International Journal of Database Theory and Application
Vol.10, No.8 (2017), pp.71-82
http://dx.doi.org/10.14257/ijdta.2017.10.8.07

Development of Ontology Engine for Interoperability of L-V-C Integrating System

Gap-Jun Son[1], Yun-Hee Son[2] and Kyu-Chul Lee[*]

[1,2,*] *Department of Computer Engineering, Chungnam National University, 220 Gung-Dong, Yuseong-Gu, Daejeon, Korea*
[1,2,*] *{songj9766, mellow211, kclee}@cnu.ac.kr*

Abstract

As Modeling and Simulation(M&S) techniques develops, L-V-C training system is getting attention to enhance effectiveness of training complementing restrictions of real operation training. L-V-C training system is consisted of Live, Virtual and Constructive and each compositions use HLA(High-Level Architecture), DDS(Data Distribution Service) and DIS(Distributed Interactive Simulation) middleware suitable for each characteristic. However, since the data formats of each middlewares are different, when the L-V-C system is used together, a heterogeneity problem of data types occurs. In this study, for integrating data formats that are different each other, L-V-C Object Data Linking Model is defined. In addition, Using Ontology that is one of Semantic Web techniques and converting uniform data format through L-V-C Ontology Modeling, we can get data interoperability. By using L-V-C Ontology Engine Library as an interface, external systems like L-V-C Gateway can use L-V-C Ontology. And, the data, which is used in L-V-C Training System has mapping information, can be confirmed by using the RelFinder that is an Ontology visual tool.

Keywords*: LVC, HLA, DDS, DIS, Interoperation, Heterogeneity, Ontology, Modeling, Mapping Middleware, Ontology Virtualization, Semantic, RelFinder*

1. Introduction

Because it requires a lot of budget and manpower to predict actual military training and war situation, it enables simulation based on high IT skill in modern times. Modeling and Simulation (M&S), which is used for military purposes, plays an important role in modern warfare[2].

In addition, the military paradigm is changing to Network-Centric Warfare, which can control all units, weapon systems and soldiers in combat areas. In the Network-Centric Warfare environment, operational network for all combat elements is key issue, therefore, interoperability is necessary with all kinds of individual simulations. In other words, they need Communication Middleware Framework for large-scale simulation training, which can integrate all kinds of individual Live-Virtual-Constructive simulations[3]. Therefore, we need connecting skill each large-scale L-V-C system in order to manage military system integrally using the combined organic one system.

For now, internal simulation training system in L-V-C system is based on HLA(High-Level Architecture), however, distributed environment of national defense system uses DDS(Data Distribution Service). And In distributed environment, in order to support real-time, DIS(Distributed Interactive Simulation) is used. Thus, heterogeneous middleware used in L(Live)-V(Virtual)-C(Construct) scheme is HLA/RTI, DDS and DIS. They use different data formats such as FOM, PDU and TOPIC. This is because operation functions and support services are different depending on the characteristics of each middleware. However, HLA, DDS and DIS tend to restrict management function in variant middleware and support service because those middlewares have different data

ISSN: 2005-4270 IJDTA

format.

In [2], Through the API mapping test between HLA and DDS, it showed the possibility of interworking between two heterogeneous middleware. However, this study has difference in using semantic web based ontology.

And, in [4] as related work, they use two kinds of middleware such as HLA and DDS to interwork L-V-C training system. It has similarity our work to resolve heterogeneity problem by using Ontology.

Since the training systems of L-V-C use different middleware, it is important to develop a communication middleware framework for L-V-C interworking in order to use them in real time. The goal is to interwork heterogeneous middleware based on Ontology that is a standard technology of semantic web and interoperate with each other. There are many ways to do this, but most of them have a problem of scalability due to a large increase in overhead as the complexity increases. Therefore, we need Ontology model and processing engine, based on the ontology that can support scalability, that link to data format of DDS/HLA/DIS and interoperate heterogeneous middleware. Unlike existing methods, Interworking using Ontology supports extensibility through inference, so the overhead does not increase even if the complexity increases. Therefore, this study is a very important technology that can play a leading role for the L-V-C integration system.

In this paper, we make a goal to get interoperability through uniformed data format based on ontology, in order to resolve heterogeneity problem in different data format in three kinds of middlewares including DIS not only two kinds like [4]. In addition, we can provide mapping information, which is HLA to DDS or HLA to DIS by using Ontology mapping relation.

2. Background

In this paper, we developed the tool, based on C++, for integrating of L-V-C training system and, in this process, we used data formats of various middleware and Ontology technique based on semantic web. This chapter describes the background techniques used to develop Ontology processing engine technology

2.1. L-V-C Training System

The LVC Training System[1][3] means the combined training environment is created by connecting Battle Command Systems and L(Live)-V(Virtual)-C(Constructive) training assets utilizing networking technology and simulation interworking techniques centering on Synthetic Environment, Synthetic Battlefield or Battle Space. The LVC Training System has attracted worldwide attention in terms of maximizing the efficiency of training on the battlefield situation judgment to middle and top level commander as well as training of lower level battle unit by highlighting advantages of independent simulation and compensating for weaknesses.

High-Level Architecture (HLA). HLA is a general-purpose architecture for distributed processing computer simulation system. Also, it is a standard of interoperability for distributed processing simulation of engineering, training, analysis of various fields, etc. with HLA, computer simulation can be interworked to other computer simulation regardless of the computer platform. Communication between simulations is managed by Runtime Infrastructure(RTI). HLA, which has these characteristics, is widely used in defense, space, air traffic control, railway and automobile industry, and manufacturing industries.

Data Distribution Service (DDS). In message-oriented system environment which was defined by OMG(Object Management Group), DDS is widely used middleware. Establish communication, in DDS, is basically composed of linked form with data provider and users(Publisher/Subscriber). Also, DDS is being used as middleware based on real-time and can provide high responsibility from supported various QoS(Quality of

Service). Based on these features, DDS is used as a middleware for large-scale interworking of real defense systems requiring highly reliable data exchange on real-time. There may be several publishers and subscribers for a topic that publishes a data flow of the type defined as Topic. Because one topic is used, interoperability is ensured even for heterogeneous systems.

Data istributed Interactive Simulation (DIS). DIS as the standard of IEEE, is a simulation protocol that allow data message called PDU(Packet Data Unit) on a distributed network to be sent and received while participating in combat simulations in distributed locations. DIS is standard protocol for simulations used by NATO countries for more than 15 years. It defines and uses format using packet called Protocol Data Unit or PDU. Thus, using DIS is efficient when considering only data items and data transmission bandwidth which is sent in L-V-C simulation. The DIS is defined as a small message type. It can activate without proactive action on heterogeneous server and operating system. And, flexible transmission and reception data items can be adjusted through PDU class definition and modification.

L-V-C Gateway. It is a platform which uses different middleware in order to implement the L-V-C Integrating System. It uses various middleware such as HLA, DDS and DIS. There are several cases using together with other middleware.

Interoperability. Contending with the software and implementation details of interoperations; this includes exchange of data elements via interfaces, the use of middleware, mapping to common information exchange models

2.2. Semantic Web Technology

The Semantic Web is an intelligent web that allows machines to communicate with each other by expressing them in a new language that can be understood by the computer instead of the current web, which is designed to be easy for people to read and interpret. It is a technology that enables the web to understand and handle the meaning of information by giving meta data, which is a language describing resources such as resource(subject), attribute(predicate) and attribute value(object). In this study, techniques based on the W3C is used for integrating the L-V-C Training System.

MSXML(Microsoft XML Core Services). It provides a comprehensive set of W3C compliant XML APIs for building high-performance XML-based applications. The detail variations, clarifications, and extensions to certain final approved web standards supported by MSXML 3.0 and MSXML 6.0 are documented in Internet Explorer Standards Support Documentation. It supports DOM, SAX, an XSLT 1.0 processor, XML schema support including XSD and XDR, as well as other XML-related technologies.

N-Triple. It is a concrete syntax for RDF. N-Triples is an easy to parse line-based subset of Turtle. The syntax is a revised version of N-Triple as originally defined in the RDF Test Cases. N-Triples triples are a sequence of RDF terms representing the <subject>, <predicate>, and <object> of an RDF Triple. These may be separated by white space.

SPARQL(SPARQL Protocol And RDF Query Language). The SPARQL is query language and protocol to process Ontology data which is composed of RDF, N-Triple etc. It has a similar structure to SQL(Structured Query Language) to inquire data stored within relational database. It consists of three parts, SELECT for return result of query, FROM which means DataSet storing Ontology and WHERE to process queries.

Rule-based Inference. The inference function is based on Ontology data and derives the nonexistent data using the new proposition through SPARQL syntax. It generally deals with the logical relation between premises and conclusions. And the conclusion is new derived proposition and the premise is an already known proposition that provides a basis for conclusion. A set of Inference Rule between premises and conclusions is generally called Rule.

3. Ontology-based L-V-C Object Modeling

L-V-C training system use various middlewares which have different data format. Different kinds of middleware couldn't get interoperability each other because those have different data types such as HLA, DDS, DIS. However, in different kinds of middleware, data formats are different but which parts have similarity can be linked sameAs relation. Using sameAs relation can link an individual object to another individual object.

In this paper, we developed C++-based Tool can get interoperability through using ontology-based data linking modeling for getting interoperability and expressing different data formats in each middleware's heterogeneous data formats to ontology type. In Section 3.2, we explain an example of L-V-C Ontology.

3.1. Data Linking Model for L-V-C

Object, In L-V-C training system, can be categorized for three parts. Those are Middleware object, Class object and Attribute object. Each of three objects have attribute which is attribute's value.

[Figure 1.] is result of L-V-C data linking Ontology modeling used uniformed format for solving heterogeneity problem on data format. In this ontology modeling, predicate vocabulary is represented by DC (Dublin Core)[5]'s vocabulary such as <dc:title> and <dc:description> that are often used in many systems. In other predicate, we define new vocabulary that is suitable for L-V-C System. ' * ' is an abbreviated prefix on L-V-C Ontology.

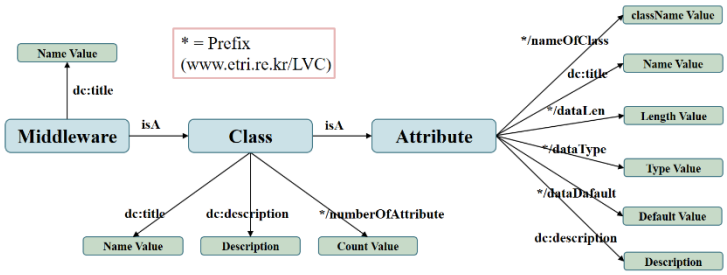

Figure 1. L-V-C Ontology Modeling

Middlware. L-V-C training system use various middleware such as HLA/RTI, DDS, DIS. Object of middleware, in L-V-C Ontology modeling, express middleware information in each object class. In L-V-C Ontology, middleware object has a Name value which means an exhibitive each of middleware's names such as HLA, DDS and DIS.

Class. Three middlewares have different data format. HLA uses Object/interaction data format, and DDS uses Topic format. Moreover, DIS uses PDU data format. Three middleware's data formats can't use in a system because they are different each other. Therefore, those necessary converts to ontology, which is an integrated data type for using heterogeneous data format. Then they can get interoperability by using uniformed data format. Class object has <dc:title>, <dc:description>, and <*:numberOfAttirbute> predicate. This predicate means class object's Name value, description and count value which number of its lower Attribute Class. Interoperable class objects in each of middlewares can have a mapping relation through <*:sameAs> relation.

Attribute. Attribute object as lower object of Class, includes its Class information such as title, description and attribute's value which is composed name, data length, data type, default value and *etc*. Interoperable attribute among attributes in each middleware has mapping relation in each ontology data through <*:sameAs> relation.

3.2 Data Linking Model in L-V-C System

[Figure 2.] is example of data linking modeling in L-V-C System. In this figure, we use two kinds of middleware called HLA and DDS. HLA and DDS are the typical middlwares for representing L-V-C system. In this example, we use an abbreviated L-V-C Ontology prefix as a vocabulary '*' to describe simply.

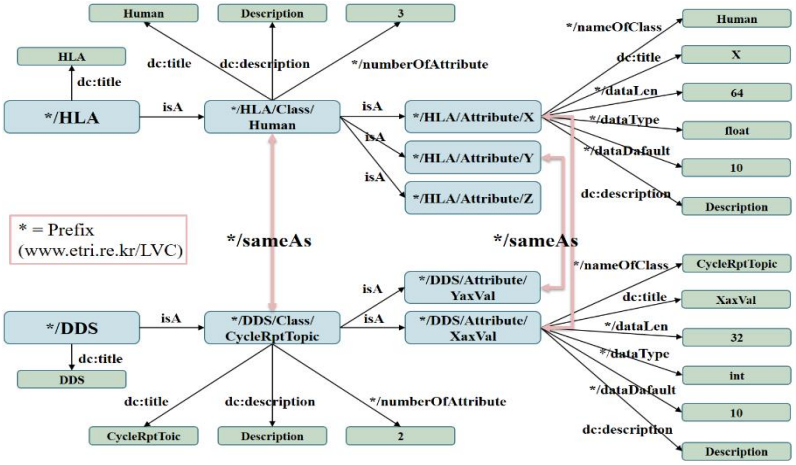

Figure 2. Example of Data Linking Model

HLA middleware object uses <dc:title> predicate and then has 'HLA' name value. Also, DDS middleware object has 'DDS' name value too. HLA middleware links Human class, and DDS middleware links CycleRptTopic class as lower class through <*/isA> relation. Each of classes can be explained for its class in detail by name value, description and number of its attribute object. A human class of HLA middleware has X, Y, and Z attribute classes, so count value linked by <*/numberOfAttribute> is defined to 3. Also, CycleRptTopic's count value is defined to 2, because CycleRptTopic has XaxVal and YaxVal attriutes. Each of attribute classes can be explained what is upper class of its attribute class through <*/nameOfClass> predicate. It has attribute's value such as title, data length, data type, data default value and description.

In L-V-C system, objects in each of heterogeneous middlewares usually have similarity. And then, they can be linked by <*:sameAs> relation. A human class of HLA middleware can be interoperable with CycleRptTopic of DDS because they have similarity. Also, X attribute object of Human class has an interoperable part with XaxVal attribute of DDS, and Y of Human can be interoperable with YaxVal of CycleRptTopic. Therefore, Human and CycleRptTopic can be linked through <*:sameAs> relation. Also X-XaxVal and Y-YaxVal (HLA class to DDS class) can be linked by <*:sameAs> too. However, Z attribute object of Human can't be linked with attribute of CycleRptTopic because CycleRptTopic doesn't have any interoperable object.

```
<http://www.etri.re.kr/LVC/HLA> <dc:title> "HLA" .
<http://www.etri.re.kr/LVC/DIS> <dc:title> "DIS" .
<http://www.etri.re.kr/LVC/DDS> <dc:title> "DDS" .
<http://www.etri.re.kr/LVC/HLA> <http://www.etri.re.kr/LVC/isA> <http://www.etri.re.kr/LVC/HLA/Class/Aircraft> .
<http://www.etri.re.kr/LVC/DIS> <http://www.etri.re.kr/LVC/isA> <http://www.etri.re.kr/LVC/DIS/Class/EntityState> .
<http://www.etri.re.kr/LVC/HLA/Class/Aircraft> <dc:title> "Aircraft" .
<http://www.etri.re.kr/LVC/DIS/Class/EntityState> <dc:title> "Entity State" .
<http://www.etri.re.kr/LVC/HLA/Class/Aircraft> <http://www.etri.re.kr/LVC/isA> <http://www.etri.re.kr/LVC/HLA/Class/Aircraft/Attribute/EntityType> .
<http://www.etri.re.kr/LVC/HLA/Class/Aircraft/Attribute/EntityType> <http://www.etri.re.kr/LVC/nameOfClass> "Aircraft" .
<http://www.etri.re.kr/LVC/HLA/Class/Aircraft/Attribute/EntityType> <dc:title> "EntityType" .
<http://www.etri.re.kr/LVC/HLA/Class/Aircraft/Attribute/EntityType> <http://www.etri.re.kr/LVC/dataLen> "0" .
<http://www.etri.re.kr/LVC/HLA/Class/Aircraft/Attribute/EntityType> <http://www.etri.re.kr/LVC/dataType> "EntityTypeStruct" .
<http://www.etri.re.kr/LVC/HLA/Class/Aircraft/Attribute/EntityType> <http://www.etri.re.kr/LVC/dataDefault> "" .
<http://www.etri.re.kr/LVC/HLA/Class/Aircraft/Attribute/EntityType> <http://www.etri.re.kr/LVC/sequence> "1" .
<http://www.etri.re.kr/LVC/HLA/Class/Aircraft> <http://www.etri.re.kr/LVC/isA> <http://www.etri.re.kr/LVC/HLA/Class/Aircraft/Attribute/EntityType.EntityKind> .
<http://www.etri.re.kr/LVC/HLA/Class/Aircraft/Attribute/EntityType.EntityKind> <http://www.etri.re.kr/LVC/nameOfClass> "Aircraft" .
<http://www.etri.re.kr/LVC/HLA/Class/Aircraft/Attribute/EntityType.EntityKind> <dc:title> "EntityType.EntityKind" .
<http://www.etri.re.kr/LVC/HLA/Class/Aircraft/Attribute/EntityType.EntityKind> <http://www.etri.re.kr/LVC/dataLen> "0" .
<http://www.etri.re.kr/LVC/HLA/Class/Aircraft/Attribute/EntityType.EntityKind> <http://www.etri.re.kr/LVC/dataType> "HLAoctetN_slsh_Aperfectalways1" .
<http://www.etri.re.kr/LVC/HLA/Class/Aircraft/Attribute/EntityType.EntityKind> <http://www.etri.re.kr/LVC/dataDefault> "1" .
<http://www.etri.re.kr/LVC/HLA/Class/Aircraft/Attribute/EntityType.EntityKind> <http://www.etri.re.kr/LVC/sequence> "2" .
```

Figure 3. Example of Data Linking Model(N-Triple)

Above picture(Figure3) show result of created L-V-C Ontology(N-Triple file) based on Data Linking Model. This example is represented by N-Triple[6]. Through predicate of <dc:title>, HLA, DDS and DIS as middleware have values of “HLA”,”DDS” and “DIS” respectively. <*:isA> predicate means Subject has an Object and each middleware has Object Class. In the above example, we can see that HLA and DIS have an Object Class of “Aircraft” and “EntityState” respectively. Also, we can confirm that “Aircraft” of middleware has an Attribute Class of EntityType, EntityType and EntityKind, has an Attribute Value of dataLen, datatype and dataDefault, etc. The Ontology is described in detail in the next chapter.

4. L-V-C Ontology Engine

The L-V-C system uses various middleware depending on the characteristics of each training system such as HLA, DDS and DIS. However, those middlewares have limit to direct linkage with other middleware like heterogeneity problem for data format each of training systems because they have different data formats.

Each data format is analyzed for interoperbility of data format(FOM, TOPIC, PDU) used in hetergenous middlewares(HLA/RTI, DDS, HLA) and it is derived as LVC object and data connection model. User input data(XML) was parsed by MSXML[7] and stored as N-Triple files through Ontology model. We also designed and developed the L-V-C Ontology System which includes function of storing the N-Triple file created in the above process in Virtuoso 7(as a Triple Store)[8]. Through this development, Interface of L-V-C Ontology System is provided for hetergeonous middleware of data mapping in LVC interworking gateway. SPARQL[9] is inquired through getting Key(example : HLA Class name) which needs for Ontology query from Interface.

We developed function of processing to stroe result value of Query to already defined structure and return them to LVC interworking gateway. The Interface of this development provide them as Library. Thus, Because Ontology data that is through Library of Ontology Engine is used, we can avoid heterogeneity problems with heterogenous data types.

Lastly, we developed the function to preform connection automatically through inference when each of them don’t have connection information and Library proivides all of function through the LVC Ontology Interface tool.

4.1 User Input Data

Users can input data and relation between each heterogenous middleware using Ontology Interface Tool. Different kinds of middleware data which is used in L-V-C System written by user to XML file. User Input Data(XML) file includes heterogenous middleware data(Object Class name, data type, length, and so which are included in each middleware) which is used in L-V-C Training System. Also, it includes mapping information between HLA – DIS and HLA – DDS.

The L-V-C training system use heterogenous middlewares which have different data formats. In Ontology Engine of this paper, users personally input the XML file used as input value. Created XML file stores values (Name, Fom, datatype, dataLen, def, and etc.) of HLA and DDS, HLA and DIS and mapping information between heterogenous middlewares. This XML file is parsed using XML Parser function of Ontology Engine.

4.2. Architecture of L-V-C Ontology Engine

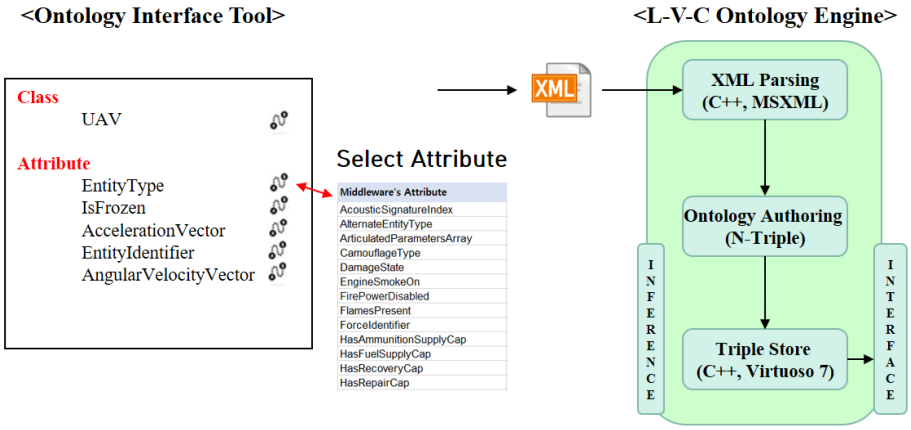

Figure 4. L-V-C Ontology Engine Overall Architecture

[Figure 4.] is an overall structure of a L-V-C Ontology Engine to solve heterogeneity problem. The Ontology Engine is consisted of three steps such as XML parsing, Ontology authoring and Triple storing.

XML Parsing. Ontology Engine parses XML file with XML Parsing function that do parse L-V-C object's attribute value in XML file. XML Parsing function is made by MSXML6.0 based on DOM and follows W3C XML standard.

Ontology Authoring. It converts XML file to N-Triple based on ontology by Ontology authoring function based L-V-C Ontology model. Result of Ontology Authoring function, we can get an N-Triple file(Separate each Triple line with a period) that is composed to <subject>, <predicate>, <object>. N-Triple as W3C standard, have advantage that human can read and figure out easily because it has simple format and Text-based syntax.

Triple Storing. This system store the result of Ontology Authoring as N-Triple file in Virtuoso database. Then we can get a result what we want by using SPARQL[6] Query in Virtuoso. Virtuoso7 that is used as Triple Store can store Ontology data such as RDF, Turtle and OWL apart from N-Triple which is used in this study. Also, because Virtuoso7 assist SPARQL as Query Language, users can get data what they want for stored data by using SPARQL Query.

Following above three steps, we can get data interoperability through converting each of different middleware's data, is used in L-V-C System, to N-triple that is uniform data format based on L-V-C Ontology and saving N-triple to Triple Store.

Because the L-V-C Ontology Engine provides interface, Library can be imported and used by other users or platforms. Using this Ontology Engine Tool, the L-V-C Gateway can choose directly and make a N-Triple file. Also, the Ontology Engine can be used as Library. When Gateway uses the library, it can create N-Triple files and store to Triple Store calling parsing function. In Section 4.4, it explains how to use the Library as Interface in external systems.

4.3. Rule-based Inference

The Ontology Engine parses user input data(XML), converts them to Ontology of N-Triple type suitable for data interworking model and stores to Triple Store. However, user input data is divided into two files which are Mapping file of HLA – DDS, HLA – DIS. Thus, specific classes of three middlewares(HLA, DDS, DIS) that are used in real LVC Training System, are all interoperable and can get mapping information through sameAs relation. But, because mapping information is divided to HLA – DDS, HLA – DIS, all the three middlewares can't be made to sameAs relation. To solve this problem, in this project, we use Rule-based Inference.

The triple line of <DDS's Class><sameAs><HLA's Class> are created through parsing the user input data that included mapping information of HLA and DDS, and getting sameAs relation mapped with Class of DDS in specific HLA Class. Through this line, two specific Class of HLA and DDS are confirmed interoperability through mapped each other by sameAs relation. Likewise, the triple line of <HLA's Class><sameAs><DIS's Class> are created through parsing user input data that included mapping information of HLA and DIS and getting.In other words, for the same HLA Class, if there is sameAs relation between HLA and DDS and there is sameAs between HLA and DIS, the DDS class and DIS class can be mapped through the sameAs relation.

[Figure 5] describes above process. The left picture is mapping information between Classes which aren't applied inference. DDS Class has the sameAs relation with HLA Class, HLA Class has sameAs relation with DIS Class. However, DIS Class and DDS Class can't have sameAs relation. Thus, if DIS Class and DDS Class that are mapped by the same HLA Class exist like the right picture, mapping information of DIS - DDS can be created using Rule-based inference. When entering the XML File, Ontology Engine

Ontology Engine perform this process automatically when entering XML File. It checks data whether to exist or not in Triple Store and if it does not exist, add N-Triple data to Virtuoso7(Triple Store).

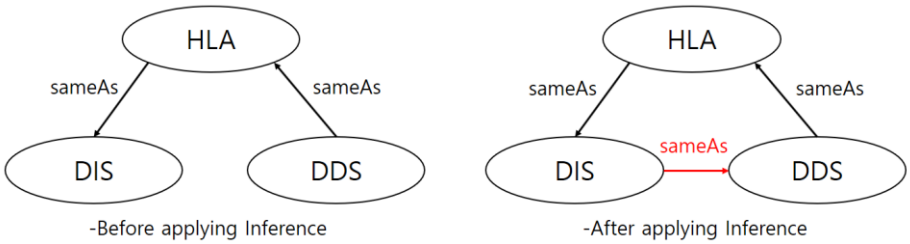

Figure 5. Example of Inference

4.4 L-V-C Ontology Engine Interface

In this study, we provide the Ontology Engine as an interface that can be used by other users and systems. So, L-V-C Gateway can use functions (XML Parsing, Ontology Author, Triple Storing) of section 4.2, using L-V-C Ontology Engine Library. The L-V-C Gateway using this Library can obtain data interoperability by using L-V-C Ontology data stored Virtuoso7 using SPARQL Query. However, in order to load and use the data in Triple Store, an external system such as Gateway needs the means of storing result of query. Thus, because this study is based on C++, we used Data Structure. Data Structure for L-V-C Integrating System is given in [Table 1.].

Table 1. Data Structure for L-V-C Integrating System

Structure Name	Data Type	Data Name	Description
MappingObjectClassInfo	MappingAttributeInfo *	mapping_attr	Attribute list that constitutes a class.
	Int	count	Meaning the number of mappings to heterogeneous middleware.
MappingAttributeInfo	Attribute *	attr	Includes mapping information between middleware for each attribute.
	int	length	Meaning the number of middleware has mapping relation (number of Attribute Structure Instance).
Attribute	String	middlewareName	Meaning the name of Attribute Class.
	String	className, attributeName, dataLength, dataType	Meaning the value of each data.
	String	defaultValue	Meaning default of data

MappingObjectClassInfo. It is a structure to store the result of query about specific Object Class for using in L-V-C Gateway. This Data Structure includes mapping information between heterogenous middlewares(HLA/DDS/DIS) and means how many LVC data are mapped through Count value.

MappingAttributeInfo. It is included in MappingObjectClassInfo Structure and include mapping information of Attribute between heterogenous middlewares and Attribute list that constitutes the Object Class. Through the length value, it means how many mapping relations are interworked to specific Object Class (if length is 3, all three HLA – DDS, HLA – DIS and DDS – DIS are mapped through the sameAs relation).

Attribute. It stores information of Attribute that belong to specific middleware among 3 kinds of middleware. For each attribute, information such as name, dataLength and dataType of middleware is stored.

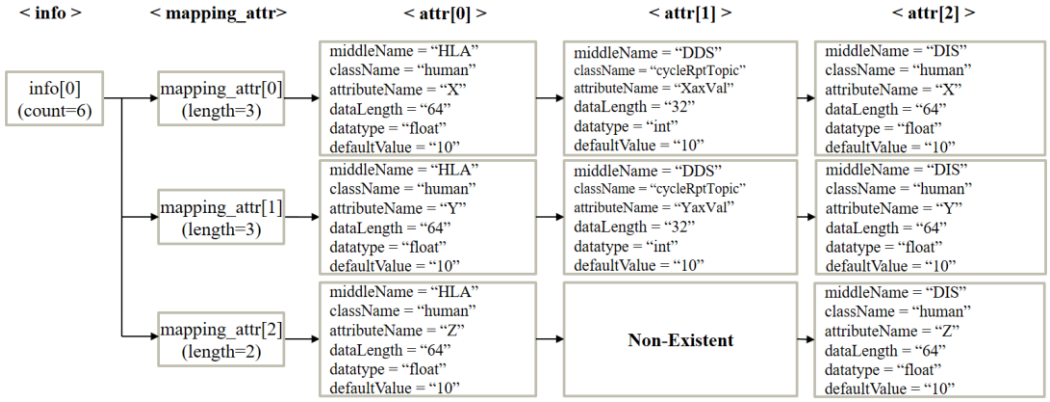

Figure 6. Example of How to Store Data by using Data Structure

[Figure 6.] is a example to show result entering L-V-C data of section 3.2 to structure of [Table. 1.]. To explain in detail, we added data of DIS. Info is a structure and includes mapping information for sub Attribute Classs and mapping information for a single Object Class. Info[0].count is 3, because mapping_attr array that has Mapping information is 3.

The mapping_attr is structure array to store Attribute Class of each middleware. In the

above example, among the Object Class of each middleware, DDS and DIS, which have most Attribute Class, have three Attributes in total. Thus, maximum size of mapping_attr array is 6.

The attr array means a Attribute Class and it means that attr[0] belongs to HLA, attr[1] belongs to DDS and attr[2] belongs to DIS. In the above example, because mapping_attr[0] has all data of attr[0], attr[1] and attr[2], it indicates that all three middleware are mapped. Thus, length is 3. Because mapping attr[2] doesn't exist attr[1] of DDS, length is 2 and it means that only the HLA and DIS Attibutes are mapped.

MappingClassInforReq function that uses Key value as parameter (specific HLA Class name) is defined in Ontology Engine Library. When this function is called with HLA Class used as Key value, DDS and DIS Class that are mapped with this HLA Class All Attribute Class of Class can be returned (One Object Class can have many Attribute Classes). The MappingClassInfoReq function is used as shown in (1) below.

$$\text{struct MappingObjectClassInfo} * \text{info} = \text{MappingClassInfoReq(string objectClassName)} \quad (1)$$

In this paper, we stored result value of function to structure as called info. The L-V-C Gateway receives info structure value and uses Ontology data. In other words, if L-V-C Gateway wants Ontology data for specific Object Class, we use Ontology Engine Library to call MappingClassInfoReq function with the HLA Class's Name of the corresponding Object Class as a parameter. Return value of the called MappingClassInfoReq function is stored to MappingObjectClassInfo structure, L-V-C Gateway can use Ontology data stored in Virtuoso(Triple Store).

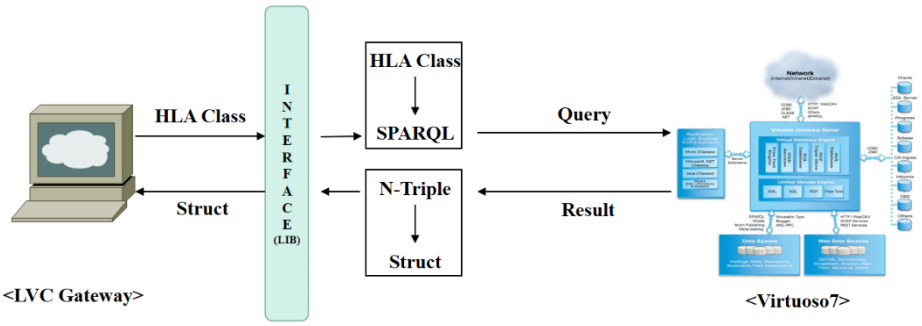

Figure 7. Overall Process of L-V-C Ontology Engine Interface

[Figure 7.] describes all process of Ontology Engine Interface. If L-V-C Gateway call the function using specific HLA Class as input value, it receive the Ontology data(N-Triple) stored in Virtuoso as result value using SPARQL. The L-V-C Gateway can solve heterogeneity of data format by using result value stored in structure of [Table. 1.].

4.5. Visualization of L-V-C Ontology

Ontology Engine create L-V-C Ontology as formed N-Triple. Because N-Triple has high readability, it is easy to see. However more triple lines, the more difficult it is to see with eyes. Thus, data that is used in L-V-C Training System is confirmed more easily after we use RelFinder[10] to visualize ontology. RelFinder extracts and visualizes relationships between given objects in RDF data and makes these relationships interactively explorable.

As an Ontology that express relationship through Relation,, RelFinder expresses not only data but also relation between data. Based on these advantages, In this paper, normal people who don't know Ontology can enable to confirm mapping information between data easily for L-V-C Integrating System.

[Figure 8.] describes thing that insert RelFinder into MFC tool that is developed by this study. The example below indicate relation between "EntityState" of DDS and "Aircraft"

of HLA which are explained in [Figure 3.]. Also, it can be confirmed to express Attribute of “Aircraft” and “EntityState” Class together.

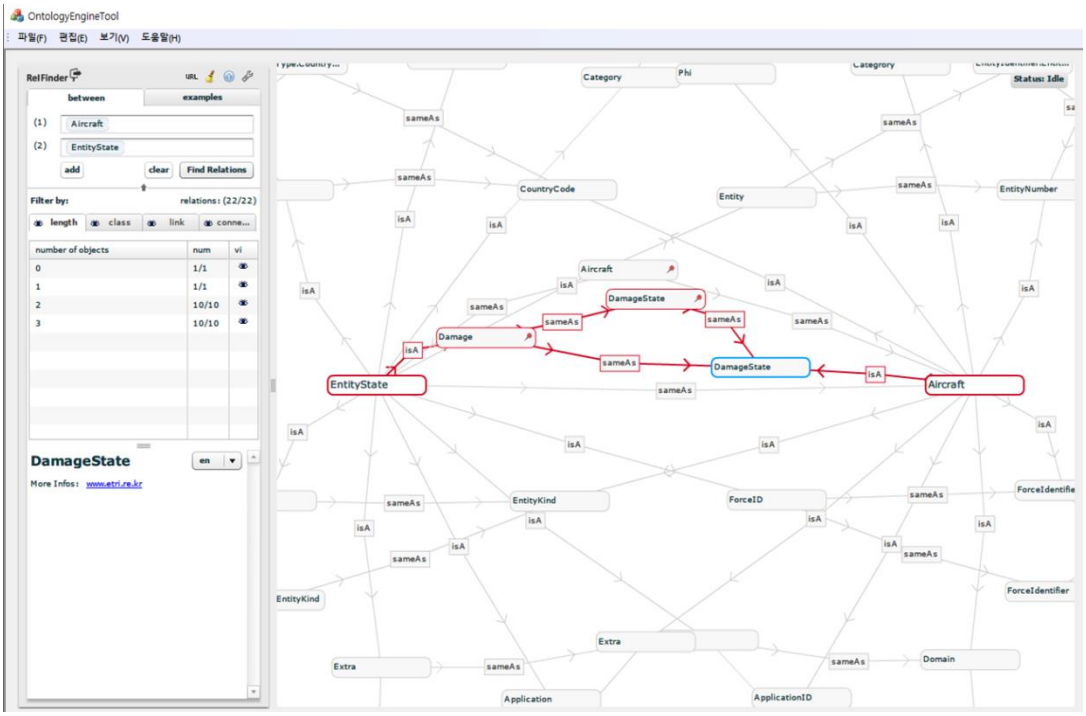

Figure 8. Examples of using RelFinder

5. Conclusion

L-V-C Training System is consisted of L system which is responsible for real operation simulation training, V system of simulation equipment training and C system which is responsible combat command system. And it enables various combat/tactics training by integrating three systems. However, each system composes of different middleware. Because each middleware uses a different data format, heterogeneity problem for data format occurs. So, each of different data formats need to be converted to uniform data format because they don't have compatibility.

In this paper, in order to solve the above problem, we use Ontology technique which is one of the semantic web techniques. We defined L-V-C Ontology modeling for integrating data of three middlewares to uninform data format, developed L-V-C Ontology Engine. Ontology Engine is consisted of XML Parsing function to parse user input file(XML), Ontology Authoring function to create N-Triple based on L-V-C Ontology, Triple Storing function to store created N-Triple automatically and Rule-based Inference function. Heterogeneity problem is solved by using data of each middleware that have different data format using this system.

Through this study, it is possible to use in external systems because we provide library as external interface. We defined interface to use this library in the external systems and can secure the data interoperability to integrate L-V-C system by using all function of Ontology Engine by Library. Lastly, we use RelFinder to visualize interworking data among the data used in the L-V-C system, so that it is easy to see which data has the sameAs relation.

For now, because Ontology Engine developed to base on C++, there may be restrictions on the use of this library by external systems. Thus, we plan to implement it in several programming languages. At this point in time, because this system use SPARQL Endpoint as localhost, it can’t share Ontology data with external systems. To solve this

problem, we plan to make the URL available to public data by using the server for the SPARQL Endpoint.

Acknowledgments

This work was supported by Dual Use Technology Program through Civil Military Technology Cooperation Center funded by MINISTRY OF TRADE, INDUSTRY & ENERGY and DEFENSE ACQUISITION PROGRAM ADMINISTRATION.

This paper is a revised and expanded version of a paper entitled [Design of Ontology Engine Architecture for L-V-C Integrating System] presented at [Gap-Jun Son, Advanced Software Engineering & Its Applications (ASEA 2016) ,Jeju Island, Korea, November 24-26].

References

[1] G.-J. Son, Y.-H. Son and K.-C. Lee, “Design of Ontology Engine Architecture for L-V-C Integrating System”, The 8th International Mega-Conference on Future Generation Information Technology(FGIT), **(2016)**.
[2] K. Cho, G. No, S. Jung, N. Keerativoranan and C. Kim, “A Method of Interoperating Heterogeneous Simulation Middleware for L-V-C Combined Environment”, Journal of KIISE, vol. 42, no. 2, **(2015)**, pp. 213-219.
[3] S.-H. Lee, I.-G. Chun and W.-S. Cha, “Development of Communication Middleware Framework for Real-Time L-V-C Interoperability”, ETRI, **(2015)**.
[4] M.-S. Oh, Y.-H. Son and K.-C. Lee, “An Ontology System for Interoperation between DDS and HLA”, In: Indian Journal of Science and Technology, vol. 8, no. 27, **(2015)**.
[5] Dublin Core Vacabulary, http://dublincore.org/documents/dcmi-terms/.
[6] RDF 1.1 N-Triples, https://www.w3.org/TR/n-triples/.
[7] MSXML 6.0, https://msdn.microsoft.com/en-us/library/ms763742(v=vs.85).aspx.
[8] Openlink Virtuoso, http://virtuoso.openlinksw.com/.
[9] SPARQL 1.1 Query Language, https://www.w3.org/TR/sparql11-query/.
[10] RelFinder, http://www.visualdataweb.org/relfinder.php.

