REVERSE ENGINEERING: FROM XML TO UML FOR GENERATION OF SOFTWARE REQUIREMENT SPECIFICATION

Hannani Aman
Faculty of Computer Science and Information Technology
Universiti Tun Hussein Onn Malaysia, Johor, Malaysia
hanani@uthm.edu.my

Rosziati Ibrahim
Faculty of Computer Science and Information Technology
Universiti Tun Hussein Onn Malaysia, Johor, Malaysia
rosziati@uthm.edu.my

Abstract—Reverse Engineering (RE) is a method to extract requirement information application from XML at higher level of abstraction. At present, reverse engineering of XML focus on transformation XML to conceptual model, which is UML. The transformation result is presented in variety of diagrams. The transformation result is beneficial for developer and designer tracking changes in analysis phase. However, it does not show changes from the requirements view. Therefore, this paper shows the framework of reverse engineering from XML to UML for generating software requirement specification. In this work, the related work of reverse engineering XML is clear but none are extending to requirements level. This paper also proposes research method to extend transformation XML to SRS.

Keywords—XML; reverse engineering; transformation;

I. INTRODUCTION

Nowadays, XML is a well known technology in internet services. The flexibility in data sharing makes it as a de facto of exchanging standard between web services. Web developer can gain data from any web services easily within other XML schema and share their document in trust. Before exchanging documents between services, xml documents need to be validated by XML schema provided. This will ensure that the exchange is gain the same trust as the provider of the document. An example of a mall web service that need to receive supplier data from supplier web service. Agreements between two services need to be achieved in order they exchange data in xml format as a communication between services. Therefore, both data xml document need to be validated with XML schema provided by supplier services before a mall service is being used.

The XML schema is a set of rules used to define the data types of data exchanges. This XML schema is used as document to validate data exchanged at development level. In object oriented based system, requirements of a system is specified in the form of UML model. A UML model may consist of different UML diagrams (eg: class diagram, sequence diagram, activity diagram and others). Each UML diagrams can be represented as a XML document to make it exchangeable to other services for example data supplier in XML format. This is called XML forward engineering. In XML forward engineering research, focus research is most on conceptual level to development level, which is UML to XML schema ([1],[2],[3],[4],[5]). Some of transformation tool based on forward engineering is developed too, such as Unisys Rose XML Tools [6] and Rational Software Architecture [7]. However, there is less research in reverse engineering of XML([8],[9],[10]). The research focuses most in transformation from XML to UML. This kind of transformation are called data reengineering is the most important data intensive application. In modern software development methodology, data reengineering is very crucial [11].

This paper is organized as follows: Section 2 is our motivation in this research. Section 3 is an overview of XML, UML and SRS. Section 4 is related work about reverse engineering in XML to UML. Section 5 is the proposed research method to generate SRS from XML schema. Finally, Section 6 is the conclusion of this paper.

II. MOTIVATION

Modern software development methodology such as Agile methodology and Extreme Programming (XP) methodology, both method emphasizing processes developing application in a short time boxing and less cost of changes in requirement management [12]. For this reason, reverse engineering tool is the best way to capture the requirement changes management in a small cost. Reverse engineering is a way to manage version and changes in an iterative (or Agile) fashion today. In a way that requirements will have versions associated with its application release and able to track back and identify any impact changes ensuring changes control and scope creeping. It also can show or track the status of development progress from the requirement view in a short time. Besides, the requirement changes that gain from application (data view) helps better artifact to communicate
with the stakeholder in showing the progress development of that application.

The important reason focusing in reverse engineering, as if the process needed to recheck XML schema developed with the earlier UML diagram created. Obviously, no UML diagram documented, if the developer has not gone through formal development phase. In both situations, requirement documentation is very important to tell the progress of project development in the track. In order to keep the documentation of changes in UML specification more proper and manageable, we suggest the reverse engineering process to extend until Software Requirement Specification.

III. OVERVIEW OF XML, UML AND SRS

A. Extensible Markup Language (XML)

XML is currently an international standard for data representation and exchange in the Internet environment. This markup language is used by the developer for their web application because of the data communication flexibility exchange across different platform. Furthermore, the flexible feature which is free tagging makes it more popular used. XML document is the level which different platform application used to exchange data. This XML document presentation is structured by XML schema also called as XML data type definition. In software development practice, a design phase needs to come with design artifact before XML implementation is developed

B. Unified Modeling Language (UML)

UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The Unified Modeling Language includes a set of graphic notation techniques to create visual models of object-oriented software-intensive systems. Unified Modeling Language is used to specify, visualize, modify, construct and document the artifacts of an object-oriented software-intensive system under development. Unified Modeling Language is the most practical language used nowadays to capture the XML features.

C. Software Requirement Specification (SRS)

SRS is a complete description of the behavior of a system to be developed and may include a set of use cases that describe interactions the users will have with the software. In addition it also contains non-functional requirements. Non-functional requirements impose constraints on the design or implementation (such as performance engineering requirements, quality standards, or design constraints).

The software requirements specification enlists all necessary requirements that are required for the project development. To derive the requirements we need to have clear and thorough understanding of the products to be developed. This is prepared after detailed communications with the project team and customer.

IV. RELATED WORK

XML schemas present logical-level schemas, but not conceptual schemas that describe their contents, and systems analysts and designers have found it difficult to communicate with co-workers using XML schemas due to their textual nature and syntax. In order to tackle this problem, researchers have proposed the idea of reverse engineering logical-level XML schemas to conceptual-level UML diagrams to facilitate easier understanding and to provide communication tools (i.e. resultant UML diagrams) for the systems being developed [13].

Yu and Steele [13] have done a survey regarding reverse engineering for UML and conclude a few issues as stated in Table 1.

TABLE 1. YU AND STEELE [13]

<table>
<thead>
<tr>
<th>No</th>
<th>Issue</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A standard optimal approach conducted on RE of XML to UML diagram has not yet reached.</td>
</tr>
<tr>
<td>2</td>
<td>Level of complexity of the generated UML diagram versus the amount of textual information to preserve from XML schema to the UML diagram.</td>
</tr>
<tr>
<td>3</td>
<td>More specific XML constructs (eg: virtual and final classes, restriction derivations, etc.) need to be incorporated into the resultant UML diagram.</td>
</tr>
</tbody>
</table>

Based from Table 1, the issue is on Data Type Definition XML only, which is the old type of XML schema. Nowadays, the most popular XML schema language is XML Schema instead of other such as Schematron and RELAX NG[14].

In 2010, Necasky [9] has done another survey regarding XML schema reverse engineering research. The aim of the reverse engineering process is to create a model from existing XML schemas (or connect a new schema to an existing model) and once it’s received, it may be used for management of the evolution of their schema’s set. Their discussion leads the result to focus on the need of easy management of large numbers of XML data sources and their integration. Their result is compared with a few significant criteria stated in Table 2.

Target Model is a criteria that recognize the level in which the RE method is used, whether it used in Platform Specific Model(PSM) which use UML or Platform Independent Model(PIM) which its independent of any model used. A second criterion is how many numbers of XML schema can be mapped to a model involves. The third criteria is which XML schema languages supported the RE method, whether the Data Type Definition or XML Schema. The fourth criteria are whether the mapping process generates new XML schema or existing XML schema. The following criteria is the level user involvement, which the reverse transformation is semi automatic that need user involvement or automatic, no need user involvement. Finally, the last criteria, evolution support whether it can be integrated with a new schema after the earlier schema generated.
TABLE 2. COMPARISON RE XML RESEARCH CRITERIA

<table>
<thead>
<tr>
<th>A</th>
<th>Target Model</th>
<th>B</th>
<th>Number of schemas supported by the model</th>
<th>C</th>
<th>XML schema languages supported</th>
<th>D</th>
<th>Mapping to an existing model</th>
<th>E</th>
<th>Level of user involvement</th>
<th>F</th>
<th>Evolution support</th>
</tr>
</thead>
<tbody>
<tr>
<td>M</td>
<td>FS</td>
<td>S</td>
<td>LS</td>
<td>A</td>
<td>Mt</td>
<td>E</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>[15] FS M</td>
<td>One</td>
<td>DTD</td>
<td>Yes</td>
<td>New</td>
<td>No</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>[16] FS M</td>
<td>One</td>
<td>DTD</td>
<td>Yes</td>
<td>New</td>
<td>No</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>[17] FS M</td>
<td>Multiple</td>
<td>DTD</td>
<td>No</td>
<td>New, Existing</td>
<td>No</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>[18] FS M</td>
<td>Multiple</td>
<td>DTD</td>
<td>Yes</td>
<td>New, Existing</td>
<td>No</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>[19] FS M</td>
<td>One</td>
<td>XSD</td>
<td>Yes</td>
<td>New, Existing</td>
<td>No</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>[20] PIM</td>
<td>Multiple</td>
<td>Any</td>
<td>No</td>
<td>New, Existing</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Klimek et al. [9] says that challenge in XML to UML transformation is accuracy of the transformation in the different nature, syntax and structure of UML and XML itself. Direct transformation may lose a lot of important information/data. In designing a UML class diagram, a few views are involved. Different view will generate different XML schema/document. Another challenge in this transformation is in integrating different view from different XML schema to one UML class diagram. Klimek et al. [20] using Model Driven Architecture proposed a precise framework to map XML to UML efficiently.

RE in data engineering is not only at the conceptual level. It needs to capture the versioning in requirement level and able to track back any impact changes from XML to requirements specification level. For this reason, the RE XML research is lacking of reverse transformation of XML to capture requirements in documented manner. Therefore, it needs RE method for data engineering for generating Software Requirement Specification. This may save cost and time in generating SRS during the maintenance phase.

V. RESEARCH METHOD

In this section, we discuss the proposed transformation steps from XML schema to SRS text using UML specification. They showed in Fig. 1. The proposed transformation model approach consists of two stages:

1. The mapping XML Schema to UML Specification
   - The first stage is capturing the XML characteristic such as namespace, element and attribute. The characteristic then will be mapped into UML specification such as class, attribute and other UML characteristics.

2. Mapping UML Specification to SRS text
   - From the UML specification results, requirement characteristic need to be recognized. The recognized requirement such as functional requirement will be organize in text files.

![Figure 1. Transformation Steps](image-url)

For Step 1, Table 4 will be used as the mapping rules to generate the UML Specification. As an example, name element in XML schema will generates UML class.

TABLE 4. MAPS XML TO UML SPECIFICATION[4]

<table>
<thead>
<tr>
<th>UML Specification</th>
<th>XML schema tag</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class</td>
<td>element, complex type, with ID attribute and key</td>
</tr>
<tr>
<td>abstract class</td>
<td>Abstract element and complex type, with ID attribute</td>
</tr>
<tr>
<td>attribute</td>
<td>Sub element of the corresponding class complex type</td>
</tr>
<tr>
<td>stereotype</td>
<td>Attribute of the corresponding element</td>
</tr>
<tr>
<td>Association</td>
<td>Referencing the associated class and keyref for type safety(key/keyref reference)</td>
</tr>
<tr>
<td>aggregation</td>
<td>Association class and an additional IDREF references to the association class element and a keyref in the corresponding reference elements in the associated classes.</td>
</tr>
<tr>
<td>qualified association</td>
<td>Extension of the reference element, keyref and key of the target class with the qualified attributes</td>
</tr>
<tr>
<td>composition</td>
<td>Reference element, with subordinated class element(hierarchy relationship)</td>
</tr>
<tr>
<td>generalization</td>
<td>Complex type of the subclass is defined as an extension of the complex type of the superclass</td>
</tr>
</tbody>
</table>
The UML specification will be generated from XML schema tag. The algorithm used in this generation is based on bidirectional formula given in [4]. From XML Schema, XML tagging is selected based on XML schema tag in Table 4. XML tag that has been focus is element, complexType, simpleType, IDref, attribute and more. The tag name is pick out and put into temporary table that has been classified as UML specification column in Table 4 based on class diagram and use case specification.

While for Step 2, the transformation from XML to SRS is using the mapping method to retain all the data specification. Similar to the work done in [21], the SRS will be generated using text file. UML specifications generated from Step 1 are classified in specific order of wordings to make a simple but precise requirement text. The requirement text will be organized such as in Requirement Statement Text. The generated text file based is for ensuring the STS is readable and reusable for the developers.

The project framework aims to generate a SRS text file from an XML schema. In order to get the target text, the XML schema must generate UML specification as intermediate steps as in Fig. 2. The reverse engineering method will try to capture specification in XML schema to and retain it in UML specification and finally put the specification in a standard manner in SRS document.

VI. CASE STUDY

We use an Inventory Tracking System that specializes in transporting shipments as case study for the proposed framework. The system seeks a mechanism to track shipments from its headquarters to its regional offices. When products are shipped, the head office sends information electronically in XML about shipping. Once the shipment has reached its destination, the confirmation is electronically sent back to headquarters.

As shown in Fig. 3, the XML schema (ShippingOrder.xsd) is developed by developer.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" attributeFormDefault="qualified"
    elementFormDefault="unqualified">
  <xs:include schemaLocation="DataTypes.xsd"/>
  <xs:element name="shippingOrder">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="order" type="Order"/>
      </xs:sequence>
      <xs:attribute name="shippingId" type="int!/>
      <xs:attribute name="origin" type="Origin"/>
      <xs:attribute name="destination" type="Destination"/>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Figure 3: ShippingOrder.xsd

By using mapping specification rules as in Table 4, UML specification will generated. In ShippingOrder schema, element name shippingOrder is transform as class in UML specification. Element name shippingId, origin, destination and
order in an attribute in UML specification. Table 5 shows the results of UML specification generated.

TABLE 5. UML SPECIFICATION GENERATED

<table>
<thead>
<tr>
<th>UML Specification</th>
<th>UML metadata</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Attribute of Origin</td>
<td>attribute</td>
</tr>
<tr>
<td>Item</td>
<td>Attribute of order</td>
<td>Item</td>
</tr>
<tr>
<td>Order</td>
<td>Attribute</td>
<td>Order</td>
</tr>
<tr>
<td>Destination</td>
<td>Attribute</td>
<td>Destination</td>
</tr>
<tr>
<td>shippingId</td>
<td>Attribute</td>
<td>int</td>
</tr>
<tr>
<td>Origin</td>
<td>Attribute</td>
<td>Origin</td>
</tr>
<tr>
<td>ShippingOrder</td>
<td>Class</td>
<td>class</td>
</tr>
</tbody>
</table>

Next, from the UML specification generated, the specification will be organized in the SRS template as shown in Fig. 4. The class will be appeared in section 3.2 in SRS. Meanwhile, attribute will be arranged according to their classes. In 3.2.1.2, classes need to be in natural language form to purposely show the functional requirement. Finally, the information in XML is retained until it will be organized in SRS template as shown in Fig. 4.

Figure 4. SRS template organized by object.

VII. CONCLUSION

In this paper, we have presented a propose method of reverse engineering from XML to generate Software Requirement Specification in a documented manner. Since the transformation XML to UML has undergo an intensive research, extending the XML to SRS will be beneficial to project development activity in modern methodologies for saves time and development cost. This framework for reversing from XML to generate Software Requirement Specification has been established. The future work includes the implementation of the environment based on this framework.

REFERENCES


