An UML-based Authoring Approach of S1000D Procedural Data Modules and Tool Support

Youhee Choi, Jeong-Ho Park, Byungtae Jang, DongSun Lim
Vehicle&Defense-IT Convergence Research Department, ETRI
Daejeon, KOREA
e-mail: {yhchoi, parkjh, jbt, dslim}@etri.re.kr

***Abstract*— The S1000D specification is developed as the standard format to describe technical publications in aerospace and military field. The S1000D supports systematically classifying technical information about various equipments in XML format. Most technical information about equipments contains procedural information about installation, operation, and maintenance. The procedural information can be effectively described using graphical description methods rather than using textual description like XML. Also, In UML, activity diagrams can be used to describe the business and operation step-by-step workflows. In the S1000D, there are many types of data modules and procedural data modules regarding procedural information. In this paper, we propose an approach to authoring S1000D procedural data modules using UML.**

Keywords- S1000D; XML; UML; Shipdex.

I. INTRODUCTION

The "S1000D International Specification for technical publication utilizing a common source database" is an international specification for the procurement and production of technical publications [1]. The S1000D covers technical publication activities in support of any airline projects and military projects. However, in the shipbuilding field, most technical manuals are supplied today on paper or in different formats, different structures and different data quality. This situation caused several problems in terms of information comprehension and electronic usage. Therefore, to utilize enterprise resource planning (ERP) and manage various technical manuals effectively, a common and standardized protocol for exchanging technical data was deemed necessary. For this reason, some European shipping companies agreed to develop the Shipdex protocol that is a common and shared data exchange protocol based on ASD S1000D issue 2.3 [2]. The Shipdex protocol allows data exchange, update, and search by standardizing technical information publication format.

The S1000D that is the basis of the Shipdex defines a "Data Module" which is defined as "the smallest self contained information unit" and a data module is described in XML format. XML is a markup language that defines a set of rules for encoding documents [3]. It is a textual data format with strong support via Unicode. XML was designed to carry data, not to display data and XML tags are not predefined. Naturally, there are various XML editors that have facilities like tag completion and on-the-fly XML validation with XML schema [4, 5, 6, 7]. Also, some XML editors allow rendering XML documents using XSLT stylesheets to show them close to the final output. However, producing an XML document using these XML editors requires learning XML schemas or DTDs. In the same way, due to these features of XML, it means that producing a S1000D data module requires pre-learning the S1000D schemas or tags. Also, it is difficult to understand data contained in data modules for non-technical users without transforming data modules supplied in XML format into other format. In addition, there are many studies to model XML with UML[10, 11, 12, 13, 14]. However, the existing researches focus on representing static structures of XML schema elements. There are few studies to model contents of XML using UML in view of semantics of XML elements.

We note that most technical publications for equipments are manuals that contain procedural information about installation, operation, and maintenance. The procedural information can be effectively described using graphical description methods. In this respect, we propose an approach to authoring a S1000D procedural data module using behavioral modeling method of UML that is the de facto standard modeling language. Applying our approach to describing S1000D procedural data modules, it allows to easily produce S1000D data modules and to easily understand contents of the S1000D data modules without knowing XML DTD schema and tags.

The remainder of this paper is organized as follows. The outline of the S1000D and the Shipdex is presented in Section 2. Section 3 presents our approach to authoring S1000D XML data using UML. The approach is applied to an example in section 4. The paper concludes with future work and conclusions in section 5.

II. S1000D AND SHIPDEX

A. S1000D

The S1000D is an international standard for the production and procurement of technical publications. It has

 ISBN: 978-1-61208-230-1

been initially developed by the AeroSpace and Defence Industries Association of Europe (ASD). In the S1000D, a data module is the smallest information unit. Data modules may be stored and managed in the CSDB(Common Source Data Base). The purpose of the CSDB is to manage the data modules so that information is not duplicated, link relationships are maintained, and version control is applied to content [8]. Because S1000D is modular, it facilitates content reuse and seamless data interchange between organizations [9]. There are many data module types which are appropriate for use in the production of all technical information required in operation and maintenance of the product. The S1000D defined various types of data modules-Descriptive, Procedural, Crew/Operator, Fault information, Maintenance planning, Illustrated parts data, Process, Wiring data, Wiring data description, Technical repository, Container, Product cross-reference table, Technical conditions cross-reference table, Business rules exchange. A data module has a basic structure which is comprised of two sections:

•Identification and status (IDSTATUS) section
•Content section

The IDSTATUS section of all data modules contains identification data (data module code, title, issue number, issue date, language) and status data (security classification, responsible partner company and originator, applicability, technical standard, quality assurance status, skill, reason for update). The Content section of a data module must be structured in accordance with data module types. The Content section of a procedural data module that we focus on contains the following elements.

•Data module title (<dmtitle>)
•Table of contents
•References (<refs>)
•Preliminary requirements (<prelreqs>) including safety conditions(<safety>)
•Procedure (<step>)
•Requirements after job completion (<closereqs>)

B. *Shipdex*

The Shipdex protocol is the international business rules developed to standardize the development and the exchange of technical and logistic data within the shipping community. It applies to ASD S1000D at issue 2.3. The Shipdex protocol has been developed by the following companies- Grimaldi Compagnia di Navigazione s.p.a, Intership Navigation Co. Ltd., Alfa Laval, MacGREGOR a part of Cargotec Group, MAN Diesel & Turbo, SpecTec Group Holdings Ltd., Yanmar Co. Ltd. The scope of this protocol is to cover the data exchange related to the information currently supplied in the form of technical manuals. The reason to develop the protocol is that shipping companies are receiving from manufactures technical manuals in different formats, different structures and different data quality. The data module types that the Shipdex protocol makes use are Descriptive, Procedural, and Illustrated parts data (IPD).

C. *Modeling XML schemas with conceptual models*

Several approaches of modeling XML schemas by using existing conceptual models, such as ER, UML have been proposed [10, 11, 12, 13, 14]. The goals of most researches are to represent a XML schema using a UML class diagram, even if the modeler has no familiarity with the XML schema syntax. In this respect, the authors have used the UML extension mechanisms to represent XML schema elements. They focus on representing static structures and data relationships of XML schema elements. The existing researches hardly consider semantics of values which would be represented using XML schema elements.

III. AN UML-BASED AUTHORING APPROACH OF S1000D PROCEDURAL DATA MODULES AND TOOL SUPPORT

A. *Process for authoring S1000D procedural data modules*

This section describes the process for authoring the content section of S1000D procedural data modules as shown in Fig. 1.

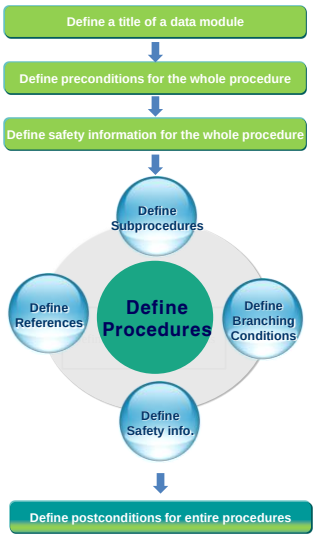


Figure 1 The process for authoring the S1000D procedural data modules

First of all, the title of a data module that represents the whole procedure must be defined. Second, preconditions for the whole procedure that are carried out before starting the procedure must be defined. Third, safety information such as warnings, cautions for the whole procedure must be defined. Then, the main procedure must be defined. To define the main procedure, a subprocedure that is a unit of an action must be defined. For each subprocedure, if they exist, referenced elements that the subprocedure refers and safety information that is related to the subprocedure should be defined. Then, if conditional flows are needed, branching conditions that determine which actions are carried out should be defined. Lastly, postconditions for the whole procedure must be defined.

 ISBN: 978-1-61208-230-1

B. *Method for describing S1000D procedural data modules*

This section describes the method for describing the content section of S1000D procedural data modules using UML. The procedural data module is used to describe procedural information. In the UML, activity diagrams can be used to describe the business and operation step-by-step workflows of components in a system [15]. Thus, we suggest using activity diagrams to describe procedural data modules.

Major elements and attributes of the procedural data module can be classified as shown in Table I.

TABLE I. MAJOR ELEMENTS AND ATTRIBUTES OF THE PROCEDURAL DATA MODULE

Procedural data module DTD/Schema			
Element	***Subelement***	***Subelement***	***Attribute***
<dmtitle>	<techname>		
<refs>	<refdm> or <reftp> <avee> or <pubcode> <dmtitle> or <pubtitle> <issno> or <pubdate>		
<prelreqs>	<supequip> <supplies> <spares>	<nomen> <qty>	id uom
<safety>	<warning> <caution>		
<step>			
<closereqs>			
<xref>			xidtype xrefid

Firstly, the 'Data module title (<dmtitle>)' must give meaning to identification of the product and it has the mandatory element <techname>. The content of the element <techname> must reflect name of the hardware or function. Since a procedural data module can be described as a unit of activity diagram, the element <techname> can be described with the name of the activity diagram or an activity that represents the whole procedure without extending the UML.

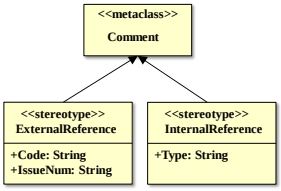
Secondly, the 'Table of contents' element doesn't contain specific contents of the data module. Thus, we can exclude it.

Thirdly, in case of the 'References' element, there are two types of references used in the S1000D.

- Internal references: References to other places within the same data module(<xref>)
- External references: References to other data modules (<refdm>) or other technical publications (<reftp>)

The former (<xref>) type of references corresponds to Figures, tables, multimedia, procedures (steps), and so on. Although, in the DTD/Schema, this type is optional, it gives the detail information about the referenced targets. In this respect, it is necessary to describe this type of references, so identification information about the referenced target should be described. Thus, we define the stereotype <<InternalReference>> by extending UML Comment element with tags that represent ID and type of the referenced target as shown in Fig. 2.

In case of the latter (<refdm> or <reftp>) type of references, these references are the mandatory elements and they should be presented on the references table. The table presents the data module/technical publication code, data module/technical publication title, and issue number. Thus, we define the stereotype <<ExternalReference>> by extending UML Comment element with tag definitions that represent code, title, and issue number as shown in Fig. 2.



S1000D				UML element		
Element	**Subelement**	**Subelement**	**Attribute**	**Stereotype**	**Tag**	**Properties**
<refs>	<refdm> or <reftp>			ExternalReference		
		<dmtitle> or <pubtitle>				body
		<avee> or <pubcode>			Code	
		<issno> or <pubdate>			IssueNum	
<xref>				InternalReference		
			xrefid			body
			xidtype		Type	

Figure 2 S1000D references element vs. UML elements

In turn, the 'Preliminary requirements' element consists of the following sub-elements.

- Required Conditions: actions to be done and/or conditions that must be satisfied before doing procedure(<prelreqs>) and any actions that are required after the procedure is complete(<closereqs>)
- Support equipment: A list of any support equipment including special tools, required to accomplish the procedure(<supequip>)
- Supplies: A list of any consumables, materials and expendables required to accomplish the procedure(<supplies>)
- Spares: A list of any spares required to accomplish the procedure(<spares>)
- Safety: A list of any safety requirements(<safety>)

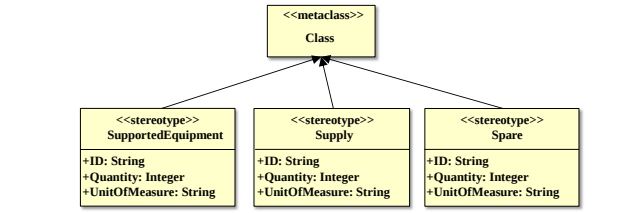
Firstly, in case of the 'Required Conditions', there are two types of Required Conditions.

- Required Conditions with no reference(<reqcond>)
- Required Conditions with external references(<reqcondm>, <reqcontp>)

In case of the former (<reqcond>) type of the 'Required Conditions', since the semantics of the sub-element (<prelreqs>) corresponds to the semantics of the pre-defined stereotype <<precondition>> for the activity, the sub-element (<prelreqs>) can be described with the stereotype <<precondition>> for the activity that represents the whole procedure. In case of the latter (<reqcondm>, <reqcontp>) type of the 'Required Conditions', we define the stereotype <<PreconditionRef>> for precondition and the stereotype <<PostconditionRef>> for postcondition by extending the UML Comment element. In addition, relationships between 'References' and 'Required Conditions' can be described as the linkage between the UML Comment element with the

stereotype <<PreconditionRef>> or <<PostconditionRef>> and the UML Comment element with the stereotype <<ExternalReference>>.

Secondly, in case of the sub-elements (<supequip>, <supplies>, <spares>), each element is mandatory in procedural data modules and can be referenced within the same data module. And each element has a <nomen> element which indicates the functional item nomenclature and a <qty> element which is used to identify the quantity of items. Also, each element has an 'id' attribute and an 'uom' attribute that indicates the unit of measure. To describe each element, we define the stereotype <<SupportedEquipment>>, <<Supply>>, and <<Spare>> by extending the UML Class element with tags that indicate ID, the quantity of items, and the unit of measure as shown in Fig. 3.



S1000D				UML element		
Element	Subelement	Subelement	Attribute	Stereotype	Tag	Properties
<prelreqs>						
	<supequip>			SupportedEquipment		
	<supplies>			Supply		
	<spares>			Spare		
		<nomen>				Class Name
		<qty>			Quantity	
			id		ID	
			uom		UnitOfMeasure	

Figure 3 S1000D 'Preliminary requirements' elements vs. UML elements

In case of the sub-element (<safety>), the type of safety conditions can be warnings or cautions. Thus, we define the stereotype <<Warning>> and <<Caution>> by extending the UML Comment element and body of the comment can be safety conditions as shown in Fig. 4.

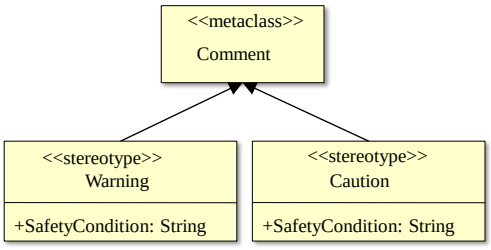


Figure 4 The stereotypes for S1000D 'Safety' elements

In case of the 'Procedure (<step>)', each sub-procedure (step) can be described with an UML Activity or an UML Action element. If a step is broken down into sub-steps, this step should be described with an UML Activity element that can include other activities that are sub-steps. Otherwise, a step can be described with an UML Action element. In addition, conditional flows of procedure should be described. First of all, to describe conditions, the type of condition should be classified. If the condition affects selecting resources that are required to accomplish a procedure, the condition can be described using an UML Comment element with the stereotype <<selection>> on the UML ObjectFlow. Otherwise, the condition affects selecting the next procedure (step), the condition can be described using an UML DecisionNode.

Finally, since the semantics of the 'Requirements after job completion (<closereqs>)' corresponds to the semantics of the pre-defined stereotype <<postcondition>> for the activity, the element (<closereqs>) can be described with stereotype <<postcondition>> for the activity that represents the whole procedure.

Table II shows that each rows' items of the left part (S1000D) can be described using the right part (UML)'s items .

TABLE II. SUMMARIES OF S1000D ELEMENTS VS. UML ELEMENTS

S1000D				UML		
Element	Subelement	Subelement	Attribute	Element	Stereotype	Tag
<dmtitle>	<techname>			Activity name		
<refs>	<refdm> or <reftp>			Comment	ExternalReference	
		<dmtitle> or <pubtitle>		Comment body		
		<avee> or <pubcode>				Code
		<issno> or <pubdate>				IssueNum
<xref>				Comment	InternalReference	
			xrefid	Comment body		
			xidtype			Type
<prelreqs>	<reqcond>			Activity	precondition	
	<reqcondm> or <reqcontp>			Comment	PreconditionRef	
	<supequip>			Class	SupportedEquipment	
	<supplies>			Class	Supply	
	<spares>			Class	Spare	
		<nomen>		Class name		
		<qty>				Quantity
			id			ID
			uom			UnitOfMeasure
<step>				Activity or Action		
	Conditional flow			DecisionNode or Comment with the stereotype <<selection>>		
<safety>	<warning>			Commnet	Warning	
	<caution>			Comment	Caution	
<closereqs>	<reqcond>			Activity	postcondition	
	<reqcondm> or <reqcontp>			Comment	PostconditionRef	

C. Transformation rules

This section describes the rules that transform from an UML model to a S1000D procedural data module XML file according to the proposed UML metamodel.

•The name of the outermost Activity can be transformed into the <techname> value of the <dmtitle> tag.

•The body of the stereotype <<precondition>> can be transformed into the <reqcond> values of the <prelreqs>/<reqconds> tag.

•The body of the stereotype <<postcondition>> can be transformed into the <reqcond> values of the <closereqs>/<reqconds> tag.

•In case of the stereotype <<PreconditionRef>> or <<PostconditionRef>>, the body of the stereotype <<PreconditionRef>> or <<PostconditionRef>> can be transformed into a value of the <reqcond> tag and if the tag <code> value of the linked <<ExternalReference>> element

 ISBN: 978-1-61208-230-1

is a data module code, it can be transformed as follows:

<reqconds><*reqcondm*><reqcond>….</reqcond>
<*reqdm*><*avee*>
….</*avee*></*reqdm*></*recondm*></reqconds>

Otherwise, it can be transformed as follows:

<reqconds><*reqcondtp*><reqcond>……</reqcond>
<*reqtp*>……</*reqtp*></*recondtp*></reqconds>

•In case of the stereotype <<ExternalReference>>, if the tag <code> value is a data module code, it can be transformed as follows:

<refs><*reftp*>……</*reftp*></refs>

Otherwise, it can be transformed as follows:

<refs><*refdm*>……</*refdm*></refs>

Also, if the <<ExternalReference>> element is linked with Activity or Action element, it can be transformed as follows:

<step~><para> …….
<*reftp*>……</*reftp*></para></step~>

or

<step~><para> …….
<*refdm*>……</*refdm*></para></step~>

•In case of the stereotype <<InternalReference>>, the part that the <<InternalReference>> element is used can be transformed as follows:

<step~><para>…<*xref xrefid*=….
xidtype=…>…</para></step~>

•In case of the stereotype <<Warning>> or <<Caution>>, if the element is linked with the outermost Activity, it can be transformed as follows:

<safety><*safecond*><*caution*>……</*caution*></*s*
afecond></safety>

or

<safety><*safecond*><*warning*>……</*warning*><
/*safecond*></safety>

If the element is linked with other Activity elements or Action elements, it can be transformed as follows:

<*step~*><*warning*>……</*warning*></*step~*>

•In case of Activity elements and Action elements, it can be transformed as follows:

<*step~*><*para*>*[Activity body|Action*
body]</*para*></*step~*>

If the source end of an incoming control flow is the UML DecisionNode, it can be transformed as follows:

<*step~*>
<*para*>*[guard condition of the control flow],*
[Activity body|Action body]</*para*>
</*step~*>

D. Structure of the S1000D Procedural Data Module Authoring Tool

This section shows the structure of the S1000D Procedural Data Module Authoring Tool. The tool supports automatically generating major contents of a S1000D procedural data module XML file by modeling an UML activity diagram.

Fig. 5 shows the relationships between subblocks of the tool and their artifacts. First of all, contents of primary S1000D procedural elements can be defined using a S1000D procedural data module editor even though not knowing the particular S1000D procedural data module structure or the UML diagram syntax. Then, an UML model generator generates an UML activity diagram on the basis of the primary S1000D procedural elements' contents. An UML model property editor allows a user to define additional specific properties of the UML activity diagram that don't have to be externally represented. A S1000D procedural data module XML file generator generates a S1000D procedural data module XML file on the basis of the UML activity diagram. A S1000D procedural data module XML file editor allows a user to define optional elements' contents.

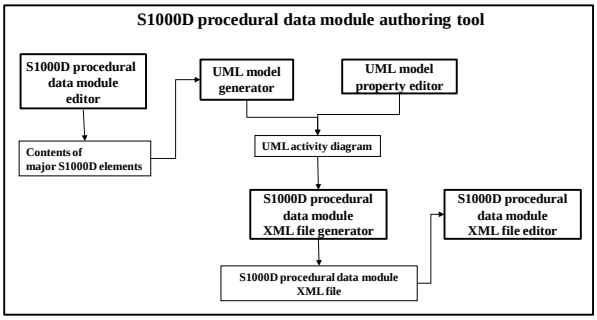


Figure 5. Subblocks of S1000D procedural data module authoring tool

IV. AN EXAMPLE

To address the practical applicability and features of our approach, we have chosen the procedure of lubricating the bicycle chain that is a typical example of the S1000D specification.

Fig. 6 shows the mark-up example of the S1000D procedural data module. In Fig. 6, the part (a) indicates the sub-element (<prelreqs>) of the 'Preliminary requirements' element. The part (b) of Fig. 6 indicates the sub-element (<supplies>) of the 'Preliminary requirements' element. The part (c) of Fig. 6 indicates the sub-element (<safety>) of the 'Preliminary requirements' element. The parts (d) ~ (k) of Fig. 6 indicate the 'Procedure (<step>)' elements.

Fig. 7 shows the UML activity diagram that describes the major elements of the mark-up example in Fig. 6. The part (a) of Fig. 6 can be described using the pre-defined stereotype <<precondition>> as shown in the part (a) of Fig. 7. The part (b) of Fig. 6 can be described using the UML Class element with the stereotype <<Supply>> as shown in the part (b) of Fig. 7. The part (c) of Fig. 6 can be described using the UML Comment element with the stereotype <<Warning>> as shown in the part (c) of Fig. 7. As shown in the part (d) of Fig. 7, the part (d) of Fig. 6 can be described using the UML Action element because it doesn't have any sub-step. On the other hand, since the part (e) has sub-steps, it can be described using the UML Activity element as shown in Fig. 7. In case of the part (g) of Fig. 7, since the

 ISBN: 978-1-61208-230-1

step needs the supported equipment (Floor covering), the step can be described using the UML action element linking with the stereotyped object (<<SupportedEquipment>>). The part (h) of Fig. 7 shows that the step contains the condition affects determining the next procedure (step). As shown in the part (h) of Fig. 7, the condition can be described using an UML DecisionNode. The part (i) of Fig. 7 shows that the step with the internal reference can be described using an UML action element linking with the stereotyped comment element (<<InternalReference>>). The part (k) of Fig. 7 shows that safety condition can be described using an UML Comment element with the stereotype <<Caution>>.

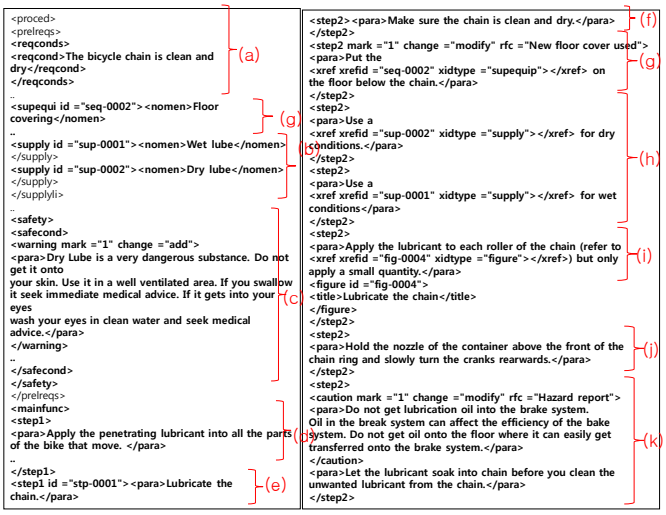


Figure 6 The S1000D mark-up example

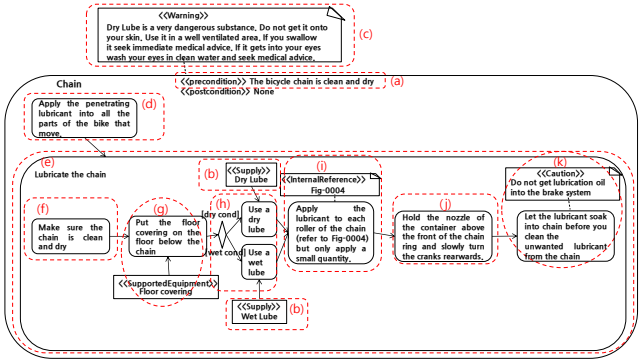


Figure 7 The example UML activity diagram

V. CONCLUSIONS

S1000D specification is developed as the standard XML format to describe technical publication. It is difficult to author XML documents without learning XML schemas or tags. Most technical publications for equipments are manuals that contain procedural information. In this respect, we proposed an approach to authoring S1000D procedural data modules using behavioral modeling method of UML. The proposed approach allows to easily produce S1000D data modules and to easily understand contents of the S1000D data modules without knowing XML DTD schema and tags. In the future, we will evaluate the proposed approach's substantiality by applying to more practical examples.

ACKNOWLEDGMENT

This work was supported by the IT R&D program of MKE/KEIT.[KI10038619, Development of Solution for Ship Safety Navigation based Maritime Ad-hoc Network].

REFERENCES

[1] ATA, ASD, and AIA, "S1000D: International Specification for Technical Publications Utilizing A Common Source Database", Issue 2.3, Air Transport Association, AeroSpace and Defence Industries Association of Europe, AreoSpace Industries Association[S], 2007.

[2] Shipdex Organization, http://www.shipdex.com [retrieved: Sep, 2012]

[3] W3C, "XML Tutorial", Available: http://www.w3schools.com/xml/default.asp [retrieved: Sep., 2012]

[4] PTC, Arbotext CSDB for S1000D, Available: http://www.ptc.com/product/arbortext/csdb-for-s1000d/ [retrieved: Sep., 2012]

[5] CORENA, CORENA S1000D solutions, Available: http://www.corena.com/what_we_offer/products/corena_s1000d/ [retrieved: Sep., 2012]

[6] Web-x, UltraCSDB S1000D suite, Available: http://www.webxsystems.com/. [retrieved: Sep., 2012]

[7] Siberlogic, SiberSafe S1000D Edition, http://www.siberlogic.com/index.html. [retrieved: Sep., 2012]

[8] Crowell Solutions, "S1000D introduction", http://www.crowsol.com/s1000d/s1000d-introduction [retrieved: Sep., 2012]

[9] CORENA, "Understanding S1000D business rules", CORENA white paper, 2010.

[10] Carlson, D. A., "Modeling XML Vocabularies with UML: Part 1", XML.com, Aug. 2001. Available: http://www.xml.com/pub/a/2001/08/22/uml.htm [retrieved: Sep., 2012]

[11] Carlson, D. A., "Modeling XML Vocabularies with UML: Part 2", XML.com, Sep., 2001. Available: http://www.xml.com/pub/a/2001/09/19/uml.html [retrieved: Sep., 2012]

[12] Carlson, D. A., "Modeling XML Vocabularies with UML: Part 3", XML.com, Oct., 2001. Available: http://www.xml.com/pub/a/2001/10/10/uml.html [retrieved: Sep., 2012]

[13] Booch, G., Christerson, M., Fuchs, M., and Koistinen, J., "UML for XML Schema Mapping Specification. Rational White Paper, Dec. 1999.

[14] Conrad, R., Scheffner, D., and Freytag. J., "XML conceptual modeling using UML", Proceedings of ER'2000, pp. 558-571, 2000.

[15] Farhad, J., "The UML Extension Mechanisms", Department of Computer Science, University College London, Dec., 2002.

 ISBN: 978-1-61208-230-1