Proceedings of the 2014 Winter Simulation Conference
A. Tolk, S. Y. Diallo, I. O. Ryzhov, L. Yilmaz, S. Buckley, and J. A. Miller, eds.

A DDS-BASED DISTRIBUTED SIMULATION APPROACH FOR ENGINEERING-LEVEL MODELS

Dohyung Kim
Ockhyun Paek
Taeho Lee
Samjoon Park

Modeling and Simulation Division, ADD
Bugyuseong daero 488 beon gil, Yuseong,
Daejeon, 305-152, REPUBLIC OF KOREA

Hyunshik Bae

M&S Technology Institute, SIMNET Co., Ltd.
892-9 Jijok-dong, Yuseong,
Daejeon, 305-330, REPUBLIC OF KOREA

ABSTRACT

AddSIM is a component-based simulation environment that has been developed for weapon system modeling and engagement simulation. While AddSIM addresses component-based model development and reuse issues, it does not fully consider distributed simulation for massive and frequently changing data. We studied some approaches to develop an engagement simulation environment based on data distribution service for distributed systems (DDS). This article introduces three different approaches to apply DDS to AddSIM and describes their advantages and disadvantages. Then, we choose the best way to develop the mixture of AddSIM and DDS. According to the proposed approach, we explain the data exchange mechanism between AddSIM nodes and time synchronization for correct execution. We also define several DDS topic types for interoperation. Finally, we describe an anti-ship warfare case study to show the difference between AddSIM and the proposed approach.

1 INTRODUCTION

In the military and defense domain, models and simulations are often classified into four levels according to the abstraction of the target systems such as engineering, engagement, mission, and campaign levels (Lalit, Joseph, and Richard 1994; DoD M&S Master Plan 1995). In particular, high-fidelity engineering models have been widely used to improve efficiency and effectiveness in engineering development and system design. Engagement simulation is also important for system developers because it is used to derive functional/performance specifications from requirements and to predict and evaluate system effectiveness. If we directly build an engagement simulation using engineering-level models, it will provide reliable simulation results and high reusability of engineering-level models. We can avoid problems due to the different abstraction level between the models.

Some studies have focused on using simulation middleware for interoperation of engineering models to directly build a high-fidelity engagement simulation using engineering-level models (Sung, Hong, and Kim 2009; Hong et al. 2011). In those studies, distributed engineering-level models and discrete event models representing such things as command and control (C2) join the federation for the upper level simulation.

Other studies have focused on using a component-based simulation infrastructure such as AddSIM. AddSIM is a component-based simulation environment that was developed for weapon system modeling and engagement simulation (Lee et al. 2012; Kim, Oh, and Hwang 2013). Simulation objects in AddSIM are developed as components and composed in a hierarchical manner. While AddSIM addresses component-based model development and reuse issues, it does not fully consider distributed simulation for

massive and frequently changing data. AddSIM provides external interfaces for interoperation, but a new way is needed to run a distributed simulation via the AddSIM kernel for user convenience and maintainability.

OMG data distribution service (DDS) middleware provides high performance with low latency and various qualities of service capabilities (OMG 2007). We think the combination of AddSIM and DDS is the best way to develop a distributed simulation between distributed AddSIM nodes.

This article introduces three different approaches to apply DDS to AddSIM and describes their advantages and disadvantages. Then, we choose the best way to develop the mixture of AddSIM and DDS. According to the proposed approach, we explain the data exchange mechanism between AddSIM nodes and time synchronization for correct execution. We also define several DDS topic types for interoperation. Finally, we describe a case study of anti-ship warfare to show the difference between AddSIM and the proposed approach.

2 ADDSIM AND DDS

2.1 AddSIM

AddSIM is the common simulation environment that uses high resolution engineering models of weapon systems to enhance interoperability, reusability, and composability of simulation models. AddSIM helps the engineering or engagement-level simulation to analyze the measure of performance (MOP) or measure the effectiveness (MOE) of weapon systems. The first version of AddSIM was developed in 2011 and the second version in 2014 by the Agency for Defense Development (ADD) in Korea. To enhance flexibility, maintainability, and scalability, AddSIM was designed in a layered architecture, as shown in Figure 1. The architecture and characteristics of AddSIM are described previously in detail (Lee et al. 2012; Kim, Oh, and Hwang 2013).

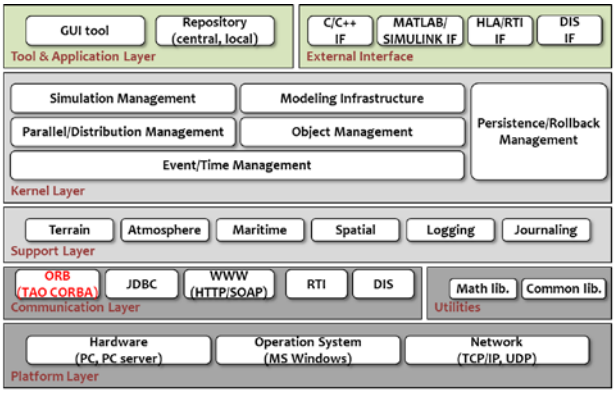


Figure 1: AddSIM logical architecture.

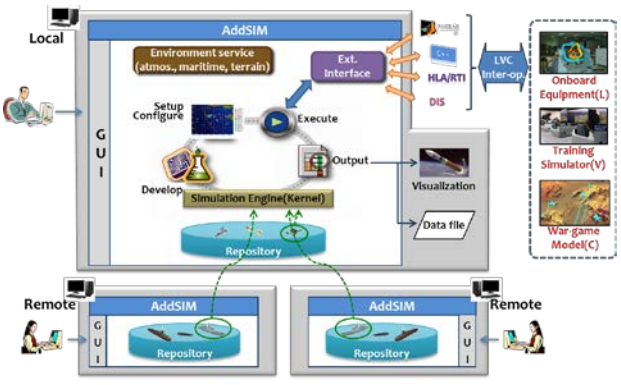


Figure 2: AddSIM operational concept.

Figure 2 shows the AddSIM operational concept. Before beginning simulation, user sets the objective and conceptual scenario for the simulation. Then, user collects the models (i.e., components) required for simulation by utilizing the GUI tool in the tool & application layer. At this time, the user can develop new models or collect legacy models by web-based searching. Legacy models, which have already been developed for another purpose, can be used in the simulation by linking at the remote computer's repository or by using AddSIM external interface modules. A top-level component is called a player and it usually represents a single weapon system. Then, user setups the scenario and execute the simulation. Simulation results can be exported as a CSV data file or visualized using SIMDIS or Vega-Prime for analysis.

AddSIM is designed to perform the simulation using distributed objects rather than executing a distributed simulation in which more than one AddSIM node runs in a parallel manner. If a simulation scenario can be set using models located in remote computers, AddSIM uses common object request broker

architecture (CORBA) as distributed computing middleware. When performing the distributed simulation, the AddSIM kernel invokes a request at the kernel of a remote computer through CORBA middleware and then the remote kernel reply to the kernel of the local computer occurs synchronously. This client-server communication method is simple and reliable if information is naturally centralized, but it has problems of high latency and low scalability when used for large scale real-time simulation. To overcome this problem, AddSIM was designed to use TAO-CORBA (the ACE ORB), which is known as distributed object middleware, and is efficient for real-time simulation (DOC 2012).

The previous version of AddSIM using TAO-CORBA still had performance problems such as high latency and low scalability when used for performance-sensitive applications, particularly when requiring massive data transactions or a high frequency (lower delta t). These massive data transaction cases are the main phenomena of engineering or engagement-level simulation, which is the target area of AddSIM simulation. Therefore, a solution to overcome this problem is required.

2.2 Data Distribution Service

There are four ways to communicate using middleware such as Point-to-Point, Client-Server, Publish-Subscribe Messaging and Replicated Data. The Client-Server way (RPC, CORBA et al.) is satisfactory if information is naturally centralized but poor at single point failure and performance bottlenecks. Publish-Subscribe Messaging is good at many-to-many communication or distributing time-critical information.

DDS is a functional specification adopted from the Object Management Group (OMG) to facilitate efficient distribution of data in a distributed system requiring data-centric publish-subscribe communications (OMG DDS portal).

DDS architecture consists of Real-Time Publish-Subscribe (RTPS), Data-Centric Publish-Subscribe (DCPS), and Data-Local Reconstruction Layer (DLRL) level, as shown in Figure 3. RTPS provides the interoperability protocol of allowing multi-vendor DDS implementations to communicate. DCPS is targeted towards efficient delivery of the proper information to the proper recipients. In this layer, API applications can be used to exchange topic data with other DDS-enabled applications according to designated Quality of Service (QoS) policies. DLRL is an optional layer that allows for a simpler integration into the application layer.

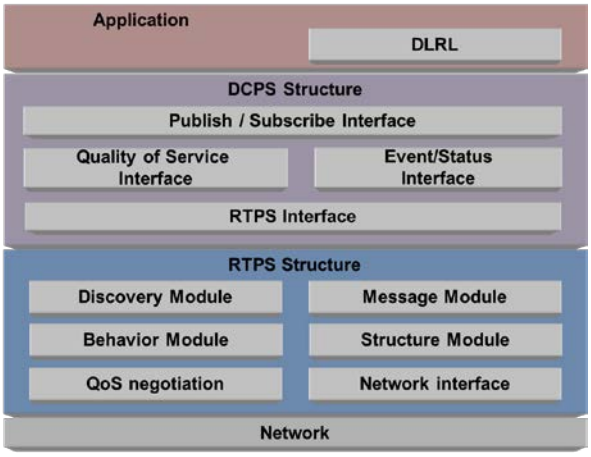


Figure 3: Architecture of DDS.

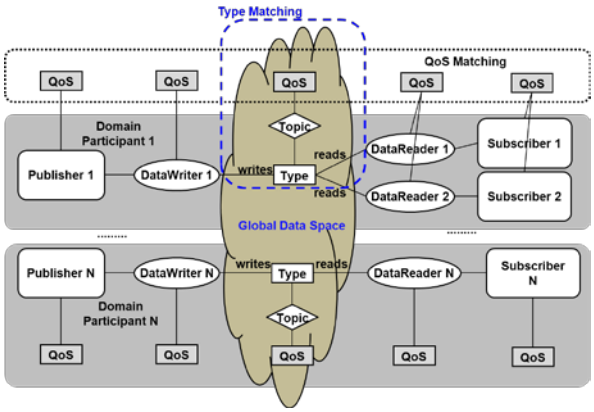


Figure 4: Entities of DCPS.

DCPS entities include topic, publisher, data writer, subscriber, data reader, and domain participants (Figure 4). Publisher produces information in the global data space using topic, and subscriber consumes the information from the global data space. Information can be transferred after matching the type and QoS.

QoS policies can be associated with all relevant entities. At this time, publications and subscriptions can match only if the declared and requested QoS is compatible.

DDS supports continuous data distribution among distributed applications and provides flexibility, scalability, and modular structure in a decoupling way. Additionally, DDS automatically handles all aspects of message delivery, and users can specify the enumerated list of QoS. DDS has automatic discovery of participants and no single point of failure. Because of these characteristics, DDS is suitable for distributed simulation required for reliable real-time data communication, and it is widely used in the aerospace and defense domains.

However, because DDS was originally designed for real-time data communication, it has some limitations to be used directly in the distributed simulation area. DDS has no APIs for specifying federation save/restore and synchronization points. It does not provide standardized APIs specifically for domain save/restore, or for defining domain-wide synchronization points. Additionally, DDS has no standardized APIs specifically for time management. However, these limitations can be overcome by using the standardized DDS APIs and QoS policies, which are equivalent to High Level Architecture (HLA)-like federation or time management service (Joshi and Castellote 2006).

Previous research on applying DDS to the simulation was performed by NADS (Lopez and Martin 2011). They focused on HLA architecture migrating to new architecture by fusing DDS middleware. They used the DDS standard as default for messaging, while the middleware object model was based on HLA metadata. The SISO-LSA study group also began to study a similar research direction (SISO-LSA Study Group webpage). However, that research is at the initial stage of forming concepts, and notable results have not been announced. DDS was not originally designed for use in the simulation area. Thus, it is not easy to directly apply DDS to simulations. However, if we selectively take advantages of DDS as in previous research, the distributed simulation capability of AddSIM can be enhanced.

3 MIXTURE OF ADDSIM AND DDS: DEVELOPMENT STRATEGY

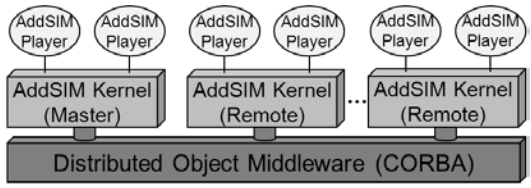
In this section, we introduce three different approaches to apply DDS to AddSIM and discuss their advantages and disadvantages. We simply call the mixed system "AddSIM-DDS". When players are simulated on distributed nodes according to the proposed approach, a mechanism is required for data exchange between AddSIM nodes and time synchronization for correct execution. Thus, we define several DDS topic types.

3.1 Three Approaches to AddSIM-DDS Development

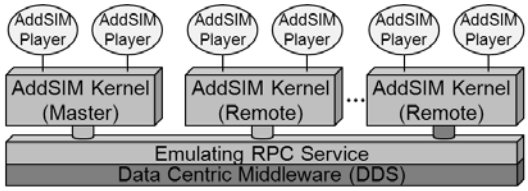
DDS can be applied to AddSIM in different ways depending on the demands for distributed simulation and ease of development. Figure 5 (a) shows the AddSIM simulation structure using distributed objects, and Figure 5 (b–d) shows three possible ways to use DDS middleware with distributed AddSIM nodes, respectively. As shown in Figure 5 (a), AddSIM uses TAO as distributed computing middleware to simulate between distributed AddSIM player objects. In this case, distributed objects communicate synchronously using TAO remote procedure calls (RPC). Time and event management for synchronization of all AddSIM nodes are achieved in a single AddSIM, called a master. All other AddSIM nodes communicate with the master AddSIM using client-server communication to advance their logical time. Complete synchronization between AddSIM nodes is achieved by using simple sequential algorithm, but as described in section 2.1, the AddSIM kernel does not support distributed simulation in which more than one AddSIM node runs in a parallel manner.

By implementing RPC services over the DDS in application type I, AddSIM nodes can communicate with each other in the same way as AddSIM (Figure 5 (b)). In this way, AddSIM-DDS is achieved without serious modification of the current AddSIM architecture. This also provides QoS support for an RPC interaction using rich QoS policies of DDS. Several efforts have been made to emulate RPC using DDS, and the latest standardization effort focus on adding RPC over DDS (Losa 2013; Kümmel, Hutschenreuther, and Schill 1997). However, this approach does not match the original purpose of DDS, data-centric loosely

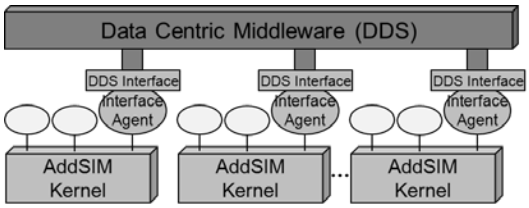
coupled communication, and it does not provide any distributed simulation capabilities via the AddSIM kernel.



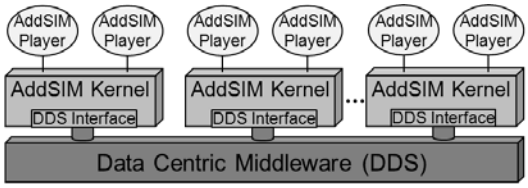
(a) AddSIM: simulation with distributed objects



(b) Type I: implement RPC over DDS



(c) Type II: use DDS as an external interface



(d) Type III: use DDS via AddSIM kernel

Figure 5: Simulation structure of AddSIM and three types of approaches to AddSIM-DDS.

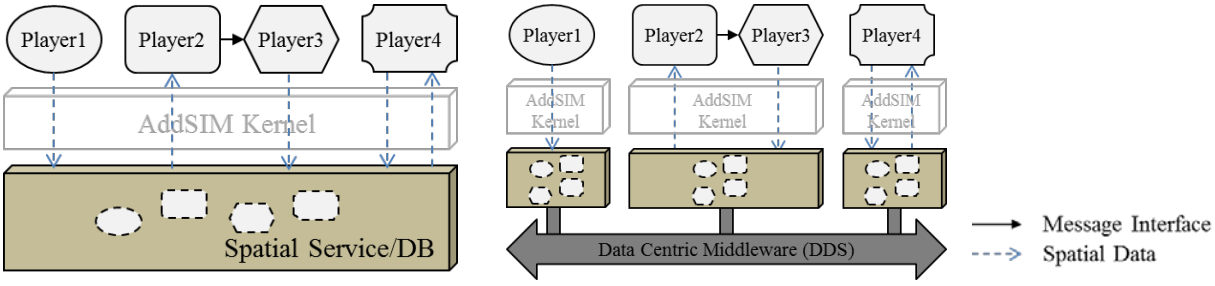
AddSIM does not support distributed simulation via a simulation kernel. Instead, it provides four different types of external interfaces such as the C/C++ interface, a MATLAB interface, the HLA/RTI interface, and the DIS interface. The HLA/RTI interface agent has a role of a gateway through which AddSIM players can interact with other objects in the federation. Similarly, DDS can be used as one of the AddSIM external interfaces as shown in Figure 5 (c). In this way, the AddSIM kernel is unaware of federation, and an interface agent for DDS performs all interactions with DDS. Such a system is easy to develop, and provides high flexibility and extensibility through modifications of the interface agent and DDS interface libraries. However, this approach can be difficult to developers of the interface agent because all the work of interacting with DDS falls to them.

In a type III application, distributed AddSIM nodes can communicate with each other using DDS under the AddSIM kernel (Figure 5 (d)). In this way, the AddSIM kernel becomes more complicated and standardization of exchanging DDS topics is required, but it provides easy-to-use environments to developers. It also provides good maintainability. We decided to adopt this approach to develop distributed simulations between AddSIM nodes via their kernels.

3.2 Data Exchange and Time Synchronization in AddSIM-DDS

When players are simulated on distributed nodes, a mechanism is needed for data exchange between AddSIM nodes and time synchronization for correct execution of the entire simulation. There are two different ways in AddSIM to exchange data between players. One is exchanging data through the AddSIM spatial service and the other is to use the directly connected message interface between players.

AddSIM provides a spatial service for exchanging spatial location information for participating players. Every player updates their new spatial location information into the AddSIM spatial database. Then, the other players inquire about the information required to calculate their behavior logic. This method uses asynchronous communication, so it can be changed smoothly into the DDS publish-subscribe model.



(a) (b)

Figure 6: Data exchange in AddSIM and AddSIM-DDS.

Figure 6 (a) shows the data exchange concept using the spatial service in AddSIM. We propose the spatial service architecture for distributed AddSIM nodes, as shown in Figure 6 (b). Every distributed AddSIM node has its own spatial database. Each database maintains the same contents by distributing updated data through DDS middleware.

In our proposed approach, AddSIM uses DDS middleware via its kernel. DDS is based on a topic based publish-subscribe paradigm. Therefore, distributed AddSIM nodes must use topic type representing spatial information to synchronize their spatial databases. The spatial database in AddSIM holds information such as identifiers and radar cross section (RCS) value of a player and position, linear velocity, angular velocity of a player over time. Table 1 depicts the data structure of spatial information in AddSIM. According to this data structure, we could define the topic type for spatial information- **`SpatialInfoType`**. Considering interoperation with HLA, it is mapped to **`BaseEntity. PhysicalEntity`** object in the Real-time Platform Reference Federation Object Model (RPR FOM) standards, as shown in Table 2 (a).

Table 1: Data structure for spatial information.

```
typedef struct _SSpatialData {
    CStringValue    szID;
    CStringValue    szParentID;
    STime           stTime;
    double          dblPosition[3]
    double          dblRotational_velocity[3];
    double          dblLinear_velocity[3];
    double          dblRotational_acceleration[3];
    double          dLinear_acceleration[3];
    double          dOrientation[3];
    bool            bDestroyed;
    EIFFType        eIFF;
    double          dRCS;
    double          dExtAttribute[10];
} SSpatialData;
```

Table 2: Physical entity object and munition detonation interaction of RPR FOM.

Object	Attribute	Datatype	UpdateType	Update Condition
BaseEntity. PhysicalEntity	AccelerationVector	AccelerationVectorStruct	Conditional	AccelerationChange
	AngularVelocity	AngularVelocityVectorStruct	Conditional	AngVelocityChange
	EntityType	EntityTypeStruct	Static	N/A
	EntityIdentifier	EntityIdentifierStruct	Static	N/A
	Orientation	OrientationStruct	Conditional	OrientationChange
	WorldLocation	WorldLocationStruct	Conditional	LocationChange
	VelocityVector	VelocityVectorStruct	Conditional	VelocityChange
	ForceIdentifier	ForceIdentifier		

(a)

Interaction	Attribute	Datatype
MunitionDetonation	DetonationLocation	WorldLocationStruct
	FiringObjectIdentifier	RTIObjectIdStruct
	FinalVelocityVector	VelocityVectorStruct
	FuseType	FuseTypeEnum16
	MunitionType	EntityTypeStruct
	QuantityFired	unsigned short
	TargetObjectIdentifier	RTIObjectIdStruct
	WarheadType	WarheadTypeEnum16

(b)

Players communicating with direct interfaces should be in the same node because they are tightly coupled during the simulation. However, an assessment player who assesses damage caused by munitions might be distributed if necessary. Exchanging information by assessment players to assess the damage could be defined as a standardized topic type mapped to `MunitionDetonation` interaction in the RPR FOM, as shown in Table 2 (b).

The distributed AddSIM nodes must be synchronized using a conservative or an optimistic synchronization protocol to correctly execute the entire simulation. We adopted a conservative protocol in which each AddSIM node must execute its model logic in a time stamp order. We defined the topic type for the time stamp information representing a specific point in logical simulation time, and we developed time management service over DDS. Figure 8 illustrates the general behavior flow of the AddSIM-DDS node during its life. The AddSIM-DDS kernel includes six categories for distributed simulation service including spatial service management, time management, DDS services domain module, topic-definition module, publication module, and subscription module. User-developed players use four types of DDS service (green-shaded blocks) via the AddSIM kernel, so the interoperation workload is minimized.

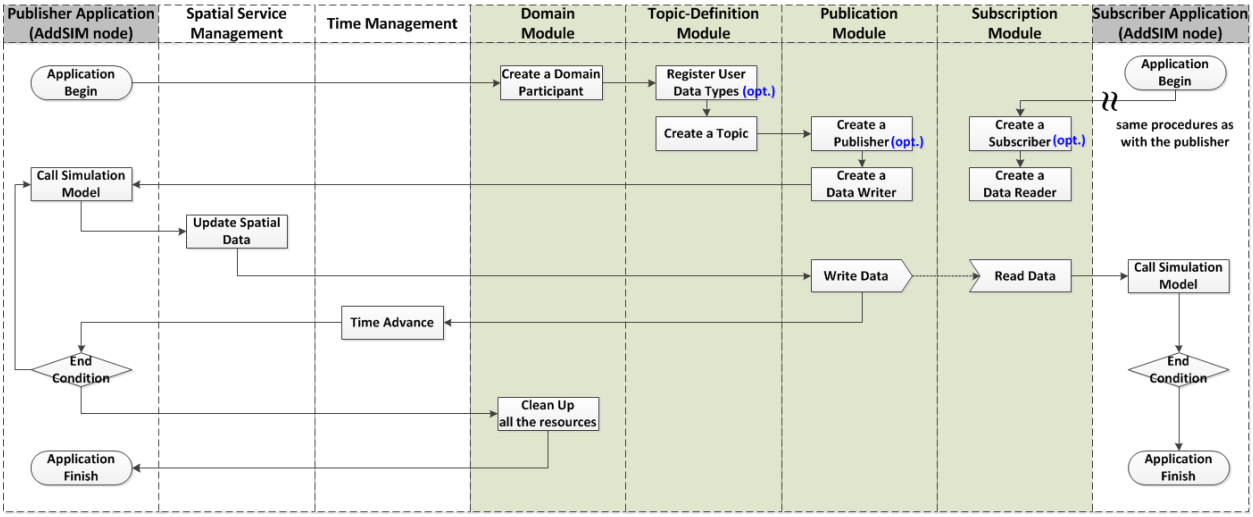


Figure 7: The General Behavior Flow of the ADDSIM-DDS Node.

4 CASE STUDY: ANTI-SURFACE SHIP WARFARE

In this section, we describe a case study which is a distributed engagement simulation based on engineering-level models. Through this case study, we explain how AddSIM-DDS is different from AddSIM when executing a distributed simulation. The case study scenario will be utilized for the AddSIM-DDS test-bed. We describe a brief scenario of the case study, participants, and their roles and a comparison of AddSIM and AddSIM-DDS when simulating the scenario on distributed nodes.

4.1 Anti-Surface Ship Warfare Scenario

The objective of anti-ship warfare simulation is to derive functional and performance specifications from required operational capability (ROC), and to predict and evaluate the effectiveness of the defense system. A brief scenario for the anti-surface ship warfare simulation is described in Figure 8. The simulation process was comprised of the following six steps.

0 Initial status: The patrol airplane moves over the sea for surveillance. The blue and red warships move from their initial positions.

1 When the target warship (red) arrives in the defense area of the blue force, the patrol airplane detects the target using a searching algorithm.
2 The patrol airplane tracks the target and sends the target information to the warship (blue)
3 The warship (blue) commands to launch the anti-ship missile (ASM) after receiving the target information.
4 The ASM flies to the pre-programmed way-point through its own dynamics by using inertial navigation guidance. Then, the missile performs homing guidance with a seeker to the target.
5-1 The warship (red) detects the ASM and analyzes it.
5-2 The warship (red) launches the anti-air missile (AAM) to defend against the ASM.
6 If the AAM fails to defend, the warship (red) commands the close-in weapon system (CIWS) to fire.

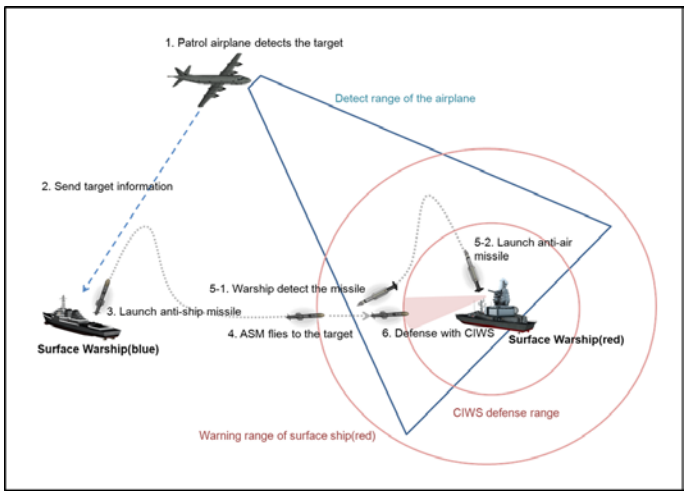


Figure 8: A brief scenario of anti-ship warfare.

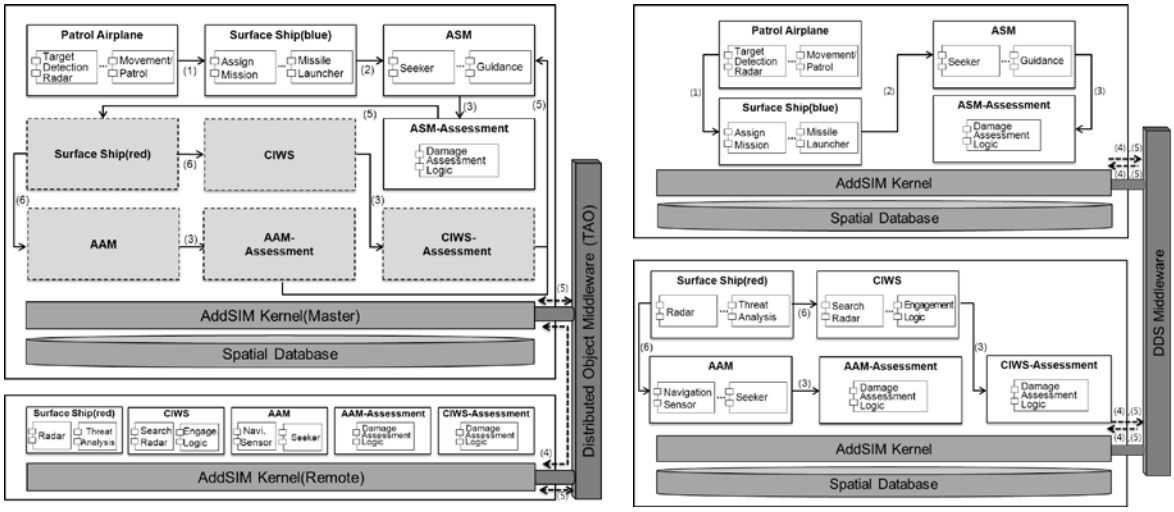
Table 3: Participants and their roles.

Participant/Application	AddSIM Player	Roles
Anti-ship application	Patrol Airplane	Sea surveillance Target identification/Report
	Surface ship (blue)	Receive reports Threat Analysis Assign mission /Launch missile
	Anti-Ship Missile (ASM)	Attack surface ship (red)
	ASM-Assessment	Damage assessment of engagement between ASM and Surface ship (red)
Air defense application	Surface ship (red)	Detect target, Threat Analysis Launch AAM, CIWS
	Anti-Air Missile (AAM)	Defend ship against ASM (long range)
	Close in Weapon System (CIWS)	Defend ship against ASM (close range)
	AAM-Assessment	Damage assessment of engagement between AAM and ASM
	CIWS-Assessment	Damage assessment of engagement between CIWS and ASM

We assumed that a simulation is a participant in the DDS domain. In a simulation, there are several players who have their own logic and behavior. The roles of participants and players are shown in Table 3.

4.2 The Comparison of Simulation on Distributed Nodes in AddSIM and AddSIM-DDS

According to the our case study scenario, we can compose two types of simulation such as a distributed object simulation in AddSIM and a distributed simulation in AddSIM-DDS, as shown in Figure 9 (a, b). The blocks represent the AddSIM players who participate in the simulation. Players may have their sub-components to perform their roles and they need frequent data exchange. For example, the radar component of the warship (red) detects a target inside its detection range by obtaining spatial information of the target, which changes in milliseconds. The data elements exchanged among players are target information (1), fire command of warship (blue) (2), damage assessment request (3), player's spatial information (4), damage assessment results (5) and fire command of the warship (red) (6) (see numbers in Figure 9 (a, b)). Some of the data elements can be directly connected between players using the interface. For example, the target information interface, which is sent from the patrol airplane to the warship (blue), can be defined as shown in Table 4.



(a) AddSIM (b) AddSIM-DDS

Figure 9: Simulation on distributed node.

Table 4: Target information interface.

Attribute name	Data type	Description
Detect_time	Double	Detected time
Target_ID	Double	ID of target player
Target_type	Double	Type of target player
Target_pos(3)	Double	Altitude, longitude, height
Target_vel(3)	Double	Ve, Vn, Vu
Target_attitude(3)	Double	psi, theta, phi

Figure 9 (a) shows the distributed object simulation in AddSIM that uses TAO-CORBA for distributed object management. Dark shaded players are remote object references. To compose a distributed object simulation, the user makes references to the players from the distributed node, e.g. surface warship (red),

CIWS, and AAM players. During running of the simulation, the master kernel notices that the simulation refers to a player object located at the distributed node, and then the master kernel requests at a remote kernel through the middleware and then the remote kernel replies to the master kernel synchronously. This method may have a performance problem when the simulation requires frequent data exchange. Figure 9 (b) shows the distributed simulation in AddSIM-DDS. The dotted arrow indicates data exchange between distributed nodes. Each simulation can run in a parallel manner on distributed AddSIM kernels. Data elements such as (4) and (5) are defined as standardized topic type such as `SpatialInfoType` and `MunitionDetonation`. Both AddSIM and AddSIM-DDS have directly connected interfaces between players on the same node and also have data interactions between distributed nodes.

5 CONCLUSION

DDS provides a powerful communication infrastructure with its real time performance, high rate messaging, and various QoS capabilities. Applying these DDS capabilities to AddSIM will improve the distributed simulation capability of AddSIM. Among three different approaches for applying DDS to AddSIM, we choose a method that distributed AddSIM players use DDS via it's kernel for interoperation. This approach provides easy-to-use environments to developers and good maintainability because user-developed players are not connected directly to the DDS middleware. According to the proposed approach, topic types must be defined to exchange spatial information, assessment information, and time stamp information. Distributed AddSIM nodes can synchronize their spatial databases using the topic type representing spatial information, and can exchange assessment and time stamp information with each other.

The proposed approach was applied to an anti-surface ship warfare simulation which is made up of multiple engineering-level models to examine the differences between AddSIM and AddSIM-DDS. Furthermore, it will be used to validate the functionality and usability of AddSIM-DDS.

The development of AddSIM-DDS is currently underway and requires further efforts. All components and structure of AddSIM, including object management for distributed players, should be redesigned according to the new requirements. Although our approach provides a simple and effective way for mixing AddSIM and DDS, it might restrict the communication type between distributed AddSIM nodes due to predefined topic types. The ideas presented here do not cover all topics about AddSIM-DDS, but hopefully they can be extended with active research over the next few years.

REFERENCES

Corsaro, A. 2009. "The DDS Tutorial – Part I, II." PrismTech, Inc. http://www.omgwiki.org/dds/sites/default/files/Tutorial-Part.I.pdf

DOC (Distributed Object Computing) Group. 2012. TAO (The ACE ORB). http://www.cs.wustl.edu/~schmidt/TAO.html.

DoD (Department of Defense). 1995. *DoD Modeling and Simulation (M&S) Master Plan,* DoD 5000.59-P, Under Secretary of Defense for Acquisition and Technology.

Fujimoto, R. M. 1998. "Time Management in the High Level Architecture." *SIMULATION Special Issue on High Level Architecture*, 71(6): 388-400.

Hong, J. H., K. M. Seo, M. G. Seok, and T. G. Kim. 2011. "Interoperation between Engagement-and Engineering-level Models for Effectiveness Analyses" *The Journal of Defense Modeling and Simulation: Applications, Methodology, Technology*, 8(3): 143-155.

Joshi, R., and G.P. Castellote. 2006. "A Comparison and Mapping of Data Distribution Service and High-Level Architecture." Real-Time Innovations, Inc. http://www.rti.com

Kim, D., H. Oh, and S. Hwang. 2013. "Integrating Legacy Simulation Models Into Component-Based Weapon System Simulation Environment." *Proceedings of the 2013 Summer Computer Simulation Conference*, 251-256.

Kümmel, S., T. Hutschenreuther, and A. Schill. 1997. "QoS Support for Advanced RPC Interactions." *Proceedings of the IEEE Conference on Protocols for Multimedia Systems–Multimedia Networking*.

Lalit K., G. Joseph, and O. Richard. 1994. "Systems acquisition manager's guide for the use of models and simulations." Report of the DSMC, Defense Systems Management College Press.

Lee, T., S. Lee, S. Kim, and J. Baik. 2012. "A Distributed Parallel Simulation Environment for Interoperability and Reusability of Models in Military Applications." *Defense Science Journal*, 62(6): 412-419.

Lopez-Rodriguez, J.M., and R. Martin. 2011."How to Develop True Distributed Real Time Simulations? Mixing IEEE HLA and OMG DDS Standards." Nextel Aerospace Defense & Security(NADS), http://simware.es.

Losa, J. 2013. eProsima RPC over DDS, OMG Technical Meeting, Berlin, http://www.eprosima.com.

Oh, H. S., and D. Kim. 2011. "Generic simulation models to Evaluate Integrated Simulation Environment." *Proceedings of the 2011 Asia Simulation Conference.*

OMG Data distribution Service Portal, http://portals.omg.org/dds/.

OMG(Object Management Group). 2007. Data Distribution Service for Real-time Systems Specification Version 1.2. http://www.omg.org/spec/DDS/1.2/PDF/formal-07-01-01.pdf.

Schmidt, D.C., D.L. Levine, and S. Mungee. 1998. "The design of TAO real-time object request broker", *Computer Communications*, 21(4): 294-324.

SISO-LSA Study Group, http://www.sisostds.org/standardActivities/StudyGroups/LayeredSimulation ArchitectureLSASG.aspx/.

Sung, C. H., J. H. Hong, and T. G. Kim. 2009. "Interoperation of DEVS Models and Differential Equation Models using HLA/RTI: Hybrid Simulation of Engineering and Engagement Level Models." *Proceedings of the 2009 Spring Simulation MultiConference*.

Xiong, W., P. Fan, and H. Zhang. 2012. "HLA Based Collaborative Simulation with MATLAB Seamlessly Embedded." *International Journal of Machine Learning and Computing* 2(2):123-130.

Zeigler, B. P., T. G. Kim, and H. Praehofer. 2000. *Theory of modeling and simulation*. 2nd ed., Academic Press, Orlando, FL, USA.

Zheng, S., J. He, J. Jin, and J. Han. 2009. "DDS Based High Fidelity Flight Simulator." *Proceedings of the 2009 WASE International Conference on Information Engineering*, 548-551.

AUTHOR BIOGRAPHIES

DOHYUNG KIM is a Senior Researcher at Modeling and Simulation Division, ADD, Republic of Korea. Agency for Defense Development. He received his M.S in Electrical Engineering from KAIST. His practical research includes reliability, availability and maintainability (RAM) simulation, discrete event systems modeling and simulation, and military engagement simulation. His email address is edooroo@gmail.com.

OCKHYUN PAEK is a Senior Researcher at Modeling and Simulation Division, ADD, Republic of Korea. Agency for Defense Development. She received her M.S in Computer Science from Chungbuk National University. Her research interests include defense modeling and simulation and software product line engineering. Her email address is ohpaek@add.re.kr.

TAEHO LEE is a Senior Researcher at Modeling and Simulation Division, Agency for Defense Development (ADD), Republic of Korea. He received his Ph.D. in Information and Communication Engineering from KAIST. His research deals with Live-Virtual-Constructive simulation, software process improvement, and software reliability. His email address is lth2094@add.re.kr.

SAMJOON PARK is a Principal Researcher and division chief of Modeling and Simulation Division, Agency for Defense Development (ADD), Republic of Korea. He received his Ph.D. in Operations Research Lab from KAIST. He developed several kinds of optimistic models for logistics support analysis and product data management information system since 1990 in ADD. He is now working for development of AddSIM as a project manager. His research area is now focused on engagement level simulation to evaluate weapon system performance. His email address is samjoon@add.re.kr.

HYUNSHIK BAE is a Director at M&S 1 Department, SIMNET Co., Ltd. He received his B.S in Computer Science from Daegu University. His research includes defense modeling and simulation and expert systems. His email address is baefurm@simnet.co.kr.