ISBN 978-981-14-1684-2
Proceedings of 2019 the 9th International Workshop on Computer Science and Engineering
(WCSE 2019 SUMMER)
Hong Kong, 15-17 June, 2019, pp. 668-674
doi: 10.18178/wcse.2019.06.099

Design and Implementation of an XML Schema Based XML Data Editor

Dongyang Liang [1], Shasha Li [1], Jie Yu[1] and Bin Ji [1 +]

[1] College of Computer, National University of Defense Technology, Changsha China

Abstract. The collection, editing and processing of all kinds of equipment data is the core of all kinds of weapon equipment integrated support business information system. The scalability, stability and reliability of traditional systems are poor. This paper designs and implements an XML Schema based XML data editor. The editor takes a loosely coupled design, which is structurally divided into transformation from XML Schema into Java codes, profile generation tools, data editing interface, and data validation four modules, and the function of each module is defined. Finally, implement each function module and integrate them into the XML data editor. The validation results of the XML data editor shows that the XML data editor is available, reliable and stable.

Keywords: Equipment Integrated Support; XML Schema; XML; data editing; data validation

1. Introduction

Integrated equipment support takes an integrated approach to provide the necessary supply, repair and maintenance of the normal use of in-service equipment [1]. Since 1964, many countries in the world have successively promulgated relevant standards and carried out work related to weapons equipment support [2, 3]. The development of integrated equipment support needs to be supported by the integrated support systems [4]. These systems collect, edit, and process integrated equipment support data, including engineering drawings, design specifications, technical reports, analytical and experimental data, reliability and maintainability data, usage and maintenance manuals, supply and maintenance data, and various safeguards data [5], then apply the data to the system to digitize the equipment life cycle information.

As new high-tech products are continuously applied in equipment, in-service equipment will continue to be upgraded during use, which leads to frequent changes in the structure of integrated support data. Most of the information systems used in current integrated support system adopt a form-based development mode. The information system designs specific data editing forms, customizes specific data editing and processing programs, and database tables according to data structure. Therefore, you can only edit data for a specific structure. When the data changes, it takes a lot of manpower and material resources to redesign and develop data editing forms, data editing and processing programs, and database tables, which hinders the development of integrated equipment support. On the other hand, information systems based on the form-based development model are complex to develop, have a large amount of code, and are highly coupled. Therefore, such information systems can neither implement editing and processing of various structured data, nor can they be commonly used in various integrated security systems.

Our goal is to improve the integrated support capability of equipment and solve the problems in current integrated support systems that adopt form-based development mode, so this paper designs and implements an XML Schema [6, 7] based on XML data editor [8]. The editor is based on an XML Schema document describing the integrated support data structure, and edits and processes the data according to the XML

+ Corresponding author. Tel.: + 0731-88280171; fax: +0731-88280171.
E-mail address: jibn1990@163.com.

Schema document information. The structure of all kinds of equipment integrated support data can be described by XML Schema document. Therefore, the data editor can edit and process data of various structures, breaking the limitation of information system that adopts traditional form-based development mode

2. Design of XML Schema based XML Data Editor

This section gives the design architecture of the XML Schema based XML data editor. Starting from the XML Schema document (referred to as XSD) describing the integrated support data structure, the overall design of the XML data editor is shown in Figure 1.

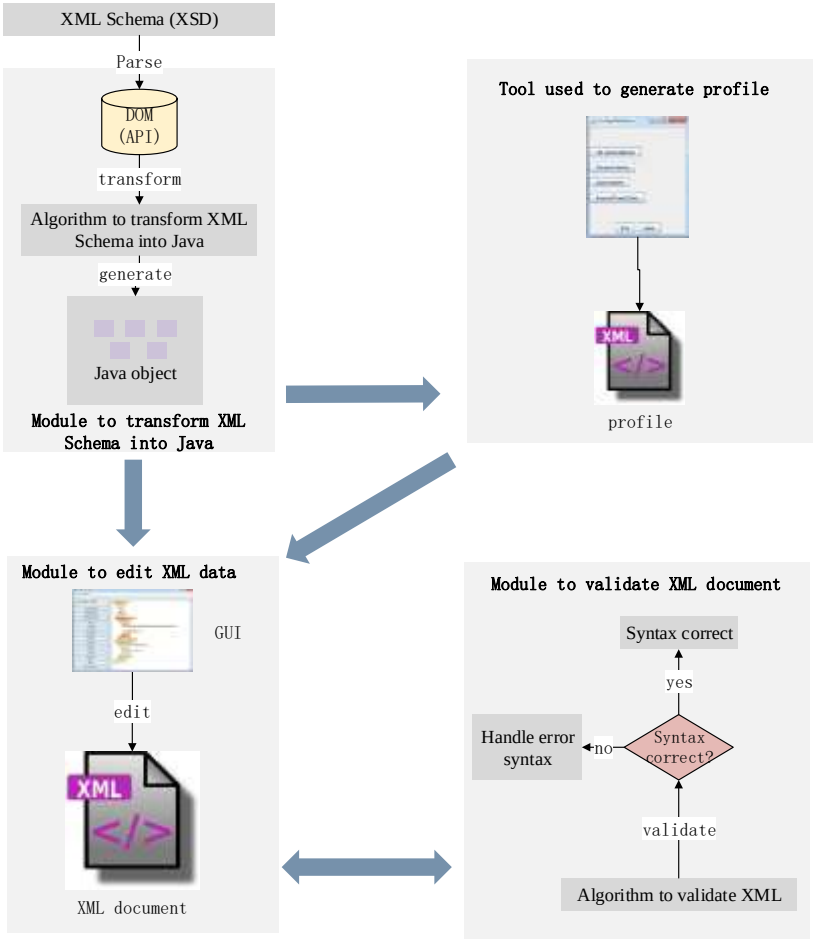

Fig. 1: Overall design of XML data editor

2.1. Transformation from XML schema into Java code

The XML Schema document contains data elements and constraint information about data elements. This information is represented by elements defined by XML Schema. The Java API currently used to obtain XML Schema document information is cumbersome to implement and is inefficient. Experts and scholars at home and abroad have done a lot of research on how to achieve efficient access to element information in XML Schema. Miu [9] and other scholars based on the Xerces2 parser to implement a Java application programming interface designed by MARKUP Technology to directly access the XML Schema component [10], to achieve direct access to the ten components of XML Schema. Wang [11] and other scholars based on JAXB to achieve XML Schema to Java class mapping. Other scholars have also proposed relevant research methods in the research process. With in-depth research we found that current methods are based on specific platforms or parsers and do not define methods for efficiently obtaining XML Schema document information.

In order to achieve efficient access to the information in the XML Schema, we propose an algorithm that transforms XML Schema into Java code, which is used to obtain an equivalent representation of XML Schema. The algorithm design can be found in the paper [12].

2.2. Profile generation tool

The profile generation tool is used to set the configuration information for the XML Schema and save the configuration information in the profile. The design and implementation of the module is based on the XML object obtained by the XML Schema transformation algorithm. The profile generation tool sets the following configuration information.

- Set the root element information: any XML file has a unique root element, the root element is the parent element of all elements in the XML file, and the type of root element is element.
- Set the encoding format of the XML file: The encoding format of the XML file can be: UTF-8, GB2312, UTF-16, and so on.
- Set the XML Schema namespace information: including SchemaLocation, noNamespaceSchema-Location, and so on.
- Set relevant information for all the elements in the XML Schema: including: the type of the element, the attribute of the element, and the like.
- Set Chinese names for all data elements in the XML Schema: the data element is the element and the attribute element.
- Set XML Schema information: This information is used to implement one-to-one correspondence between profiles and XML Schema.

After the configuration information is set, the profile generation tool saves the configuration information in the profile. The profile itself is an XML format file, and its form is as Table 1 shows.

Table 1: The form of profile format

Profile Foramt

```
<CONFIG>
  <INFO>
    <XML Schema_URL nom="URL path of XML Schema "/>
    <ROOT element="name of root element"/>
  </INFO>
  <REGISTER>
    <ENCODAGE>encoding information of XML file </ENCODAGE>
    <SCHEMALOCATION SchemaLocation="namespace info" noNamespace-SchemaLocation ="namespace info "/>
  </REGISTER>
  <ALL_ELEMENT>
    <ELEMENT name="element name" type="element type"/>
    ……
  </ALL_ELEMENT>
  <STRINGS language="Chinese code defined in ISO639 standard">
    <DESCRIPTION_CONFIG>XML Schema name</DESCRIPTION_CONFIG>
    <ELEMENT_STRING element="element name">
      <TITLE>Chinese name element </TITLE>
      <ATTRIBUTE_STRING attribute="attribute name of element's ">
        <TITLE>Chinese name of attribute </TITLE>
      </ATTRIBUTE_STRING>
      ……
    </ELEMENT_STRING>
    ……
  </STRINGS>
</CONFIG>
```

2.3. Data editing interface

The data editing interface implements editing of XML file. The design and implementation of the interface is based on the XML object obtained by the XML Schema transformation and the XML Schema profile. Among them, the Java object is used to obtain the constraint information of data elements and data elements in XML Schema document. The profile is used to obtain configuration information. The data editing interface is a data editing interface that implements editing data elements according to the constraint information of the data elements, and the functions are as follows.

- When creating a new XML file, first edit information of the root element.
- When editing the XML file, show the editing available elements in the mouse selection position.

- When editing an XML file, when an element is edited, the data editing interface automatically sets an icon indicating that the element starting label and closing label.
- If the element has attribute element as child element, the attribute child element is first edited before editing the element. For each attribute child element, it is prompted whether the child element is required. If the attribute child element has a fixed value, the element value of the attribute element is a fixed value. When editing the attribute element, it validates in real time whether the edited value of the attribute element conforms to the XML Schema's constraint on the data type of the attribute element.
- When editing an XML file, after editing an element, determine whether all the necessary sub-elements of the element have been edited, and iteratively judge the element's parent element, parent-parent element, until iterating to the XML file root element and give a hint in the form of an icon.
- For the edited element, the editing data of the element is displayed in a tree structure according to the nesting relationship of the element elements.
- The data editing interface provides auxiliary editing functions such as undo, redo, new, save, copy, and paste.

In addition, the data editing interface provides an interface to the XML file syntax validation function to integrate this functionality in the data editing interface. The simplified XML file editing process is shown in Figure 2.

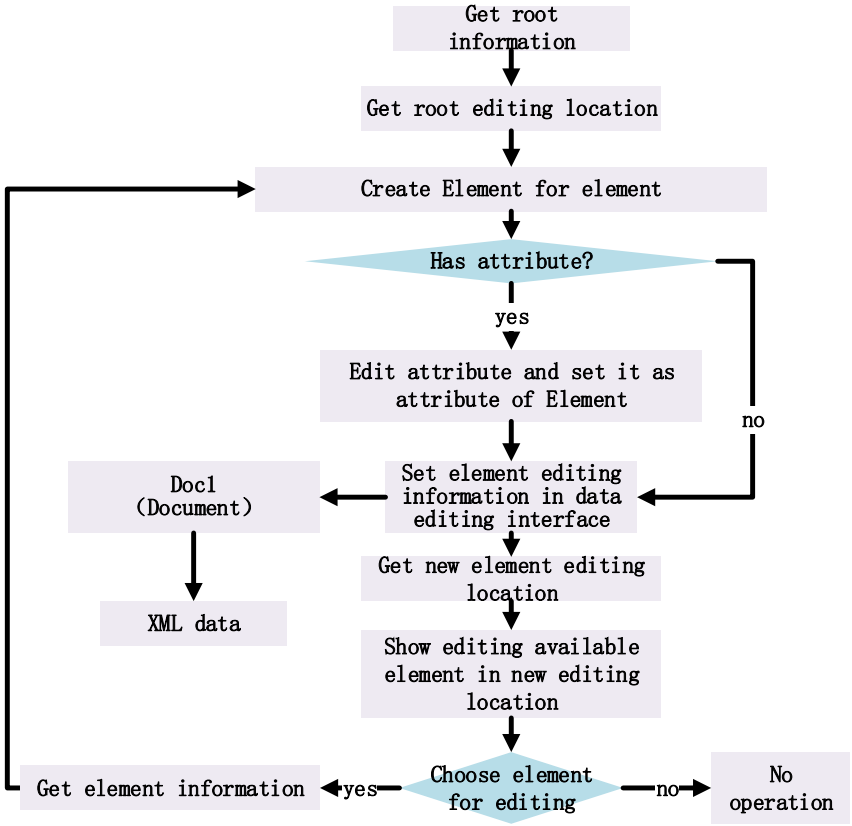

Fig. 2: XML files editing process

2.4. Validating XML document according to XML schema

Since XML Schema has become the recommendation standard of W3C, many scholars at home and abroad have studied the correctness of XML document syntax based on XML Schema. Yu [13] and other scholars designed and implemented an efficient XML Schema based XML validator based on the OnStAXParser [14] developed by the Chinese Academy of Sciences software. Wang [15] and other scholars first constructed their abstract model based on XML Schema, and then proposed a method based on XML Schema to validate XML files. These methods all implement the XML Schema based XML file syntax and can effectively output the XML file syntax error information. With in-depth research, we found that these methods either based on a specific platform or parser, or as a stand-alone system, with defects in cross-

platform portability, lightweight design, and ease of integration. In order to overcome the above shortcomings, we propose an algorithm based on XML Schema to validate XML files with good cross-platform portability, lightweight design and easy integration. The algorithm can be found in the paper [16].

3. Implementation of XML Schema based XML Data Editor

First implement the four function modules described in Chapter 2, and then integrate each function module to get the complete XML data editor. The integration of each functional module is mainly the integration of the XML Schema based XML validation and data editing interface.

The detailed implementation of transformation from XML Schema into Java code can be found in the paper [17]. The XML Schema based XML validation can be found in the paper [16]. The profile generation tool module and the data editing interface module are implemented in accordance with the detailed functions defined in Chapter 2, which is essentially an engineering implementation problem.

Finally, we integrate various functional modules to implement XML Schema based XML data editor. The final XML data editor interface is shown in Figure 3.

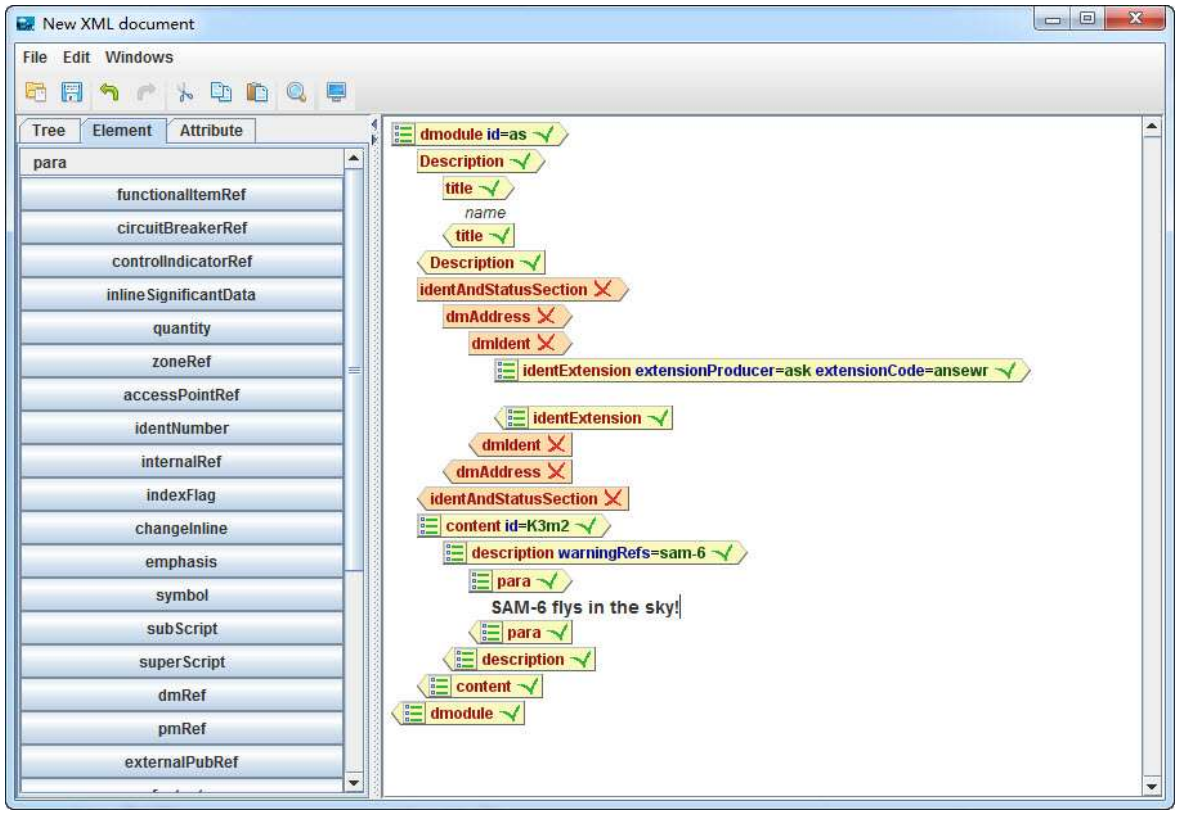

Fig. 3: XML data editing interface

4. Experiments

The experiment included validating the usability, reliability, and stability of the XML data editor. The XML Schema files use custom XML Schema documents and 19 XML Schema documents defined by the ATA/AIA/ASD S1000D Issue 4.2 [18] standard defines for 19 integrated support data. For each document, the verification process is as follows.

Firstly, use the profile generation tool to generate a profile for the XML Schema document.

Secondly, selects profiles in the data editing interface, creates new data editing, and validate the availability of the XML data editor.

Thirdly, judge whether each of the information prompting functions of the data editing interface is correctly executed, and execute the auxiliary editing functions and the syntax validation function defined in the data editing interface to validate the reliability and stability of the XML data editor. The validation results are shown in Table 2.

where the symbol "√" represents the corresponding function executes correctly. The validation results show that the XML data editor can edit and process the data for each XML Schema document, which validate the usability. Different XML Schema documents correspond to data of different structures, and the XML data editor is extensible for data of different structures. In the process of editing data, the XML data editor performs various functions such as auxiliary editing function, syntax validation function and so on, so the XML data editor is reliable and stable.

Table 2: XML data editor validation results

Function name	Validation result
new	√
close	√
save	√
undo	√
redo	√
copy	√
cut	√
paste	√
XML syntax validation	√
XML data display	√
Format transformation from XML to HTML	√
XML syntax pre-validation	√
Guidance of editable elements in new position	√
Guidance of non-negligible attribute	√
Real-time validation of attribute value	√
Real-time guidance of if element is editable or not	√
Real-time validation of if XML is well-formed or not	√

5. Conclusion

The information system based on the form-based development mode, which is adopted by the current equipment integrated support system, can only edit data of specific structure. This paper designs and implements a XML Schema based XML data editor, which solves the problems that exists in form-based development mode. Plenty of XML documents are taken to experiment to validate the availability, reliability, and stability of XML data editor. In the future work, we will continue to expand the functions of the XML data editor, and study the relational database storage of the date edited by the XML data editor.

6. References

[1] Y.L. Yu, R. Kang. Problems about basic theory and technology of equipment integrated support. *Journal of Academy of Armored Forced Engineering*. 24(6), pp.5-12 (2010).

[2] L. Song, J.H. Hu, W. Zhang, J. Yuan. Architecture of design of new submerged buoy integrated logistics support. *Ship Electronic Engineering*. 36(8), pp. 1-4 (2016).

[3] D. Xu. Research on Key Technologies of Equipment Comprehensive Support. *National University of Defense Technology*, 2006.

[4] Z.C. Xu. Equipment IETM development project overview. Beijing: *National Defense Industry Press*,2012.

[5] Y.K. Xu, J.J. Wu, Y.H. Yang, Z.Y. Pan. A study on informatization framework of all-life-cycle equipment logistic support. *Modern Radar*, 36(3), pp.1-5 (2014).

[6] W3C. W3C XML Schema Definition Language (XSD) 1.1 Part 1: Structures [EB/OL]. 2012.*https://www.w3.org/TR/xmlschema11-1/*.

[7] W3C. W3C XML Schema Definition Language (XSD) 1.1 Part 2: Datatypes [EB/OL]. 2012. *https://www.w3.org/TR/xmlschema11-2/.*

[8] W3C, eXtensible Markup Language (XML), *http://www.w3.org/ XML/*.

[9] W.H. Miao, Z.Y. Ni, J. Li. Implementing a Java API for Access to XML Schema. *Journal of Hebei University of*

Technology, 34(3), pp.106-109 (2005).

[10] Holstege M, Milowski R A, Tobin R. A Java API for the XML Schema Components [EB/OL]. *http://www.oracle.com/webfolde- r/technetwork/jsc/xml/ns/persistence/index.html.*

[11] Y. Wang, S.J. Xu. Research an Application of JAXB. *CD Technology*,2008(12), pp.30-31.

[12] B. Ji, J.J. Wu, H.J. Liu, L.R. Meng, W.Y. Peng. Research on XML Schema Transformation Algorithm. *Chinese Automation Congress (CAC)*, 2017.

[13] S. Yu, D.L. Cao, B.J. Dai.Research on high performance implementation of XML validation. *Computer Engineering and Design*, 29(4), pp.937-941 (2008).

[14] X. Ren, D.L. Cao, B.H. Jin. An efficient StAX based XML parser[C].*Proceedings of the 11th Joint International Computer Conference (JICC)*,2005.

[15] W.L. Wang, Q. Shi, Q.J. Cao. A method for XML document schema validation with abstract XML Schema model. *Computer Applications and Software*, 24(3), pp. 41-43 (2007).

[16] J.J. Wu, B. Ji, Q.G. Chen, Y.H. Yang. Research on XML Schema based XML Document Validation. *Modern Electronics Technique*, 2018.

[17] B. Ji, J.J. Wu, Y.H. Xie, L.R. Meng, Y.Q. Ma. An Algorithm to Transform XML Schema into Java codes. *Chinese Automation Congress (CAC)*, 2017.

[18] S1000D Issue 4.2, International Specification for Technical Publications Using a Common Source Database, 2016.