Defence Science Journal, Vol. 62, No. 6, November 2012, pp. 412-419, DOI: 10.14429/dsj.62.1477

A Distributed Parallel Simulation Environment for Interoperability and Reusability of Models in Military Applications

Taeho Lee, Sangjin Lee, Seogbong Kim*, and Jongmoon Baik#

Agency for Defense Development, Deajeon, 305-152, Korea
#Korea Advanced Institute of Science and Technology, Daejeon 305-701, Korea
**E-mail: sbkim@add.re.kr*

ABSTRACT

Interoperability and reusability of models are main concerns in military simulation. In order to improve the interoperability and reusability of models, the model shall be separated with a particular simulation engine, and the modelling framework of models as well as the architecture of the simulation engine should be standardized. This paper describes the architecture and operational concept of simulation environment which has been developed to enhance interoperability an d reusability of models. We named this environment adaptive distributed parallel Simulation environment for Interoperable and reusable models (AddSIM). We suggested a modelling framework to promote model development, portability and interoperability with other models. Also, we proposed a layered architecture to modularise critical capabilities including kernel layer, tool/application layer, support/service layer and external interface. This means that models can be developed independently of a simulation engine and interfaced with it using API. To validate the application feasibility of AddSIM, we set up an anti-air missile engagement situation and performed simulation. In military simulation, it is expected that reusability and interoperability of models will be enhanced by using proposed AddSIM.

Keywords: Simulation environment, simulation engines, modelling framework, interoperability, reusability

1. INTRODUCTION

Complicated software systems such as software utilized by weapon system should be reused on other projects whenever technically possible. However, this objective is not easy to achieve practically in most cases. To achieve this goal, software should be developed from scratch for the purpose of the software reuse. In addition, the basic principle of interoperability should be followed carefully to other project for the purpose of successful reuse of software.

To ensure the interoperability and reusability among the simulation objects in a distributed environment, high level architecture (HLA) was started in 1996[1]. These simulation objects which we call federates are interoperated in the HLA by way of federation capable of supporting analysis of joint training and so on. In this case, the federation requires the coordination of heterogeneous models. In spite of the success of HLA, itis relatively complex due to the guaranteeing properties such as absence of race conditions and deadlocks[2]. This problem may result in the weakening of real-time processing. However, similar technologies designed to promote interoperability between entities had not been completely standardised when they are potentially running on a multicore computer and their inner components. In addition, the standard of common architecture framework had not been built officially to reduce the development costs needed to build a software model which is integrated in the HLA[3].

Without these standards, it is almost impossible to develop to reusable entities and a model of software components due to their feature of embedding various core framework or simulation engine service within the code for coordinating their activity. Interoperability between entities and components is impossible due to the fact that each simulation engine providing its own event-processing engine bearing with its own unique interfaces.

Authors proposed a layered architecture and modelling framework for supporting software interoperability and reuse of models in military simulation. The proposed architecture and modelling framework have been implemented successfully in AddSIM and illustrated the validity through a pilot study[4].

2. HISTORICAL REVIEW OF SIMULATION ARCHITECTURE

Standard simulation architecture (SSA) was proposed by Steinman and Hardy, and it is recently evolving into open modelling and simulation architecture (OpenMSA) by the parallel and distributed modelling and simulation standing study group (PDMS-SSG) established in Interoperability Standards Organization (SISO) [5].

The evolution of simulation architecture is summarised as follows by Steinman [5].

- SIMulator NETworking (SIMNET) : Supporting real-time battlefield simulations of tanks in a virtual training

Received 7 February 2012, revised 9 October 2012, online published 12 November 2012

environment[6].

- Joint training confederation (JTC) : Integrating models from the different armed forces for supporting joint training exercises
- Time warp operating system (TWOS) : Showing that optimistic time management could achieve parallel speedup when applied to military simulation applications[7].
- Distributed interactive simulation (DIS) : Evolved from SIMNET supporting virtual battles involving (Semi-automated forces (SAF). IEEE standardized more than 100 protocol data units (PDUs) that specify message formats exchanged between DIS models[8].
- Aggregate level simulation protocol (ALSP) : Simplifying the integration of various simulations participating in the JTC, the was developed by MITRE[9].
- Synchronous parallel environment for emulation and discrete event simulation (SPEEDES) : replace of TWOS introducing new flow control techniques that were required to stabilize runtime performance for optimistic simulation[10,11].
- HLA : interoperability standard for building federations out of real time and logical time simulation.
- Standard modelling framework (SMF) and data-stream management system (DSMS) layer: Designed and developed in SPEEDES[12].
- Joint simulation system (JSIMS) : Combination of SPEEDES and HLA. Enabling development of independent models that would interoperate using a powerful SPEEDES-based common component simulation engine (CCSE)
- OpenMSA : Evolved from the combination of SPEEDES-HLA

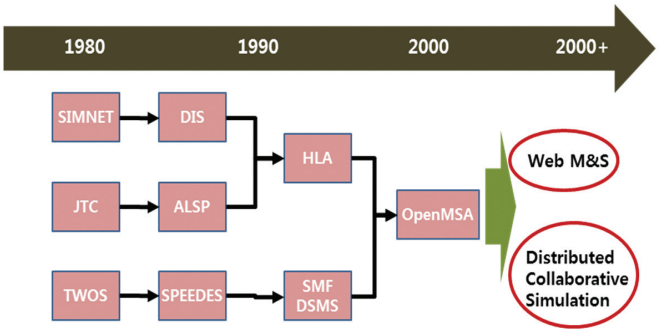


Figure 1. Evolution of standard simulation architecture[5]

Afterwards, simulation architecture in military simulation to support efficiently and efficiently network centric warfare is looking forward to evolve into such as web-based modelling and simulation (MandS), multi-core processor, distributed collaborative simulation, live, virtual, constructive (LVC) integration, etc[13].

Taking into the development trend of simulation architecture, AddSIM was designed and implemented by referencing OpenMSA, which is the standard simulation framework, and reflecting characteristics of web-based MandS and distributed collaborative simulation.

3. AddSIM

3.1 Design Goals

AddSIM is designed to enhance interoperability and reusability of models in the area of acquisition and analysis of military simulation. The detailed design goals of AddSIM are as follows:

- Capability to integrate easily various kinds of simulation in distributed environment
- Architecture of the simulation engine to support plug-in and play of componentised models which have implemented the behavior and functional logic of weapon systems into software models
- Establishing a modelling framework in order to componentise models
- Open architecture for enhancing flexibility and composability of models by separation of model from simulation engine
- Web service based on the service oriented architecture (SOA) concept to utilize and reuse componentised models stored in the local and remote resource repositories
- Providing external Interfaces such as legacy C/C++ code, Matlab, HLA/runtime infrastructure (RTI) to increase the usability of AddSIM
- Providing environmental services such as atmosphere, ocean and terrain for helping the convenience of model developers

3.2 Architecture

AddSIM is designed in the layered architecture for ease of maintenance, prevention against duplication of functions at each layer and convenience in developing models as shown in Fig. 2. Also, it is designed in the form of simulation architecture using shared memory based on middleware to improve the real-time processing capability of simulation. To do this, the Tao-CORBA is used as a middleware and multi passing interface (MPI) concept for distributed parallel processing of simulation is applied[14].

In a tools and application layer, components/players development, build/execution, and analysis of simulation, search and use of componentised models in distributed repositories are performed. The graphical editing framework based-on Eclipse is used as a development tool to increase the convenience and efficiency of making components. To support the reuse of components models, editing tool provides properties of model

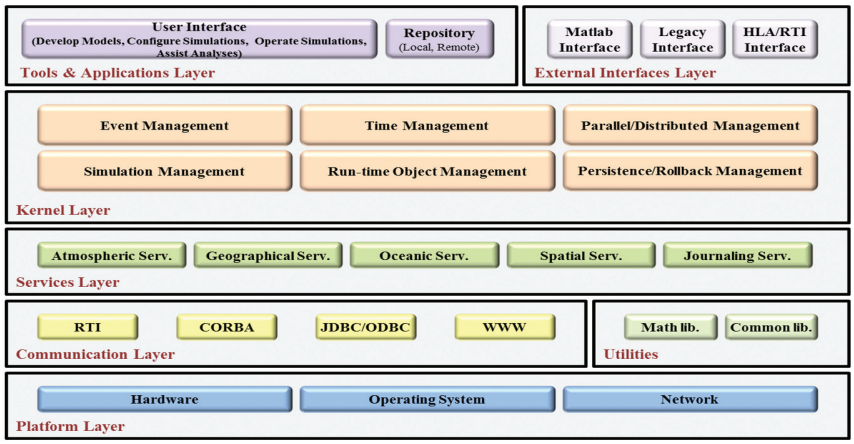


Figure 2. A layered architecture of AddSIM.

as an xml format. The web server for models is linked with the xml file automatically when the model is shared. During the time the component is developed, the xml file (which is used in the simulation configuration and operation) for the model is made. AddSIM also provides the post-analysis module to analyze the simulation result and visualization module using SIMDIS to play back the entire simulation.

Kernel layer as a core layer of AddSIM consists of six functions including parallel/distributed management for parallel processing in distributed environment as well as the basic five functions of event management, time management and simulation management, run-time object management and persistence/rollback management. Procedure executing simulation in kernel layer is as follows. After loading componentised models stored in local and remote repository, simulation object is created. In a kernel layer, componentised models generated in a tools and application layer and stored in local and remote repositories are loaded and run-time objects of simulation are executed. After that, kernel processes simulation events, i.e., initialization or update which is the communication with other runtime objects through messages, stores properties of simulation objects and conducts relay of service for a services layer.

AddSIM provides APIs for modelling the high fidelity models. Users can model easily and faithfully the weapon system by using environmental APIs of atmosphere, ocean, and geography. The atmospheric and oceanic APIs can treat the meteorological data format such as, GRIdded binary (GRIB), SEDRIS transmittal format (STF), network common data file (NetCDF) by transforming data into ASCII files. The geographical API is designed to handle the flat and ellipsoidal earth model. It is used to manage the digital terrain elevation data (DTED) and feature database (FDB) format to extract the geographical feature. User handles the simulation object's spatial information such as position, speed, and user defined data. Journaling API saves and extracts log data generated during the simulation execution and user defined variables. Table 1 shows the sample code of atmospheric API written in C++.

There are many simulation resources developed with C/C++ or Matlab in military simulation. Also, many simulation resources are federated through HLA/RTI. Simulation environment has to support the interoperability with these legacy simulation resources to enhance the usage of simulation environment. For this reason, AddSIM provides three external interfaces such as C/C++, Matlab, and HLA/RTI interface.

Table 1. Sample code of atmospheric API

```
Void pEnvTest :: Get_AtmosphereState3D( )
{
     //  declaration of variables and setting input values
     //  Height (meter), Latitude (degree), Longitude (degree)
   Double dAlt = 1,000.0;  double dLat = 11.42; double dLong = 43.65;
   SAtmosphereState3D stR;
   //  querying three dimensional atmospheric information
   This -> m_ifsAtmosphere -> Get_AtmosphereState3D(dAlt, dLat, dLong, stR);

   //  output of three dimensional atmospheric information
   This -> LogMessage2(Informative, "%f, %f, %f, %f",
                       stR. dAirPressure, // Atmospheric pressure (kg/m3)
                       stR. dAirDensity, // density (N/m2)
                       stR. dSpeedOfSound, // speed of sound (m/s)
                       stR. dTemperature );  // temperature (℃)
```

3.3 Characteristics

AddSIM has some distinguishing characteristics compared with existing simulation environments. The first characteristic is the separation between a simulation engine and models. Modelling framework in AddSIM has been developed upon open simulation architecture for modelling and simulation (OSAMS) (which is being studied as an open modelling framework in PDMS-SSG of SISO) and base object model (BOM), SISO standard for simulation object model[15,16]. A legacy simulation model usually has been developed together with a simulation engine. OSAMS is designed to prevent the deterioration of reusability due to tight coupling of simulation model and engine. As shown in Fig. 3, events of models are handled through a component interface, an API provided by engine.

A simulation model is developed using APIs provided by a simulation engine. As the model and engine are separated, they can be developed independently. As AddSIM adopts this concept, simulation models can be developed as a component independently and are linked with a simulation engine by a plug-in and play way.

The second characteristic is the standardisation of modelling framework. A simulation model is designed to have a hierarchical structure as shown in Fig. 4. The top level is the simulation model which includes several players. Each player consists of several components, and furthermore each component can include sub-component recursively. For example, the simulation model to describe the engagement of surface-to-air missile can possess four players such as a missile, an aircraft, detecting radar, and a launcher for a missile. An aircraft is made up of some components such as propulsion, fire control, weapon, and control system. The weapon of aircraft has several sub-components such as air-to-air missile, air-to-surface missile, etc.

Definition of component, player, and interface is as follows.

- Component: It is a building block (an element of a player or upper component) which executes a specific function independently. The behavior of an element is modeled as a user defined code. A component is compiled into a dynamic link library (DLL) and linked with AddSIM.
- Player: It is a top level component and a part of simulation model. Usually it represents a weapon system such as a tank, an aircraft, a missile. The behavior of a player is also modeled as a user defined code.
- Interface: It is a passage of events through components and players. Using these, components and players can communicate each other.

In the modelling procedure, common meta-model is used to improve interoperability and reusability of model. AddSIM also uses meta-

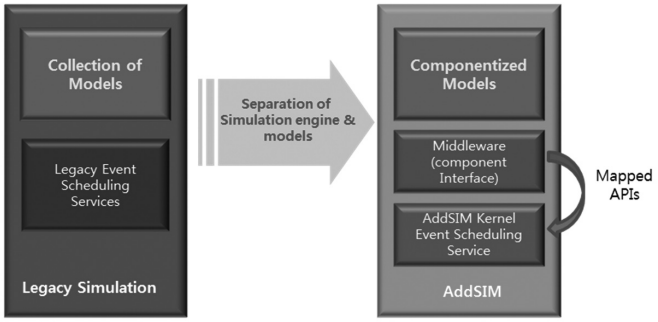


Figure 3. A modelling framework of AddSIM.

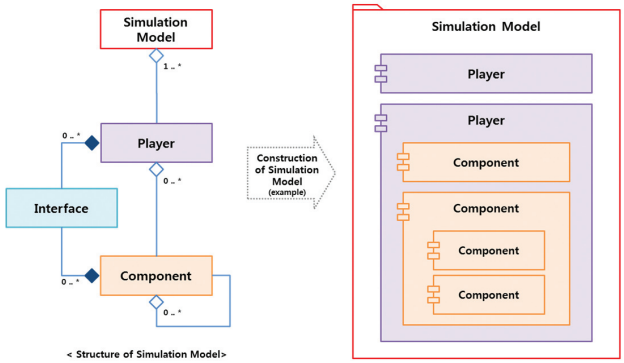


Figure 4. A modelling structure of AddSIM.

model for component/player modelling. As shown in Fig. 5, meta-model in the AddSIM defines the relation among component, player, interface, member function, variable, and data type. Using the hierarchical structure and common meta- model for component/player, AddSIM can enhance interoperability between components/players, and reusability of components/players.

Figure 6 shows the difference among JMASS, OneSAF, and AddSIM. A componentised model in the JMASS is internally loaded. Therefore, a model is recompiled and statically reused with plug-in and rebuild way. OneSAF supports a plug-in and play way using dynamical link library in an entity level. Because jar (java archiver) may contain dynamic link library itself, it has a limitation when a component which is included in a part of other components is reused. AddSIM supports dynamic configuration of executing simulation objects because componentised models participate in the simulation with dynamically linked library way in AddSIM. Due to

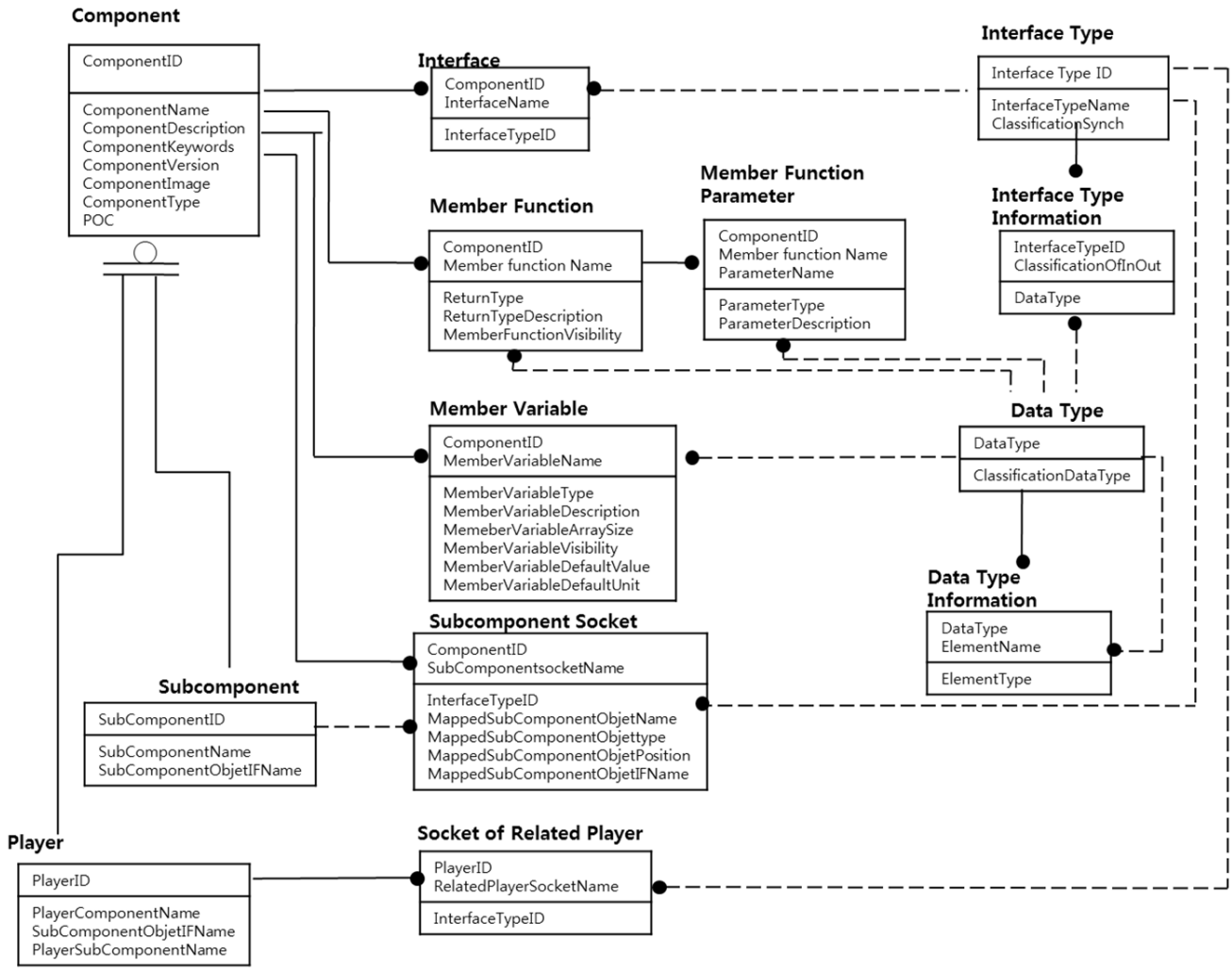


Figure 5. A meta-model in AddSIM.

this, AddSIM supports the reusability, compatibility and interoperability of models.

To configure the dynamic loading for simulation, components and players are compiled by way of componentising. Meta-information for a component such as configuration information, communication information, and control information is stored and controlled in the XML style. While a simulation is executed, a kernel interprets that file for configuring simulation objects. As AddSIM provides dynamical loading of simulation objects, components stored in remote repositories are retrieved or are used without any modification of components by downloading.

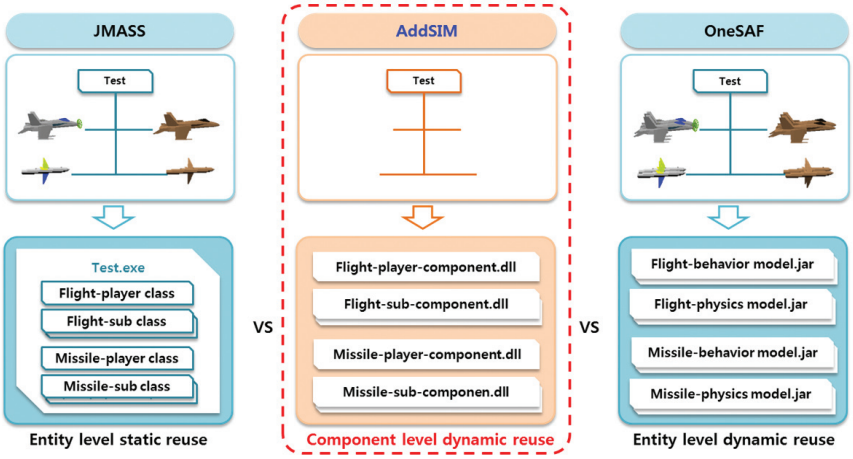


Figure 6. Comparison of reuse methodology among JMASS, AddSIM and OneSAF.

The third characteristic is web service based on SOA concept. To support distributed simulation smoothly, the distributed resource repository based on web is provided. Using the web service, users can retrieve and reuse components stored in a remote repository. Figure 7 shows the operational concept of distributed repository.

There are two types of repository, integrated and local. The integrated repository uses universal description, discovery, and integration (UDDI) engine to support response to user requests such as finding components or simulation configuration. The local repository contains simulation resources such as components, simulation configuration, and simulation scenario. The repository is implemented by using web service and interoperates with a tools/application layer. By interoperation with a tools/application layer, modelers easily can store and share resources. Also, modelers can use remote resources by drag and drop.

Finally, AddSIM engine provides the infrastructure and related functions capable of working number of event processes and synchronizing time between event processes in order to do parallel processing at the same time. Time synchronization algorithm for parallel processing can be divided into conservative way and optimistic way. In the optimistic way, there are time warp, breathing time bucket (BTB), breathing time warp (BTW), etc. Among the optimistic way, AddSIM engine is designed to utilize BTB algorithm and rollback handling for time synchronization between event processes when proceeding parallel processing. In BTB algorithm, each process broadcasts the oldest local even among those it will execute. This is called a local event horizon (LEH). A process must suspend its even processing if it has received an older LEH than the one it is currently processing. The oldest LEH among all processes become the next global event horizon (GEH). Each process may send out all messages and process

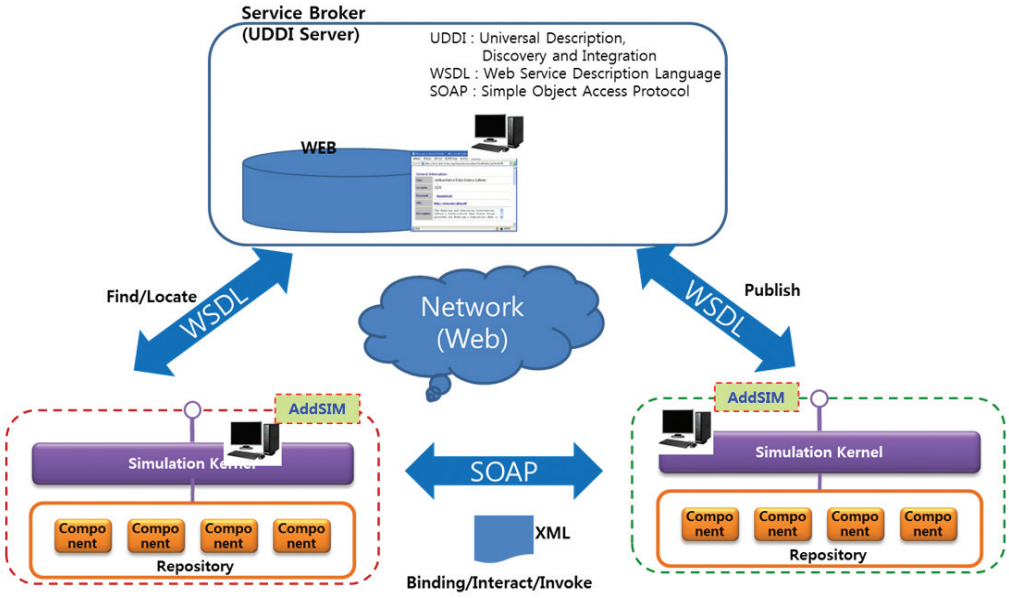


Figure 7. Operational concept of distributed repository.

all events before this new GEH. Processes which have already processed beyond GEH must roll back their computation to GEH. No anti-messages are sent out[17].

AddSIM engine offers the infrastructure and related functions capable of generating runtime objects located in a remote place and passing the interaction messages between runtime objects. As shown in Fig. 8, all constituents of the kernel are operated based on CORBA. Management of runtime object located in remote place is performed by remote kernel, but event management is performed by master kernel through the configuration of the constituent information when kernels are connected.

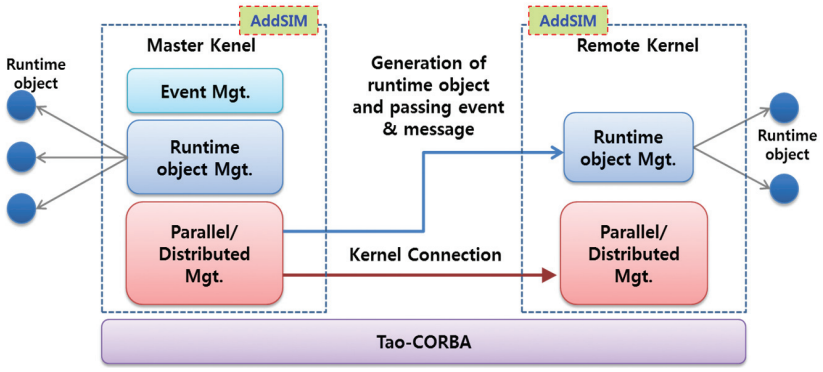


Figure 8. Operation mechanism of distributed processing.

4. OPERATION OF ADDSIM

Author explained operational concept of AddSIM and application example of AddSIM using four players which are described earlier.

4.1 Operational Concept

As shown in Fig. 9, concepts implemented in AddSIM are the separation of simulation engine and model, integrated/ distributed resource repository based on web service adapting SOA concept, simulation architecture based on open simulation architecture, and modelling framework for modelling simulation models. Due to these concepts, AddSIM can support a whole lifecycle of simulation process. To enhance the reusability and interoperability of existing simulation models, AddSIM has the external interfaces such as C/C++, Matlab, and HLA/RTI interface.

4.2 Application of AddSIM

To validate the application feasibility of AddSIM, we have made an anti-air missile engagement situation as shown in Fig. 10. In this case, there are four players (red aircraft, detecting radar, launcher, and missile). These players are newly coded or converted from the legacy models in AddSIM enough to fulfill the framework of the meta model as shown in Fig. 5. The scenario is as follows. A red aircraft approaches a radar station and the radar tries to detect the approaching red aircraft. When the radar detects the red aircraft, the radar sends the detecting signal to a missile launcher. The launcher calculates the estimated aircraft position with detecting information from the radar and homing guide point for a missile. The missile gets the homing guide point and launching signal from the launcher. The missile flies to the homing guide point with the inertial guide algorithm. After it reaches there, it uses seeker to search and track the red aircraft. The ending condition is that the distance between the missile and the red aircraft is within a specified threshold range. Figure 10 shows the relationship among players.

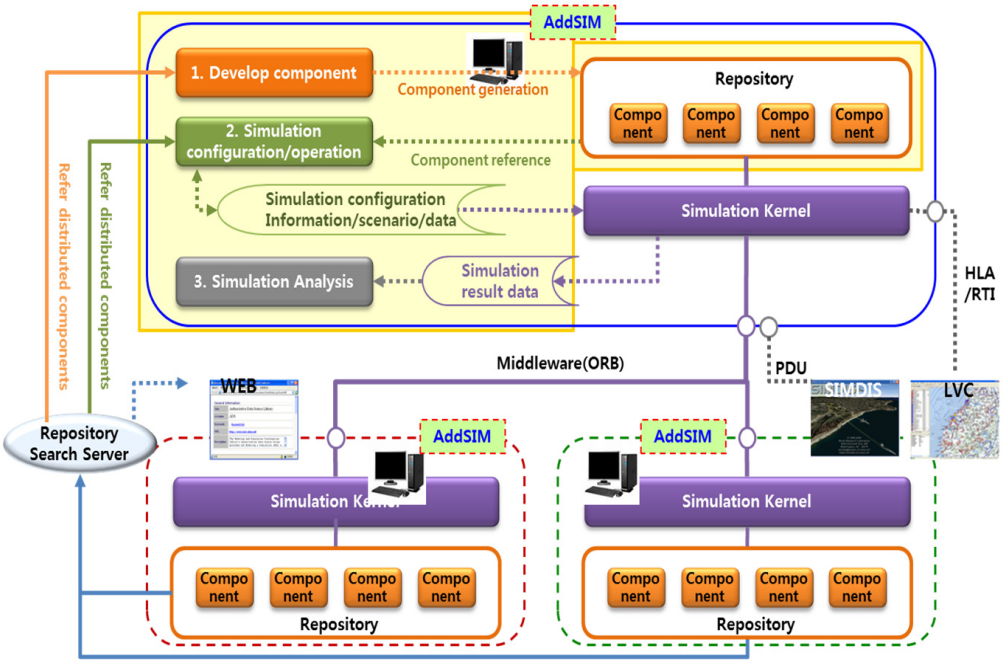


Figure 9. Operational concept of AddSIM (notional).

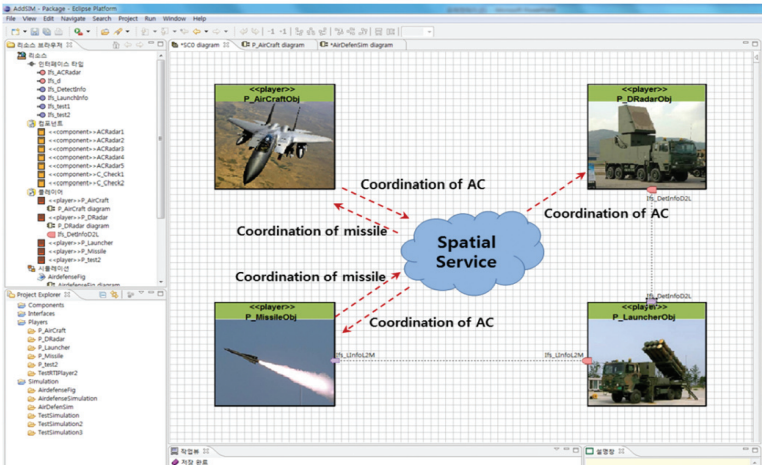


Figure 10. Simulation configuration of a sample example.

All data generated during simulation execution are saved in the journaling DB, the local simulation archive of AddSIM. Simulation results are analysed by using specific data or all data in the DB. The simulation output formats consist of CSV file, analytic report, and visualisation format (SIMDIS format). Figure 11 shows a snap shot of an anti-air missile engagement using SIMDIS. From these activities, AddSIM can be utilized in the Korean military domain.

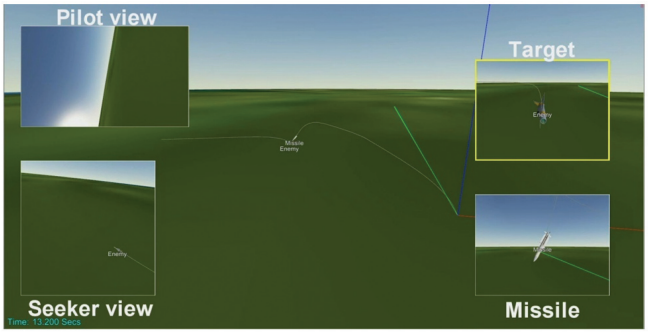


Figure 11. A snap shot of simulation visualization using SIMDIS.

5. CONCLUSIONS

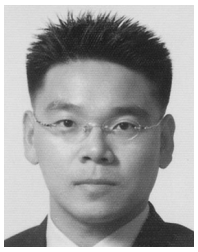
This paper describes the architecture and operational concept of simulation environment called AddSIM which has been developed to enhance interoperability and reusability of models. To this end, models should be separated from a specific simulation engine and the modelling framework of models as well as simulation engine architecture should be standardised. To satisfy these concepts, the architecture of AddSIM, pursuing an open simulation architecture such as OpenMSA, is developed and the modelling framework similar to OSAMS is set up. Based on the modelling framework developed, model developers can make model components systematically. Also AddSIM provides the web service based on the SOA concept to use local resources and share remote resources smoothly. Users easily can use components as plug-in and play way because they are made of dynamic link library. As the dynamic link library is a binary code, the logic, and source code of model can be also protected from leakage. By the result of this research, AddSIM can give the basic simulation environment for the enhancement of reusability and interoperability of models, especially in Korean military application.

REFERENCES

1. Griffin, S.P.; Page, E.H.; Furness, C.Z. & Fisher, M.C. Providing uninterrupted training to the Joint Training Confederation (JTC) during transition to high level architecture (HLA). *In* the Proceedings of the 1997 SimTecT Conference, Canberra, Austrailia, March, 1997, pp.17-20.
2. Dingel, J.; Galan, D. & Damon, C. Bridging the HLA : Problems and solutions. *In* the Proceedings of the 6th IEEE International Workshop on Distributed Simulation and Real-Time Applications, Fort Worth, TX, 2002, 33-42.
3. Institute of Electrical and Electronics Engineers. IEEE standard for modelling and simulation (MandS) high level architecture(HLA)- Framework and Rules 516-2010, 25 March 2010.
4. Oh, H. & Kim, D. Generic simulation models to evaluate integrated simulation environment. *In* the Proceeding of 2011 AsiaSIM, Seoul Korea, 2011.
5. Steinman, J. & Doug, H. Evolution of the standard simulation architecture, Overview of the full SIW paper in Modelling and Simulation. *Soc. Model. Simul. Int.*, 2004, **3**(2), S9-S11.
6. Kanarick, C. A technical overview and history of the SIMNET project. *In* the Proceedings of the 1991 Advances in Parallel and Distributed Simulation conferences, PAnaheim CA. USA, 23-25 January 1991. pp. 104-111.
7. Steinman, J. SPEEDES: Synchronous parallel environment for emulation and discrete event simulation. *In* the Proceeding of the SCS Western Multiconferences on Advances in Parallel and Distributed Simulation, San Diego CA. USA, 17-20 January 1993.pp. 95-103.
8. IEEE, IEEE standard for distributed interactive simulation -Communication services and profiles, IEEE 1278.2, 21 September 1995.
9. Weatherly, R.; Wilson, A. & Griffin, S. ALSP–Theory, experience, and future directions. *In* the Proceeding of the 1993 Winter Simulation Conference, Orlando FL. USA, 1993, pp. 1068-1072.
10. Steinman, J. The WarpIV simulation kernel. *In* the Proceeding of the 2005 Principles of Advanced and Distributed Simulation (PADS) workshop. Monterey CA. USA, 1-3 June, 2005. pp.161-170.
11. Wieland, F.; Hawley, L.; Feinberg, A.; Di Loreto, M.; Blume, L.; Ruffles, J.; Reiher, P.; Beckman, B.; Hontalas, P.; Bellenot, S. and Jefferson, D. The performance of a distributed combat simulation with the time warp operating system. *Concurrency Comput.: Practice Experience*, 1989, **1**(1), 35-50.
12. Steinman, J. SPEEDES: A multiple-synchronization environment for parallel discrete-event simulation. *Int. J. Comp. Simulation*, 1992, **2**, 251-86.
13. Dutta, Debasis. Simulation in military training: Recent development. *Def. Sci. J.*, 1999, **49**(3), 275-85.
14. http://www.cs.wustl.edu/~schmidt/TAO.html
15. Steinman, J.; Lammers, G. & Valinski, M. A Proposed Open

Cognitive Architecture Framework. *In* the Proceeding of 2009 Winter Simulation Conference, Orlando FL. USA, 2009, pp.1345-355.
16. SISO, Base Object Model (BOM) - Template Specification, SISO-STD-003-2006, 31 March 2006.
17. Steinman, J. SPEEDES: A unified approach to parallel simulation. *In* the Proceeding of the 6th Workshop on Parallel and Distributed Simulation, Newport Beach CA. USA, 20-22 January 1992, pp. 75-83.

Contributors



Mr Taeho Lee is currently pursuing his PhD (Software Engineering) from KAIST. He is working as a Senior Researcher at Joint Modelling and Simulation Directorate, Agency for Defense Development (ADD), Republic of Korea. His main research interests include: Distributed and parallel simulation engine, software process improvement, and software reliability.

Dr Sangjin Lee obtained his PhD (Industrial Engineering) from Korea Advanced Institute of Science and Technology, Republic of Korea, in 2008. Presently, he is working as a Senior Researcher at Joint Modelling and Simulation Directorate, ADD, Republic of Korea. His main research interests include: Developing distributed and parallel simulation engine.

Dr Seogbong Kim obtained his PhD (mechanical engineering) from Korea Advanced Institute of Science and Technology, Republic of Korea, in 2009. Presently, he is working as a Principal Researcher at Joint Modelling and Simulation Directorate, ADD, Republic of Korea. His research interests include: Developing distributed, parallel collaborated simulation environment.

Dr Jongmoon Baik received his PhD (computer science) from University of Southern California in 2000. Currently, he is an associate professor in the Computer Science Department at Korea Advanced Institute of Science and Technology. His research activity and interest are focused on software six sigma, software reliability and safety, and software process improvement.