A Dynamic Defense Modeling and Simulation Methodology using Semantic Web Services

Kangsun Lee* **and Byungchul Kim**
Department of Computer Engineering, MyongJi University
San 38-2 NamDong, YongIn, Kyunggi-Do
449-728, Republic of Korea

ABSTRACT

Defense Modeling and Simulations require interoperable and autonomous federates in order to fully simulate complex behavior of war-fighters and to dynamically adapt themselves to various war-game events, commands and controls. In this paper, we propose a semantic web service based methodology to develop war-game simulations. Our methodology encapsulates war-game logic into a set of web services with additional semantic information in WSDL (Web Service Description Language) and OWL (Web Ontology Language). By utilizing dynamic discovery and binding power of semantic web services, we are able to dynamically reconfigure federates according to various simulation events. An ASuW (Anti-Surface Warfare) simulator is constructed to demonstrate the methodology and successfully shows that the level of interoperability and autonomy can be greatly improved.

Keywords: Semantic Web Services, Ontology, Defense Modeling and Simulation

1. INTRODUCTION

As contemporary warfare becomes complex and network-centric, simulation is recognized as the only technique to analyze war strategies and train war fighters accordingly based on the simulation results. A federate is a basic structural unit of war-game simulators and usually combined with other federates to form a "federation"(i.e. simulation)[1]. Federates should provide the followings characteristics:

- Interoperability: In order to build a realistic federation, federates need to be interoperable regardless of operating systems, development platforms, programming languages and middlewares.
- Autonomy: Federates should adapt themselves according to various war events, such as an enemy detection, a new movement, etc. Any changes to the war environment may require dynamic reconfiguration of the whole federation. Failures to meet such changes make the simulation results useless. Therefore, a federate should autonomous enough to decide when and where to join and leave the federation based on the varying war conditions.

HLA (High Level Architecture)[1] is an IEEE standard simulation framework to support the interoperability and reusability of various simulation applications in defense community. Communication between federates is managed by a RTI (Run-Time Infrastructure)[2]. However, the level of interoperability of HLA /RTI is so low that non M&S applications are hard to be interoperable within the standard framework. Also, extensibility is hard to be achieved in non-military WAN and long distance connection. SOA (Service-Oriented Architecture) [3] is an enterprise-oriented conceptual framework to promote the reusability and interoperability of heterogeneous systems. As we need simulation interoperability and reusability within the defense community and commercial enterprise as well, combining HLA with SOA (Service-Oriented Architecture) has attracted many attentions. The fundamental advantages of the combination are ease of interfacing heterogeneous systems for interoperation, availability of commercially developed supporting technologies, and compatibility with the Web based SOA [4]. Many research works have been proposed to combine HLA with SOA, including XMSF (Extensible Modeling and Simulation Framework), OGSA-Based (Open Grid Services Architecture) Simulation Framework, SI3 (Service Integration/Interoperation Infrastructure) and HLA Evolved Web Service API [4-7]. Although these frameworks have made great progress on service-oriented HLA, they lack explicit support dynamic reconfiguration of federates [8]. In this paper, we propose an ontology-driven methodology for defense modeling and simulation to improve interoperability and autonomy of federates. Our methodology 1) encapsulates war-game logic either with HLA-compliant federates or with services of SOA, 2) organizes similar web services into groups and adds semantic information for future dynamic discovery and binding, 3) composes web services for a given war game scenario, and finally 4) monitors web services and dynamically changes obsolete web services, on the fly, by

* Corresponding author: All correspondences should be made to ksl@mju.ac.kr

matching semantic ontology with war events. An ASuW(Anti-Surface Warfare) simulator has been developed with the proposed methodology and successfully demonstrated that we could achieve a higher level of interoperability and autonomy comparing to conventional static simulations.

This paper is organized as follows. In Section 2, we propose our ontology framework for defense modeling and simulation. Section 3 illustrates our methodology with an example of ASuW simulation. We conclude in Section 4 with future works to achieve.

2. ONTOLOGY FRAMEWORK FOR DEFENSE MODELING AND SIMULATION

A federate participates a war-game simulation by cycling the following 4 phases: *movement, detection, engagement,* and *analysis.* Figure 1 shows our ontology framework. The ontology framework has four slots to specify movement, detection, engagement and analysis operations of a federate, respectively.

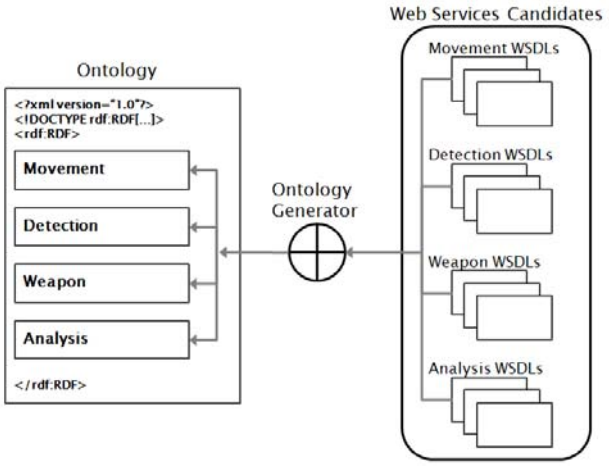

Figure 1. Ontology Framework

We use WSDL2.0 (Web Service Description Language) [9] and OWL [10] to describe the semantics of web services.

WSDL describes interfaces to web services. The interface contains a set of operations each of which has a signature (i.e. operation name, input, output and exception messages). WSDL 2.0 is a lightweight approach for adding semantics to web services. We define an extension to the WSDL 2.0 and use OWL for describing semantics of WS federates. Similar web services are grouped together for dynamic adaptation. For example, missiles are modeled into missile web services and grouped as other weapon services together, for example, harpoon services, and ammunition services. Based on the distance and type of enemies, the optimal weapon service is selected for a simulation; For example, if an enemy is detected as an aircraft in a long distance, a missile web service is employed for a simulation, while an ammunition web service is selected if the enemy is a ship in near distance. As Figure 1 shows, the ontology framework provides a holder to manage web services for automatic discovery and execution. Figure 2 shows an instance of the proposed ontology. `Cannon` weapon web service is described in WSDL2.0 and semantic information, such as type (missile), effective range (min and max), speed and hit accuracy, are described in OWL. We use the semantic information to dynamically select/deselect `Cannon` service according to varying war events. Detailed explanation will be found in Section 3 with an example of Anti-Surface Warfare.

```
<?xml version="1.0" ?>
<description xmlns="http://www.w3.org/ns/wsdl"
targetNamespace="http://selab.mju.ac.kr/Missile/Cannon"
xmlns:tns="http://selab.mju.ac.kr/Missile/Cannon"
xmlns:missile="http://selab.mju.ac.kr/Missile/Cannon"
xmlns:wsoap="http://www.w3.org/ns/wsdl/soap"
xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
xmlns:wsdlx="http://www.w3.org/ns/wsdl-extensions"
:
<documentation>
<xsd:schema xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
xmlns:ship="http://selab.mju.ac.kr/ship/missile/schema#">
<ship:KindOfWeapon rdf:ID="Cannon">
<ship:missileType rdf:datatype="http://www.w3.org/2001/XMLSchema#String">
Cannon</ship:missileType>
<ship:minRange
rdf:datatype="http://www.w3.org/2001/XMLSchema#nonNegativeInteger">
30</ship:minRange>
<ship:maxRange
rdf:datatype="http://www.w3.org/2001/XMLSchema#nonNegativeInteger">
36</ship:maxRange>
<ship:speedOfWeapon
rdf:datatype="http://www.w3.org/2001/XMLSchema#nonNegativeInteger">
3079</ship:speedOfWeapon>
<ship:successfulRate
rdf:datatype="http://www.w3.org/2001/XMLSchema#nonNegativeInteger">
90</ship:successfulRate>
</ship:KindOfWeapon>
</xsd:schema>
</documentation>
<types>
:
</types>
<interface name="CannonInterface">
:
</interface>
<binding name="CannonInterfaceHttpBinding"
:
</binding>
<service name="Cannon" interface="tns:CannonInterface">
<endpoint name="CannonEndpoint" binding="tns:CannonSOAPBinding"
:
```

Figure 2. Ontology Instance: `Cannon` web service

3. AN EXAMPLE: ANTI-SURFACE WARFARE SIMULATION

Figure 3 shows the architecture of semantic WS-based ASuW(Anti-Surface Warfare) simulation. Three federates have been developed for the Anti-Surface Warfare.

- `OwnVessel` federate is modeled with web services. It is responsible for movement of the ship and detection of potential enemies. Based on the target detected, it searches for the best weapon to employ using the weapon ontology and strategy ontology. After engagement, it analyzes the damage level of the own vessel.
- `EnemyVessel` federate is also modeled with web services and has the same logic as `OwnVessel` federate.
- `C2(Command and Control)` federate is modeled with a HLA-compliant Java object. It monitors `OwnVessel` federate and `EnemyVessel` federate and visualizes current status of simulation. It is also responsible for generating various war events, for example, `Target Joined`, `Typhoon Approaching`, `Target Leaved`, and etc.

The three federates are distributed over the network and are interoperable via LRC (Local RTI Component) and WSPRC (Web Service Provider RTI Component)[8]. In this section, we will show how `OwnVessel` and `EnemyVessel` federate can reconfigure themselves by dynamically switching their weapon services according to distance and engagement strategies.

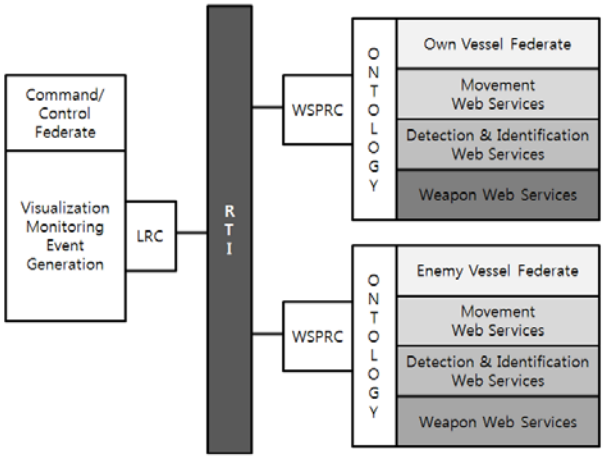

Figure 3. Architecture of semantic WS based ASuW simulator

3.1 Ontology Construction

As we construct the weapon system of a vessel by assembling weapon parts (ex. harpoons, torpedoes, missiles, etc) in real world, the weapon ontology of a vessel is constructed by selecting weapons on our ontology interface as shown in Figure 4.

The `OwnVessel` federate has `SM2`, `ExtendedSM2`, `ChunYongMissile`, `GoolKeeper`, `HeaSungMissile`, `Cannon`, and `Torpedo` anti-surface weapons of ROKN (Republic of Korea Navy). Upon selecting these weapons, the weapon ontology of the own vessel is automatically generated with WSDL2.0+OWL as shown in Figure 5.

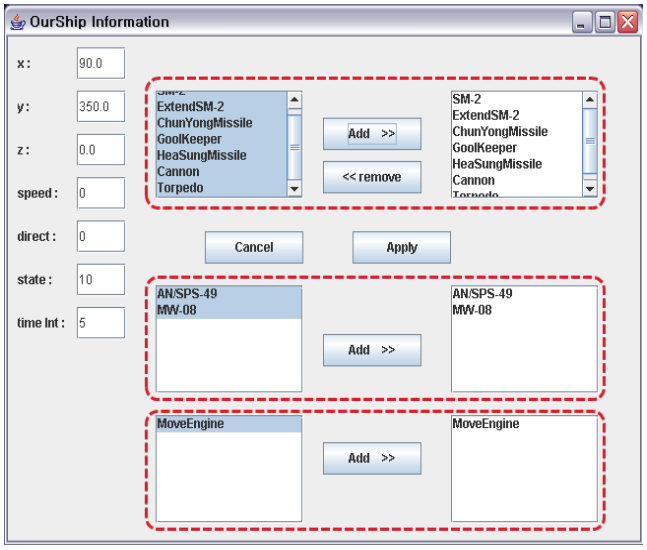

Figure 4. Ontology Interface of ASuW simulator: SM-2, ExtendedSM-2, ChungYong, GoolKeeper, HeaSungMissile, Cannon and Torpedo are selected

```
<owl:ObjectProperty rdf:ID="weaponOfShip">
<rdfs:range rdf:resource="#KindOfWeapon" />
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasWeapons">
<rdfs:range rdf:resource="#WeaponsOfAegis" />
</owl:ObjectProperty>
<ship:WeaponsOfAegis rdf:ID="AegisShip">
<ship:weaponOfShip rdf:resource="#SM2" />
<ship:weaponOfShip rdf:resource="#ExtendSM2" />
<ship:weaponOfShip rdf:resource="#ChunYongMissile" />
<ship:weaponOfShip rdf:resource="#GoolKeeper" />
<ship:weaponOfShip rdf:resource="#HeaSungMissile" />
<ship:weaponOfShip rdf:resource="#Cannon" />
<ship:weaponOfShip rdf:resource="#Torpedo" />
</ship:WeaponsOfAegis>
<ship:KindOfWeapon rdf:ID="SM2">
<ship:missileType rdf:datatype="http://www.w3.org/2001/XMLSchema#String">
Missile</ship:missileType>
<ship:minRange
rdf:datatype="http://www.w3.org/2001/XMLSchema#nonNegativeInteger">
74</ship:minRange>
<ship:maxRange
rdf:datatype="http://www.w3.org/2001/XMLSchema#nonNegativeInteger">
167</ship:maxRange>
<ship:speedOfWeapon
rdf:datatype="http://www.w3.org/2001/XMLSchema#nonNegativeInteger">
9600</ship:speedOfWeapon>
<ship:successfulRate
rdf:datatype="http://www.w3.org/2001/XMLSchema#nonNegativeInteger">
85</ship:successfulRate>
<ship:countOfWeapon
rdf:datatype="http://www.w3.org/2001/XMLSchema#nonNegativeInteger">
30</ship:countOfWeapon>
</ship:KindOfWeapon>
…
```

Figure 5. `OwnVessel` Weapon Ontology

3.2 Dynamic Simulation

The ASuW simulation cycles 1) movement, 2) detection, 3) engagement, and 4) analysis phase forever until either the own vessel or the enemy vessel is destroyed. During the simulation, `OwnVessel` federate and `EnemyVessel` federate dynamically reconfigure themselves by switching weapon web services according to the varying war events. The optimal weapon web service is selected by querying the following into the ontology.

Query in Native Language:
By considering the current distance between the target, the weapon type of the target, our weapon types and the number of remaining weapons, What is the optimal weapon to apply for now?

Query in MySQL:

```
select ship:kindof Weapon where ship:minRange < distance and ship:maxRange > distance and ship:weaponType == weaponTypeAgainst fromTarget and ship:countofWeapon > 0 from OurWarShip
```

Table 1 summarizes simulation results and shows weapon services dynamically selected for various war conditions.

Table 1. Simulation Results

	OwnVessel		EnemyVessel	
Time	Position (x,y,z)	Selected Weapon	Position	Selected Weapon
0	90.350.0	SM2	787.352.0	n/a
3	99.349.0	SM2	787.352.0	n/a
7	109.348.0	ChunYong	774.354.0	n/a
10	189.341.0	ChunYong	761.356.0	n/a
15	219.338.0	HaeSung	710.365.0	HaeSung
17	249.336.0	HaeSung	659.374.0	HaeSung

Table 2 summarizes the implementation environment of the ASuW simulator.

Table 2. Implementation Environment

Federate / Environment	OwnVessel, EnemyVessel	Command/Control
OS	Windows XP	Windows XP
Main Memory	2Gb	1Gb
CPU	2.33Ghz	1.5Ghz
Programming Language	JAVA	
Platform	JAVA 5.0 JDK + Apache Tomcat 4	
Scenario DB	MySQL DataBase	
RTI	poRTIco v0.8 [11]	

Figure 6 visualizes the simulation results.

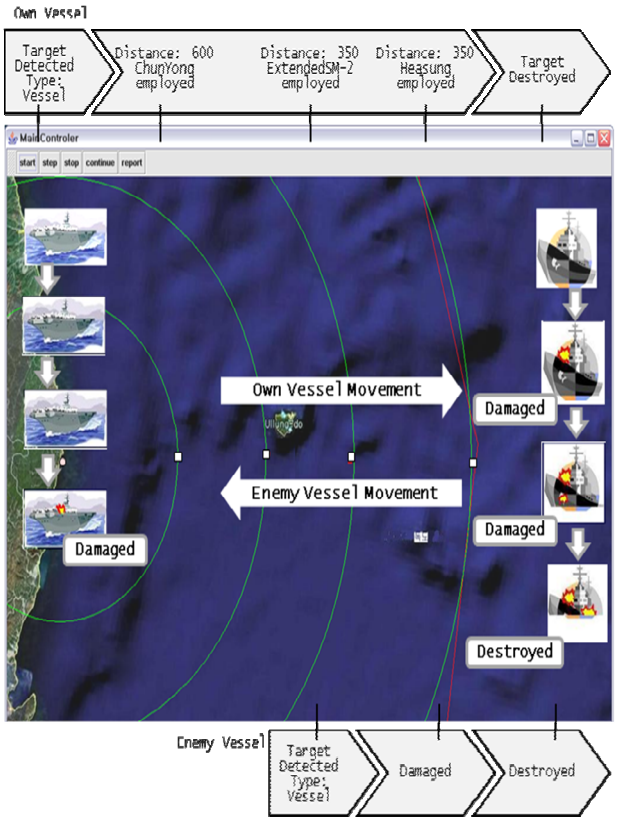

Figure 6. Visualization of ASuW Simulation

4. CONCLUSION

In this paper, we proposed a semantic WS (Web Service)-based methodology for defense modeling and simulation. Our methodology can greatly extend interoperability of federates by allowing communications between military M&S resources (i.e. HLA-based federates) and non-military M&S resources (i.e. SOA-based federates). Unlike the HLA-based static simulation where federates and interactions are fixed in integration time, our methodology can reconfigure federates on the fly by dynamically searching, discovering and binding web services according to the various war events.

In the future, we would like to enhance the ontology framework to support more complex reconfiguration. Also, we also plan to explore various war-game case studies to continuously improve our methodology.

5. REFERENCES

[1] IEEE Std 1516-2000. **IEEE standard for modeling and simulation (M&S) high level architecture (HLA) - framework and rules**, 2000

[2] IEEE Std 1516-2000. **IEEE standard for modeling and simulation (M&S) high level architecture (HLA) – federate interface specification**, 2000
[3] Erl T., **Service-Oriented Architecture: Concepts, technology and design**, Prentice Hall PTR, 2005
[4] Brutzman D, Zyda M, Pullen M, Morse KL, **Extensible Modeling and simulation Framwork(XMSF) challenges for Web-based modeling and simulation**, http://www.mocesinstitute.org/xmsf/XmsfWorkshopSymposiumReportOctober2002.pdf
[5] Li BH, Chai XD, Di YQ, Yu HY, Du ZH, Peng XY., **Research on service oriented simulation grid: Autonomous Decentralized Systems**, 2005, IAD 2005, Proceedings, 2005, pp 7-14
[6] Strelich TP, Adams DP, Sloan WW. **Simulation-based transformation with the service integration/interoperation infrastructure**, Technology Review Journal, 2005, Vol. 13, No. 2, pp 99-115
[7] Möller B, Dahlin C., **A first look at the HLA Evlopved Web Serive API**, In proceedings of Euro Simulation Interoperability Workshop, 2006
[8] Wenguang WANG, **Service-Oriented High Level Architecture**, Simulation Interoperability Standards Organization, 08E-SIW-022., 2008
[9] **Web Services Description Language (WSDL) Version 2.0**: RDF Mapping, (2007), http://www.w3.org/TR/wsdl20-rdf/
[10] Lee W., Lacy. **OWL: Representing Information Using the Web Ontology Language**, Trafford Publishing., 2005
[11] poRTI, http://www.porticoproject.org/index.php?title=Main_Page, Latest Stable Version (Portico v0.8)

ACKNOWLEDGEMENT

This work was supported by Defense Acquisition Program Administration and Agency for Defense Development under the contract UD080042AD, Republic of Korea