTM is read by the GSM transformation modules replaces game logical model into physical game software constructs using specific game software framework such as unity, unreal and XNA by replacing constructs with the corresponding. Finally, the GSM is interpreted by a source code generator and composes the software code. In order to support platform independent serious game development, the GTM separates the game specific part and core technology part. Therefore, the developer can develop platform independent serious game by choosing core technologies, which implements the game specific part.

Although the Tang's framework supports domain experts to build a prototype of serious game and make further improvement on the designs of serious games without facing the technical barriers during the development phase, it has previous mentioned limitations. In the framework, each game scenario is defined as a setup using a selection of game objects to create an environment, a sequence of game events, and a set of game objectives that challenges both player skills and knowledge of the game. In other words, the framework utilises the static scenario interpreter to provide educational contents. Therefore, a trainer of a generated serious game based on the framework may suffer to develop dynamic training situations. Moreover, it may difficult to extend a serious game which was generated by the framework to interoperate among heterogeneous systems, since the framework did not consider the interoperation concept.

3 Background

In this section, we introduce the federation development and execution process (FEDEP) and system of system entity structure (SoSES). Since our methodology is based on the IEEE standards, we extend the existing development process to the serious game domain. Furthermore, our methodology adopts the system of SoSES representation in order to enhance the reusability of federates.

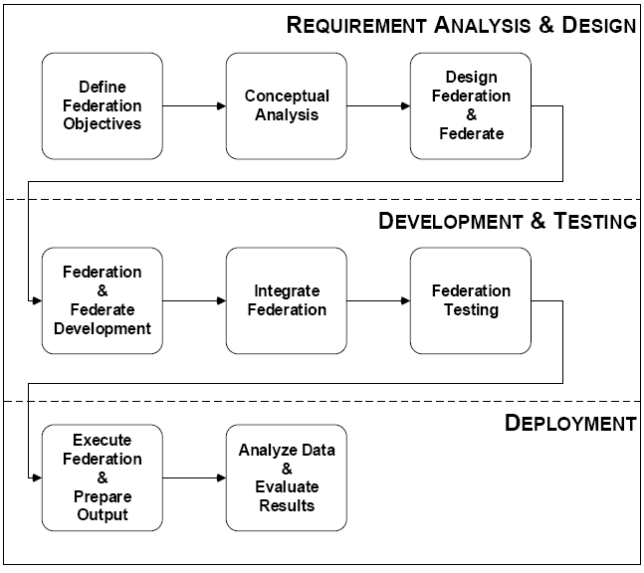
3.1 Federation development and execution process

In the modelling and simulation fields, HLA has been approved as an IEEE standard to specify interoperating, heterogeneous simulations within distributed environments. In this standard, a standalone simulator is called a federate, and the set of federates that comprise a larger system to

achieve the same purpose is called a federation (IEEE, 2010). If a simulator is compliant with HLA protocols, we call it an HLA-compliant simulator. The FEDEP is a recommended development process used to develop HLA-compliant simulators and federations (IEEE, 2003). FEDEP is a standardised and recommended process for developing interoperable, HLA-based federations.

As shown in Figure 2, a developer should first define the objectives and requirements of the federation and confine the scope of the federations development to the identified requirements. When the objectives of a federation are fixed, a developer should perform a conceptual analysis of the target system. Then, the developer will design and implement the federation and each federate. In this phase, the developer should identify the input/output data of each federate, in order to create the simulation object model (SOM). The SOM contains the types of data that the simulator will exchange during the simulation. The federation object model (FOM) is the set of SOMs that constitutes the federation data. After identifying the dataset, the developer has three options: use an existing federate, develop a federate from scratch, or modify a legacy simulator into an HLA-compliant simulator. After federates are implemented, the developer may integrate them into a federation and test it. After integration and testing, a user executes the federation and analyses the data. Finally, federates are revised based on the analysed results.

Figure 2 Phases of federation development and execution process



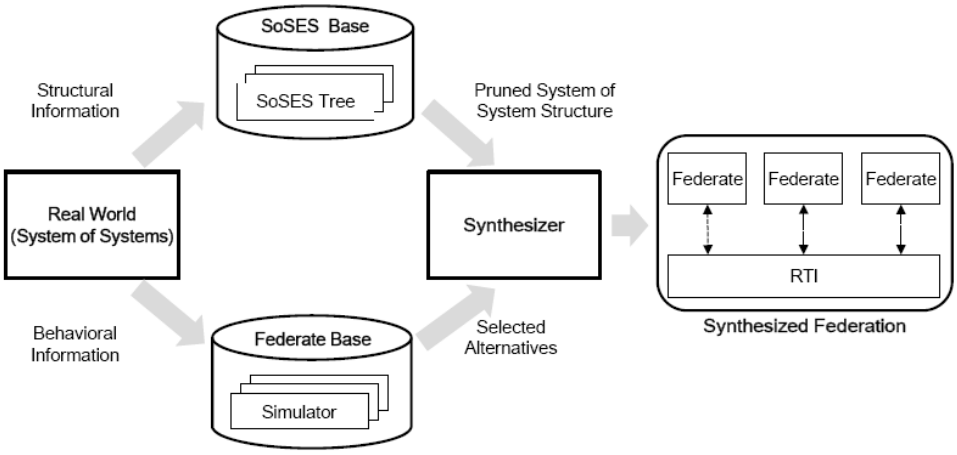
However, the FEDEP is insufficient for developing a federation among the serious game and the constructive simulations for several reasons. First, a serious game has usually been implemented already; thus, it is almost impossible to modify the game application to support HLA protocols. Second, the time units of the serious game and the constructive simulator may be different; thus, the developer must tune the time resolution between them. For example, the default time unit of a constructive simulator may be hours, but the default time for a serious game may be milliseconds. Therefore, time synchronisation between a serious game and constructive simulators is different from interoperation between simulators. Third, standard distance values that differ between the serious game and the constructive simulator should be calibrated. For example, the standard distance value of a constructive simulator can be in kilometres, and the space of the training ground can be 100 m^2 or more. However, such a training ground will hinder the training experience in a serious game. Usually, the designer of a virtual training ground wants to maximise training experience; therefore, training grounds are usually relatively small and bounded.

3.2 System of SoSES/federate base framework

A real world system has flexible structures and may have various alternatives components to organise the system. To capture the characteristics of a real world system, multifaceted system modelling was proposed (Zeigler, 1984). The multifaceted system modelling is a modelling method that represents all the components and alternatives of a system, and it models the real system either an individual system or a collection of individual systems, which is called system of systems. One of the methods to model individual system in multifaceted system modelling and simulation framework is SoSES/model base framework (Kim et al., 1990). Whereas, to model the system of systems, the system of SoSES/federate base (FB) framework was proposed (Kim et al., 2013). The SoSES/FB framework is extension of SES/MB based on the HLA/RTI. In order to model the system of systems, the SoSES framework provides the SoSES tree to capture the structure of a collection of systems and utilises HLA/RTI to simulate the systems in a distributed environment. As shown in Figure 3, the SoSES base stores SoSES trees constructed from SoSES formalism. Each SoSES tree denotes a federation and the SoSES Base stores the requirements of alternative federates which are qualified to the federation. Moreover, the FB has executable HLA-compliant federates and its data. When a SoSES tree is selected, system entity nodes are replaced into the executable federates in the FB.

However, SoSES/FB alone is not suitable for developing or extending a serious game via interoperation. Unlike typical federation development, serious games for military training development should consider the user's behaviour and the time synchronisation between a game and its simulators. Generally, when the designers of a serious game for military training build a virtual training field, they arrange the virtual objects to maximise the training experience. As a result, the size of a virtual training field is relatively small, and the placement of the virtual objects leads the user to acquire virtual training experience. On the other hand, the objective of a constructive simulator is to acquire reliable simulation results from the simulation models. Therefore, the developer should narrow the gap between the serious game and the constructive simulator, in order to build and extend the serious game for military training via interoperation.

Figure 3 Concept of SoSES/FB



4 Military serious game federation development and execution process

Before moving to the central part of the MSGFDEP, we must identify the components of the serious game for military training and their roles. The proposed serious game for military training consists of an existing serious game that provides virtual battlefield situations for training and several constructive simulators to describe the situations in detail. Let us suppose that trainees exercise military operations in urban terrain (MOUT) using the proposed serious game for military training. In this case, the existing serious game provides battlefield situations, such as the number of soldiers and the constructions that are involved, while the constructive simulators compute numerical calculations, such as atmospheric diffusion and damage assessment. During a simulation, the calculations of the constructive simulators are reflected in the serious game. Consequently, the separation between the existing serious game and constructive simulators enables to reuse individual components, and trainees can experience expandable battlefield situations easily by communicating with various constructive simulators in the existing serious game. From the viewpoint of system engineering, the serious game for military training is considered to be a system of systems. Therefore, developing a federation that consists of a serious game and a constructive simulator and building a system of systems are alike. When a developer wants to build a federation, the SoSES/FB framework supports the federation synthesis process based on its objectives. The SoSES denotes the structure of the federation and helps the developer to choose which federate will join the federation. After the user selects a federate, the framework automatically bring federates from the FB and synthesises a federation from them. In other words, the SoSES is a blueprint of a federation, and the FB is a repository for federates.

Figure 4 shows the flowchart of the MSGFDEP. First, the developer should consider the objective of the federation and perform conceptual analysis. During the conceptual analysis, the developer must consider which serious game application and constructive simulators should form a federation. In this phase, the developer decides to develop a game agent or federate from the beginning or utilise existing federates from the FB. The differences between FEDEP and the MSGFDEP can be characterised by the federation synthesis, game agent development, and parameter tuning phases. In the following section, we will explain each phase in detail.

4.1 SGA development process for serious game for military training

As shown in Figure 5, if any SGA is not available in the FB, the developer should design and implement the SGA. In our previous study, we have designed a SGA to interoperate a serious game and a constructive simulator. In this study, we considered multiple constructive simulators that participate the interoperation among a serious game and constructive simulators. Therefore, a user should consider the design and implementation of SGA based on which constructive simulators are participating the serious game federation.

The design and implementation of a SGA is similar to development of a federate in the FEDEP. However, the proposed development methodology extends the FEDEP. The differences between the FEDEP and the MGSFDEP are in the game loop analysis, game agent design, game agent development, and parameter tuning phases.

4.1.1 Game loop analysis

In order to interoperate a constructive simulator and a given serious game application, the developer should identify the necessary information for the constructive simulator and game application. For example, the constructive simulator should know the position of the user participating in the virtual training, and the game application should know the states of the users, which are determined by the constructive simulators. In order to acquire such information, the developer should understand the game loop of the game application. As described by Valente et al. (2005), input data acquisition, data processing, and rendering occurs simultaneously while the game is running; in order to

handle the process, the game loop is made up of the read player input, update, and render stages. Therefore, in order to interoperate a serious game application and a constructive simulator, the simulation results from the constructive simulator should be reflected before the render stage. In order to reflect the simulation results before the render stage, the developer has two options: modify the server structure of the serious game or modify the client program of the serious game. For example, the former option may involve inserting additional game logic into the update stage, while the latter option may reflect the simulation results during the read player input stage. Between these two options, the former option may be more suitable for implementing interoperation features into the serious game; however, the latter option may be more suitable for cases in which the server and the client of a serious game have already been developed. In this study, we assume that the client and the server of the serious game have already been developed. To tackle this problem, we built a special client for a serious game application, which is SGA, so that the client surrogates the constructive simulator to reflect the simulation results to the serious game. Therefore, in the game loop analysis phase, the developer should understand the protocol between the server and client of the serious game.

4.1.2 SGA design/development

When the analysis of the game logic of a serious game application is finished, the developer should design and develop the SGA. As mentioned earlier, the SGA is a gateway for the game to interchange information between multiple constructive simulators and a serious game. The objectives of the SGA are to manage the mapping between the information from the serious game and the information from the constructive simulators, and transfer the information to the other side as quickly as possible. Therefore, the developers of the SGA may focus on the data conversions between the serious game and constructive simulators, rather than rendering the objects in the serious game. Figure 6 shows the architecture of the SGA.

Figure 4 Flowchart of the MSGFDEP

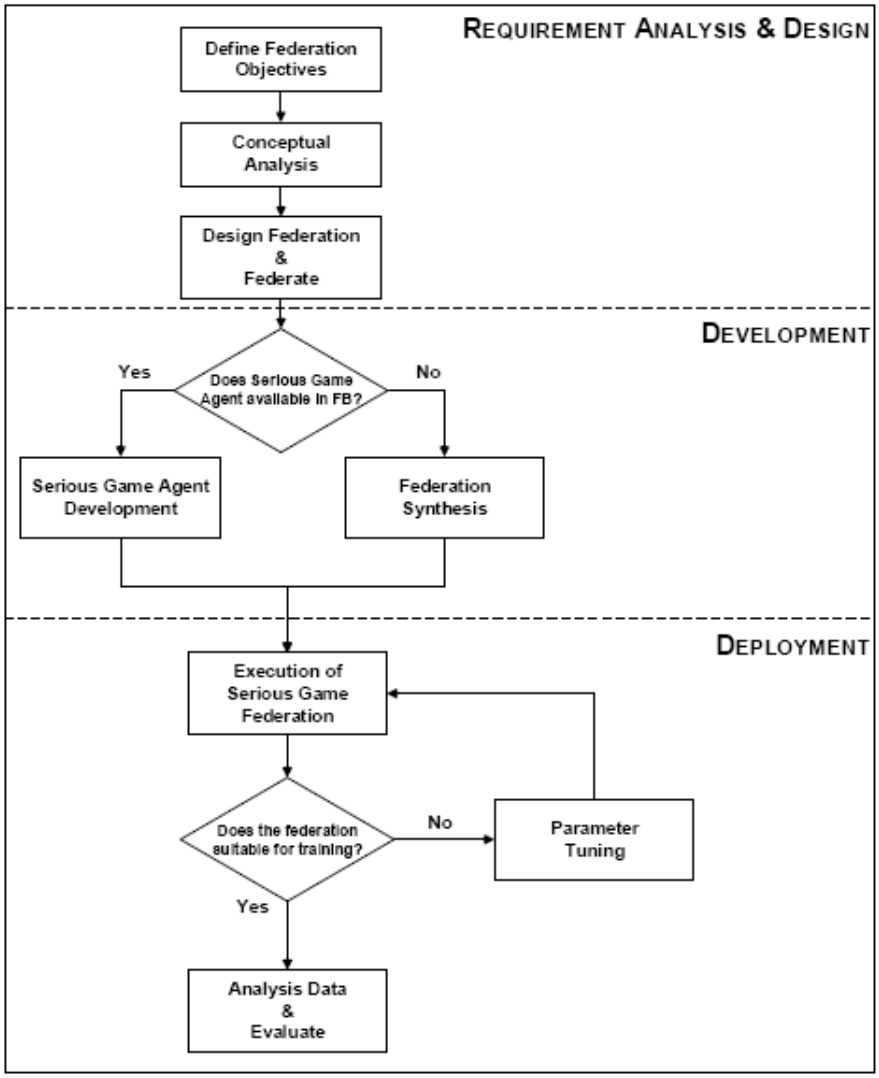


Figure 5 Development and execution process for SGA

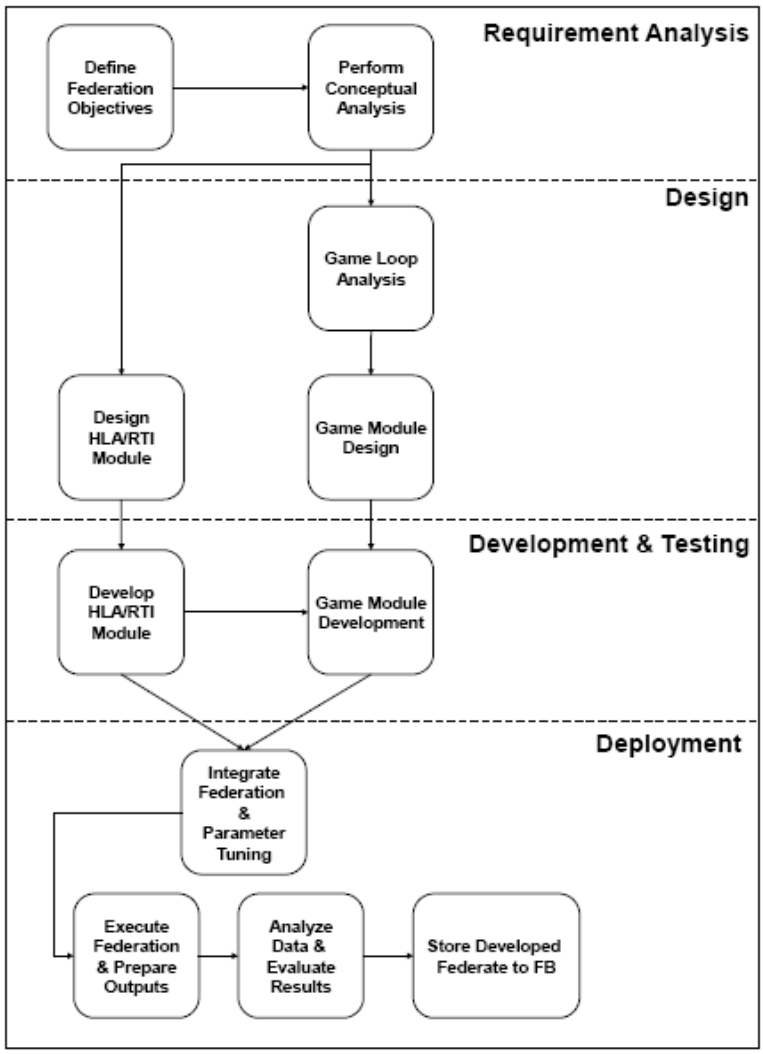


Figure 6 Architecture of SGA

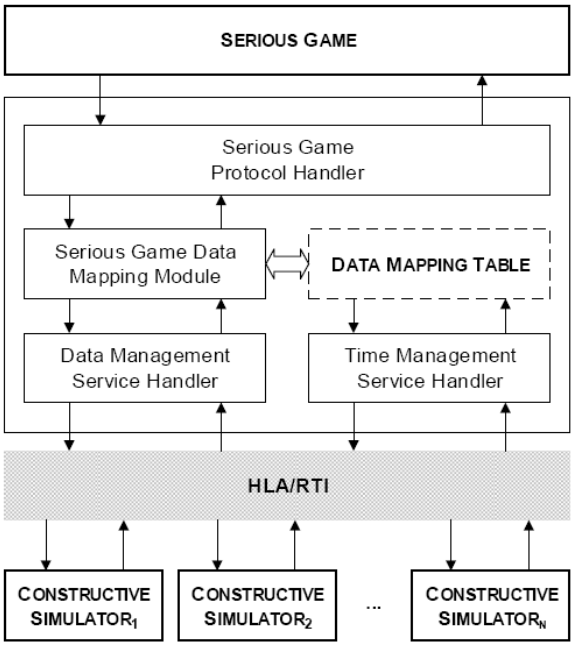
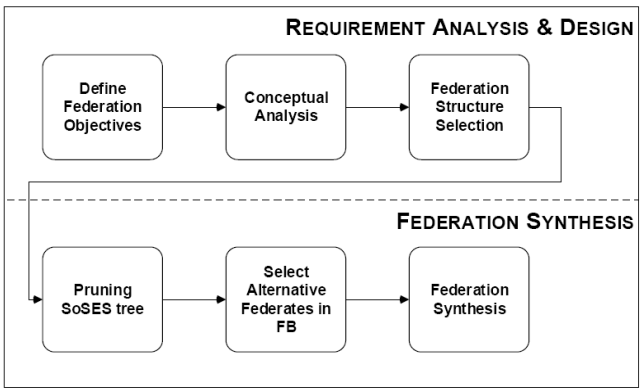


Figure 7 Synthesis process for serious game federation



The SGA has four primary modules to interface between the serious game and constructive simulators, the data management service handler, the time management service handler, serious game data mapping module, and serious game protocol handler. To manage the HLA services, the data management service handler and the time management service handler handles the invocation of HLA services and the HLA service callbacks. The serious game protocol

handler processes the data conversion during the server and the client of the serious game.

During the execution of serious game federation, the SGA receives the information from HLA/RTI and transfers it to the serious game. Similarly, the SGA receives the information from the serious game and transfers it to the HLA/RTI. Since the serious game and HLA/RTI use different network protocols, the serious game data mapping module processes the data conversion. Especially, we have separated the data mapping table from the conversion modules, the developer may reuse the SGA in other serious game federation. Finally, when the development of the game agent is finished, the developed game agent is stored and federates to the FB.

4.2 Federation synthesis process for serious game for military training

When a developer decides to reuse the simulation federates and the serious game, which are stored in the FB, the SoSES/FB framework supports the developer in synthesising the federation, based on the developer's requirements. As shown in Figure 7, the synthesis process for serious game federation follows the flowchart of MSGFDEP.

After the conceptual analysis is finished, the developer may decide which federation structure is the most appropriate to the federation objectives. When the federation structure is selected from SoSES base, the SoSES/FB framework supports the developer to prune the SoSES tree. Since the SoSES tree denotes the structure of chosen federation and the alternative federates, the user may select the alternative federates which are qualified to the objectives of the federation. After the pruning phase, the SoSES/FB framework fetches federates from the FB, and synthesises them into a serious game federation.

4.3 Execution process of military serious game federation

Since the objectives of game application and the constructive simulators are different, the developer should consider the human factors. Before we discuss this phase, we need to analyse the characteristics of the serious game and the constructive simulators. The objectives of a constructive simulator are to measure and analyse the performance index of a simulation model. A developer designs and implements the constructive simulator to obtain reliable simulation results. Therefore, the simulation time and simulation space must reflect the real world. However, the scales of time and the space are relative to the users. For example, the speed of a vehicle in the simulator may be denoted as km/h, which is important because the data affects the simulation results.

On the contrary, the trainers of a serious game will not consider the exact speed of a vehicle; they may regard the relative speed as more important. Moreover, the distances between objects may differ. If the simulator uses a different distance scale in the serious game, the simulator may generate unintended simulation results. In contrast, if the serious game utilises the distance scale of the simulator, the trainee may become bored, because implementing a training field with real scales will generate an enormous virtual training field. As a result, the developer should consider the scales of time and space and tune the parameters iteratively, until the requirements and implementation of the federation are met.

5 Case study: NBC evacuation training simulator

This chapter will detail our empirical research. In order to generate dynamic situations during serious gaming, we utilised a virtual world application called In-world Editor as a serious game and the chemical gas simulator, damage assessment simulator, and sensor simulator as constructive simulators. First, we will introduce the serious game application and the constructive simulator. Then, we will share our experience about interoperating both of them.

5.1 Serious game application

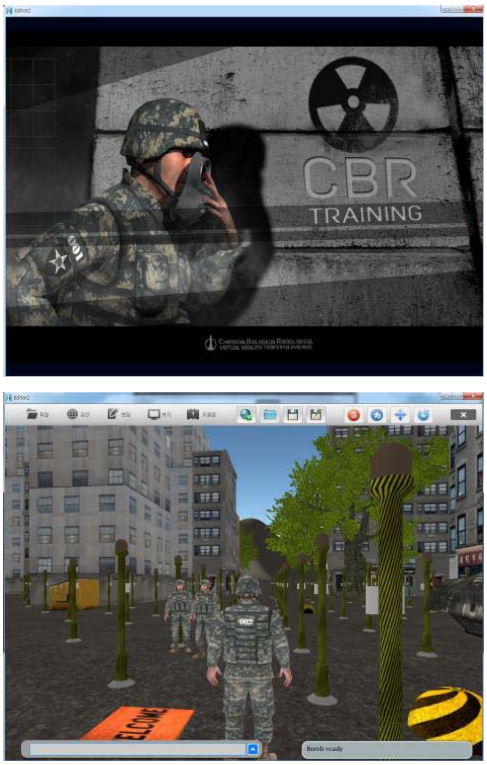
The In-world Editor is a virtual world application based on the Unity 3D Engine and Photon server application (Unity Technologies, 2014; Exit Games, 2014). In order to provide a sense of reality with the well-built virtual training environment, the user can rearrange the objects during the game play. Moreover, the application supports the script, so that the object in the virtual world can interact with the users. In addition, it supports multiple users to interact with each other. Each user shares a virtual training environment and gets training with other users through each client. They can allocate virtual objects to the field and arrange the position of the object, which other user has allocated. Following Figure 8 shows the screen capture of the In-world Editor.

To provide the more immersive experience to the users, this application provides some functionality to build a virtual training environment. First, user can allocate and remove the various virtual objects freely such as buildings, cars, trees, sensors, bombs, and so on. These virtual objects are in the object list, and user can design and add a new object to the list. All the objects in the virtual training environment are saved as XML files in the photon server and the server manages them. The client just shows them to the user. Therefore, if the virtual training environment is modified, all users can be trained in the identical training environment. Following Figure 9 shows the screen capture of the before-and-after object allocation and object list.

Furthermore, user can interact with allocated virtual objects. For example, if the obstacle object is allocated on the road, user cannot go through that obstacle objects. Therefore, the trainer can lead the trainee to the training contents. In addition, user can interact with some objects with mouse click such as detonating a bomb, activating a sensor, and so on. Using this, trainer can plant a bomb and detonate the bomb at any time. In addition, the In-world Editor supports trainer to wear a gas mask and the trainer

who wears a gas mask can survive for a longer time in chemical cloud. Using these objects, trainer can build a well-built virtual training environment. Additionally, the In-world Editor supports administrative functionality. The trainer can use script command to control the virtual world environment. Following Figure 10 shows the screen capture of the interaction with objects, such as detonating a bomb, wearing a mask.

Figure 8 Screen shots of the serious game client (see online version for colours)



The In-world Editor provides the interaction between users and the virtual world object through 3D graphics; however, it cannot give the realistic simulation results to the trainees. For example, if the trainer wants to build a training scenario for the trainee to handle the evacuation situation during the chemical warfare, the developer should modify or insert the game logic for chemical warfare. In order to extend the functionality without changing any game logic of the In-world Editor, we utilise the chemical gas simulator, by interoperating them.

5.2 Constructive simulators

The serious game visualises entities like trainees, buildings in the virtual training field. However, the serious game cannot update states of entities based on state transition equations or rules. To update states of entities in virtual training field, additional constructive simulators are necessary. In the proposed virtual chemical evacuation-training environment, the constructive simulators, which consist of a chemical gas simulator and a damage assessment simulator, are inserted to the federation to update densities of chemical compounds and the dosage amounts of a trainee.

Figure 9 Object allocation and object list in the serious game client (see online version for colours)

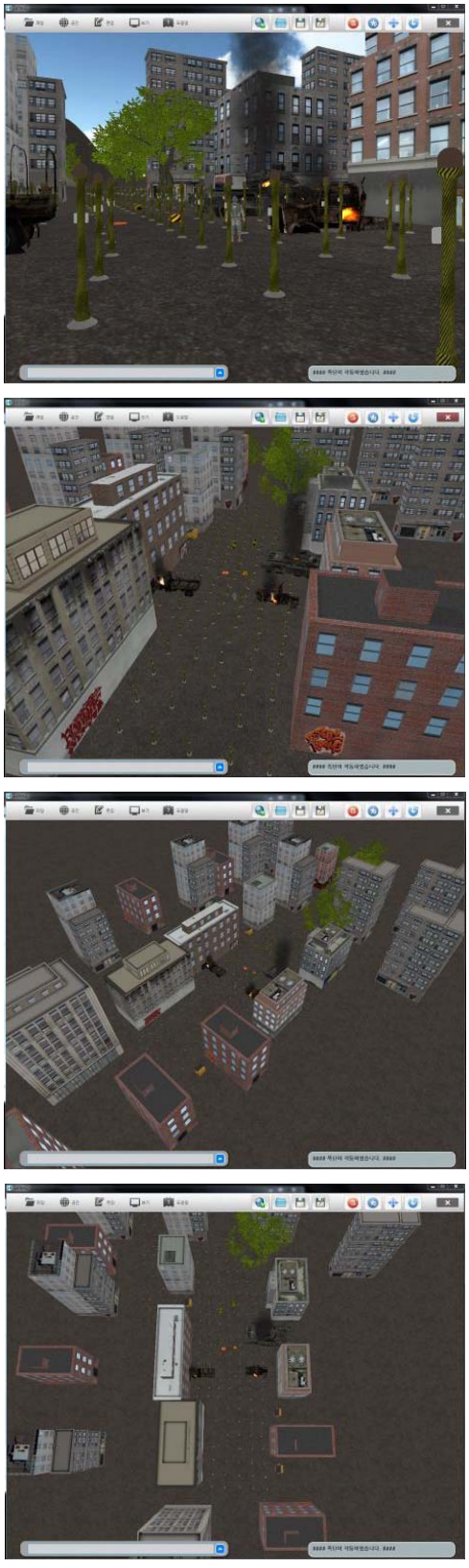
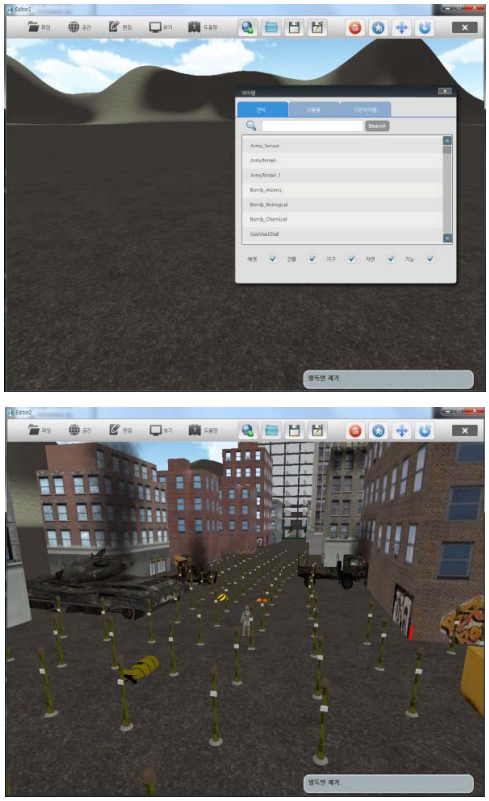


Figure 10 Interaction with objects in serious game client (see online version for colours)



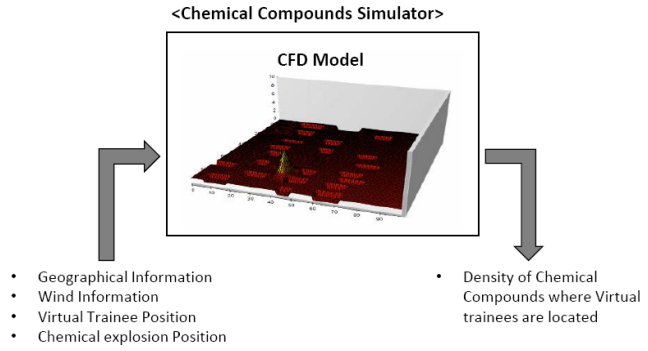
5.2.1 Chemical gas simulator

To obtain a realistic distribution of chemical compounds, the effect of geographical features and wind should be considered when chemical gas flows. The modelling and simulation of chemical diffusion considering such effects have been studied in the computational fluid dynamics (CFD) research area. Depending on the purpose of the virtual training and the computation power of the execution environment, the chemical gas simulator should simulate a CFD model based on a numerical method which satisfy the real-time constraints. Before simulating the CFD model, the virtual training field should be discretised into grids, and the farfield boundaries and the solid boundaries of each entity in the virtual training field should be determined. Then, the CFD model computes the states of grids and boundaries iteratively based on the governing equations, boundary conditions, and states of the neighbouring grids as the simulation time advances. During the computation of farfield boundaries, wind factors such as direction and velocity are reflected to the simulation, so that the diffusion pattern of chemical compound may be influenced by the obstacles. Among computed states, the density information of grids where virtual trainees and sensors are located is transferred to damage assessment simulator and sensor simulator.

5.2.2 Damage assessment simulator

To evaluate the damage of each virtual trainee from chemical compounds, the damage assessment simulator is required. The damage assessment simulator calculates accumulated chemical compounds of each virtual trainee based on the density of chemical compounds in its position, and the defence rate depending on wearing mask or not. The accumulated chemical compounds of some virtual trainee exceed the defined threshold, the simulator sends the trainee-death message to the serious game.

Figure 11 Conceptual figure of chemical simulator (see online version for colours)



5.2.3 Sensor simulator

To evaluate the ON-OFF condition of sensors, the sensor simulator was required to participate to the federation. The sensor simulator continuously monitors the densities of chemical compounds where sensors are located, and if the density of the chemical compound exceeds the defined sensible threshold, the simulator send Sensor-ON message to the game engine. If the density of the chemical compound diminishes under the threshold and the state of positioned sensor is ON, the simulator send Sensor-OFF message to the game engine.

Figure 12 Conceptual figure of damage assessment simulator

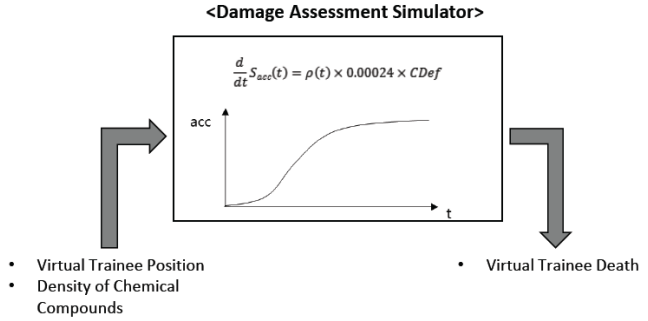
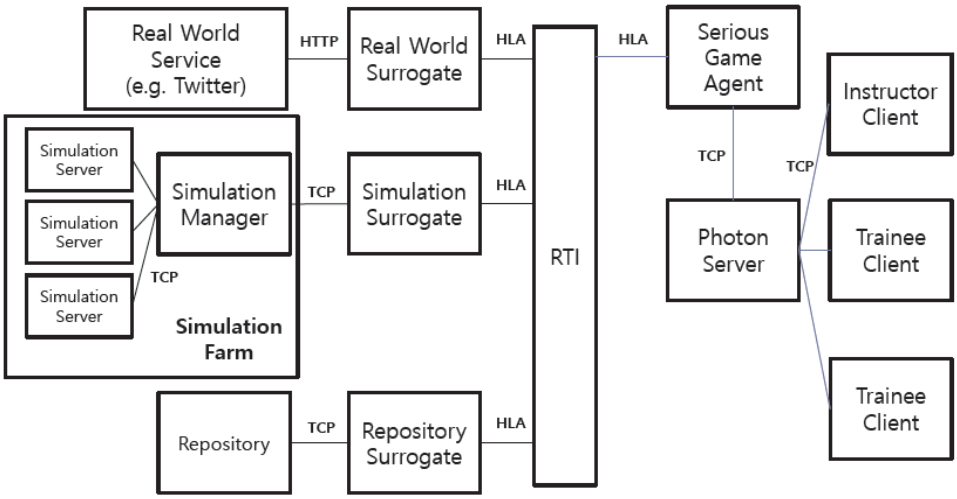


Figure 13 Architecture of serious game federation for NBC evacuation training



5.3 Interoperation between serious game and constructive simulator

In this section, we will introduce the technologies applied during the interoperation between the game application and the constructive simulator. Figure 13 shows the architecture of the serious game federation for NBC evacuation training. In this serious game federation, it provides the real world service, such as chatting and broadcasting the current training status, and simulation services. Especially, several simulation servers handle the NBC simulation during the virtual training based on the HLA/RTI. As we mentioned earlier, we interoperate the serious game and constructive simulators based on the HLA/RTI, the SGA interfaces the HLA/RTI so that the serious game can be assumed as a federate.

During the development of the serious game federation, the SGA was not provided. Therefore, we have adapted SGA development process to develop the game agent to support the In-world Editor. Figure 14 shows the documents used during the game loop analysis phase. In order to speed up the pace of development, we utilised PowerPoint documents to determine the data structure between the constructive simulator and the serious game.

Figure 16 shows the calibration concept during the parameter tuning phase. The left portion of Figure 16 shows the geographical features that the trainer has arranged. In order to control the path of the evacuation, the trainer may place more virtual objects. The right portion of the figure shows that the constructive simulator has received the geographical features from the serious game. Since every client in the game should receive information about the objects, which are allocated in the virtual space, the SGA receives the information and transfers the data to the HLA/RTI. Figure 15 shows the part of FOM for the introduced serious game federation. The environmental situations at the training field may change during the training and it may influence the simulation results and immersive experience to the trainees. In order to reflect such information, we have identified environmental information, as shown in Figure 15, and SGA converts the environmental information using the FOM.

Figure 14 Document templates for message definitions

Protocol Definition

- **Game Client** to **Chemical Gas Model**
  - Input
    - » Chemical Detonation Event (Detonation Position)

```
typedef struct
{
    int _x;
    int _y;
}POSITION_2D, DETONATION_POS
```

    - » Virtual Trainee Position

```
typedef struct
{
    int _AgentID
    int _x;
    int _y;
}AGENT_POS
```

    - » Solid Obstacle Coordinates

```
typedef struct
{
    UUID            _type_obstacle
    POSITION_2D _top_left;
    POSITION_2D _bottom_right;
}SOLID_OBSTACLE
```

Protocol Definition

- **Game Client** to **Damage Assessment Model**
  - Input
    - » Virtual Trainee Position

```
typedef struct
{
    int _AgentID
    int _x;
    int _y;
}AGENT_POS
```

- **Damage Assessment Model** to **Game Client**
  - Output
    - » Virtual Trainee Death

```
typedef struct
{
    int _AgentID;
    bool _death;
}AGENT_DEATH
```

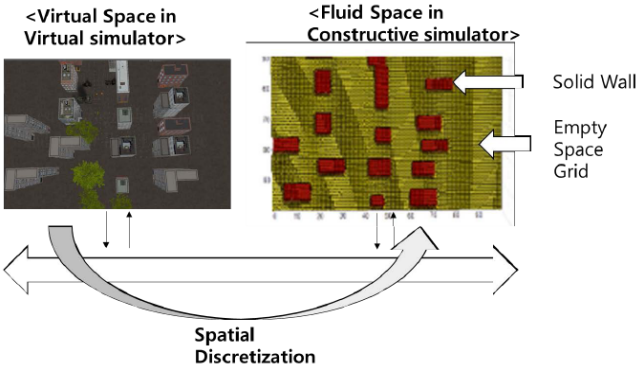
Then, the constructive simulator receives the information and initialises the geographical features of the field. While transferring the geographical information during

interoperation, the SGA discretises the geographical data spatially.

Figure 15 FOM for the serious game federation (see online version for colours)

```
<interactionClass
   name="VW_Env_Info"
   sharing="PublishSubscribe"
   dimensions="saeconds UTCn"
   transportation="HLAbestEffort"
   order="Receive"
   semantics="VW_Env_Info">
  <parameter
     name="Sunshine"
     dataType="HLAASCIIstring"
     semantics="Sunshine"/>
  <parameter
     name="Humid"
     dataType="HLAASCIIstring"
     semantics="Humid"/>
  <parameter
     name="Rainfall"
     dataType="HLAASCIIstring"
     semantics="Rainfall"/>
  <parameter
     name="Season"
     dataType="HLAASCIIstring"
     semantics="Season"/>
</interactionClass>
```

Figure 16 Serious game federation of NBC evacuation training (see online version for colours)



Source: Choi et al. (2013)

6 Lessons learned

After developing serious games for military training using HLA/RTI, we gained several insights. First, depending on the demanded accuracy of the serious game, the constructive simulator can utilise various turbulence CFD models. However, several accurate CFD models cannot guarantee the timing constraints of real-time simulation because of the huge computational time required. Therefore, we had to find appropriate CFD models to satisfy the requirements of a serious game. Moreover, after we found the appropriate CFD models, we had to tune the parameters iteratively until they were appropriate for the CFD model.

Second, in order to develop a federation between the serious game for military training and the constructive simulator without modifying the serious game for military training, the protocol between the server and the client of the serious game for military training should be opened up to the developer. Since we are developing an SGA, which acts as a gateway to the other simulator, the developer should understand the game loop of the serious game. The problem is that commercial games do not offer open game protocols. As a result, we had a difficult time acquiring a serious game in which to develop the federation.

Third, in order to affect the user or the virtual objects during gameplay, a serious game should support administrative features or server-side scripting features. Since the serious game we used was limited for other training contents and we are extending the serious game using HLA/RTI, we could share information easily from the serious game to the constructive simulator. However, if the serious game does not provide the functionality for the user to influence the behaviour or states of virtual objects and other users, then it will be very limited in helping trainees to gain training experience. For example, before we discovered In-world Editor's administrative functionality, we displayed the simulation results in the chat area. Because of its functionality, we chose this virtual world application over several other applications. The virtual world application can make up and arrange virtual training fields easily, and supports server-side scripting, so that we can influence the users and the virtual world objects easily.

7 Conclusions

Developing or extending a serious game for military training can be tedious and difficult work. In order to support developers in extending serious games more easily, we have proposed MSGFDEP to develop a serious game for military training using HLA/RTI. The methodology extends FEDEP, which is the federate development standard for IEEE1516, by utilising the SoSES/FB framework. The main characteristic of the methodology is that, when a game agent and a constructive simulator are provided, a developer can easily synthesise the federation using the SoSES/FB framework. In case a game agent or a constructive simulator does not exist, the methodology provides a means to develop a federation. We expect that the proposed MSGFDEP will assist developers who want to extend existing game applications to serious games, or extend existing constructive simulators to training simulators.

References

Adobbati, R., Marshall, A.N., Scholer, A., Tejada, S., Kaminka, G.A., Schaffer, S. and Sollitto, C. (2001) 'Gamebots: a 3D virtual world test-bed for multi-agent research', in *Proceedings of the Second International Workshop on Infrastructure for Agents, MAS, and Scalable MAS*, Montreal, Canada, Vol. 5.

Bohemia Interactive Simulations (2014) 'Virtual Battle Space 3' [online] http://bisimulations.com/ (accessed 13 August 2014).

Choi, C., Seok, M-G., Choi, S., Kim, T.G. and Kim, S. (2013) 'Serious game development methodology by via interoperation between a constructive simulator and a game application using HLA/RTI', in *International Defense and Homeland Security Simulation Workshop*.

Delta3D (2014) [online] http://www.delta3d.org/ (accessed 12 August 2014).

Exit Games (2014) 'Photon network engine' [online] https://www.exitgames.com/en/Realtime (accessed 12 August 2014).

Fong, G. (2006) 'Adapting COTS games for military experimentation', *Simulation and Gaming*, Vol. 37, No. 4, pp.452–465.

Garro, A., Longo, F. and Nicoletti, L. (2013) 'Disasters management: a serious game architecture centered on a modeling and simulation infrastructure', *SCS M&S Magazine*, Vol. 4, No. 1.

Hudson, K. and Degast-Kennedy, K. (2009) 'Canadian border simulation at Loyalist College', *Journal of Virtual Worlds Research*, Vol. 2, No. 1, pp.3–11.

IEEE (2003) 'IEEE recommended practice for high level architecture (HLA) federation development and execution process (FEDEP)', IEEE Std 1516.3-2003 pp.1–32.

IEEE (2010) 'IEEE standard for modeling and simulation (M&S) high level architecture (HLA) – framework and rules', IEEE Std 1516-2010 (Revision of IEEE Std 1516-2000) pp.1–38.

Kim, B.S., Choi, C.B. and Kim, T.G. (2013) 'Multifaceted modeling and simulation framework for system of systems using HLA/RTI', in *Proceedings of the 16th Communications & Networking Symposium. CNS '13 San Diego, CA, USA: Society for Computer Simulation International*, pp.4:1–4:7.

Kim, T-G., Lee, C., Christensen, E.R. and Zeigler, B.P. (1990) 'System entity structuring and model base management', *IEEE Transactions on Systems, Man and Cybernetics*, Vol. 20, No. 5, pp.1013–1024.

Massei, M., Tremori, A., Poggi, S. and Nicoletti, L. (2013) 'HLA-based real time distributed simulation of a marine port for training purposes', *International Journal of Simulation and Process Modelling*, Vol. 8, No. 1, pp.42–51.

Metello, M.G., Casanova, M.A. and de Carvalho, M.T.M. (2008) 'Using serious game techniques to simulate emergency situations', in *GeoInfo*, pp.121–182.

Möller, B., Löfstrand, B., Lindqvist, J., Backlund, A., Waller, B. and Virding, R. (2005) 'Gaming and HLA 1516 interoperability within the Swedish defense', in *2005 Fall Simulation Interoperability Workshop*.

Park, S.C., Kwon, Y., Seong, K. and Pyun, J. (2010) 'Simulation framework for small scale engagement', *Computers & Industrial Engineering*, Vol. 59, No. 3, pp.463–472.

Ryan, M., Hill, D. and McGrath, D. (2005) 'Simulation interoperability with a commercial game engine', in *European Simulation Interoperability Workshop*, Citeseer, pp.27–30.

Schmidt, D.C. (2006) 'Model-driven engineering', *Computer-IEEE Computer Society*, Vol. 39, No. 2, p.25.

Shanks, G.C. (1997) 'The RPR-FOM. A reference federation object model to promote simulation interoperability', in *1997 Spring Simulation Interoperability Workshop*.

Tang, S. and Hanneghan, M. (2010) 'A model-driven framework to support development of serious games for game-based learning', in *Developments in E-systems Engineering (DESE)*, pp.95–100.

Tang, S. and Hanneghan, M. (2011) 'Game content model: an ontology for documenting serious game design', in *Developments in E-systems Engineering (DeSE)*, IEEE, pp.431–436.

Tang, S., Hanneghan, M. and Carter, C. (2013) 'A platform independent game technology model for model driven serious games development', *Electronic Journal of E-Learning*, Vol. 11, No. 1, pp.61–79.

Tremori, A., Massei, M., Madeo, F. and Reverberi, A. (2013) 'Interoperable simulation for asymmetric threats in maritime scenarios: a case based on virtual simulation and intelligent agents', *International Journal of Simulation and Process Modelling*, Vol. 8, No. 2, pp.160–167.

United States Army Simulation and Training Technology Center (2014) 'Military open simulator enterprise strategy' [online] http://militarymetaverse.org/ (accessed 13 August 2014).

Unity Technologies (2014) 'Unity' [online] http://unity3d.com/unity (accessed 11 August 2014).

US Department of Defense (1998) 'DoD modeling and simulation (M&S) glossary', DoD 5000.59-M, pp.1–175.

Valente, L., Conci, A. and Feijó, B. (2005) 'Real time game loop models for single-player computer games', in *Proceedings of the IV Brazilian Symposium on Computer Games and Digital Entertainment*, pp.89–99.

Zeigler, B.P. (1984) *Multifacetted Modelling and Discrete Event Simulation*, Academic Press Professional, Inc., San Diego, CA, USA.