THE SOFTWARE REQUIREMENTS FRAMEWORK FOR DOCUMENT CHANGES USING REVERSE ENGINERING APPROACH

Hannani Aman and Rosziati Ibrahim

University of Tun Hussein Onn Malaysia, Malaysia, {hanani, rosziati@uthm.edu.my}

ABSTRACT. Document changes are center of focus in maintenance phase during software development life cycle (SDLC). In Agile development process, document maintenance become crucial as the focus is on technical maintenance. This paper presents a software requirements framework for document changes using reverse engineering approach. The framework is called XML Document Tracker (XML_DocTracker). Based on the framework, a tool is developed to automate the generation of software requirement document, due to requirements document changes. The approach focuses on XML application changes. Class diagram is used to capture changes semantically rather that syntax of code. Then, the changes are kept as version factors for generating SRDdue to requirements document change. This reverse approach is better because in agile development methodology, developer shall change the codes directly without making any changes in requirements. The advantage of using the tool is to ease the agile developer task in document generation.

Keywords: reverse engineering approach, document changes, framework

INTRODUCTION

A change on application during development or maintenance is unavoidable. This is due to additional functions, the occurrence of errors, changes in requirements and many more. Any changes on application need to be captured in documented way. Since agile developer documentation is very tedious and laborious, many approaches has been introduced to assist documentation in many domains of developments (Kaisti et al., 2013). Therefore, we proposed an alternative approach that will assist the agile developers generating document changes.

W3C has introduced XML as a language that support a variety of applications, for diverse applications such as browsing, content analysis, standards commerce communications and interchange content. It has to share structured data among various diverse software systems and application domain using its universal format. Specification and standards of XML application have been described by XML schema. XML schema has been adapted in various industries. Each industry has their working group to establish their specification called XML based standard. As working group evolves, a few version of XML schema have been released. For instance, electronic Business of XML (ebXML) on electronic business by OASIS(ebXML Specifications, 2006) and Rich Site Summary for web interchanged data from Netscape (Rich Site Summary (RSS) by Netscape,2014). Each of this release is called version.

A single change in XML schema to the latest version may influence user's requirements document that had been used in the previous version. User needs to detect any changes in the XML document manually. In order to recognize the requirement that changes, he needs to understand the changes in the XML schema. Since XML Schema is text-based format, it is hard for the user to detect, understand and rezognize any changes made in the requirement. Furthermore, viewing version changes need to be capture in the document in order to maintain new document version.

In this paper, we present a framework on XML schema versioning for the following motivations. Firstly, we are motivated to show on how reverse approach is used in impacting XML schema on the requirements document. Secondly, to show on how to use the conceptual model in viewing changes of requirements when XML Schema evolves. In short, this research proposes a reverse method to study the impact of XML schema changes on the conceptual model and from there to generate the requirements document changes.

RELATED WORK

Document changes in XML application starts when (Tan & Goh, 2004) suggest a need to highlight changes and differences between a preceding version or a variant against original standard of XML. For the purpose of maintenance, new functionalities on compatibility establishment are proposed. As the XML standard evolves, focus on updated schema effect were proposed using new language called eXupdate (Cavalieri, Guerrini, & Mesiti, 2011). While keeping primitive changes of XML files has been introduced by (Brahmia, Grandi, Oliboni, & Bouaziz, 2012), these changes only focus on syntax level of XML application in which document capture is too technical to be conducted.

Researchers (Klettke, 2007; Klímek, Maly, Mlynkova, & Necasky, 2012; Polák, Necasky, & Holubová, 2013) have seen the bigger picture of keeping XML Schema changes in semantic level which using the conceptual model. They have called it an evolution – a change in the domain is made once in the conceptual diagram and then is propagated to the affected XML schemas. The changes parameters used are Additional, Remove and Rename of element and attribute. Element and attribute of changes coves at platform specific model which represent specific of conceptual model (Klímek et al., 2012). These researchers have highlighted the changes propagated to XML application using colors.

However, changes need to be captured in a documented and structured way rather that highlighting changes in the conceptual model or in the schema form. Thus we suggest these changes are captured in a requirement document. Furthermore, forward direction does not show the reality of XML changes impact on requirement document. Therefore, a reverse approach is proposed in this work.

XML_DOCTRACKER FRAMEWORK

The software requirements framework for document changes is shown in Figure 1. Based on Figure 1, the proposed framework is divided into three main steps- a reverse method, changes detection and requirement document changes. For Step 1, transformation activities on XML Schema to generate Class Diagram. The transformation rules transforms XML Schema A to generates Class Diagram A. XML Schema A evolves to XML Schema A' when there are changes in XML Schema A. XML Schema A' will repeat Step 1 of reverse transformation rules to generate new Class Diagram denoted as Class Diagram A'.

In Step 2, versioning algorithm are use to detect differences between these class diagrams and marked as called traceability link. The traceability link between Class Diagram A and

Class Diagram A' is recognized as version factor. Finally in step 3, a version factor generates new version of Software Requirements document.

The step for Reverse Transformation Rules are explained in detail in Transformation Rules section. The step for Traceability Link is explained in detail in Versioning Rules section. Finally, the step for version factor for generating SRD is explained in details in Versioning Factors section.

THE TRANSFORMATION RULES

XML Schema is the main resource of XML application which will be used as the input of the framework. XML schema A and XML Schema A' consists of XML components(type definitions, element declarations, attribute declarations, attribute group definition and model group definition). A single XML schema may consist a few pages text description of a XML document. When a new version is introduced, it is hardly read and recognizes any changes. It needs to be transformed into a conceptual diagram to view the semantic changes. The needs in reversing the development process lead to this transformation rules.

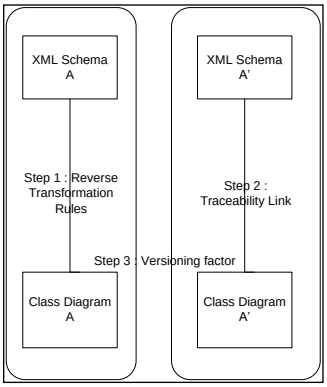

Figure 1. XML_DocTracker Framework

Informal transformation rules may lead to ambiguous results because of individual interpretation of words. In order to ensure the correctness and accuracy of the reverse transformation rules, one to one mapping transformation using formal method has been proposed. Formalization of transformation rules generates a class diagram denoted as Class Diagram A. XML Schema A' are a new version of XML Schema A. XML Schema A' needs to be transformed using Transformation Rules to have an equivalent compared with the preceding schema.This will generate Class Diagram A'. Details about these formal transformation rules can be found in our previous work (Aman & Ibrahim, 2014). The simplified processes of this step are shown in Figure 2.

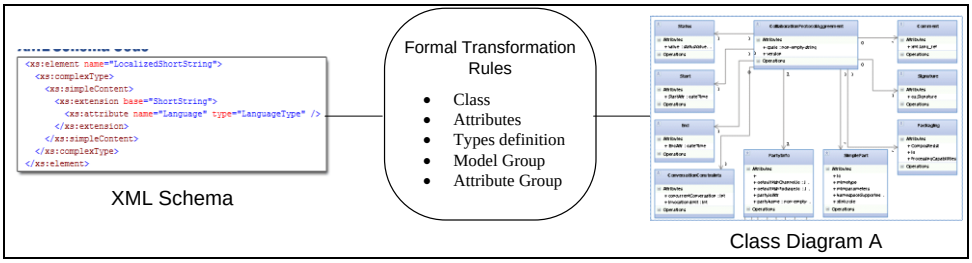

Figure 2. Reverse Transformation Process

THE VERSIONING RULES

The modification is generated on 3 atomic operations classified on additional, removal and rename operations of XML basic components (element, attribute, type definition, model group and attribute group). An additional change involves in adding any element or attribute in the schema wheares a removal change involves any removing element of XML components. Meanwhile, rename operation involves modification element's name to a new name with the same attributes. These are atomic operation of an evolution of XML schema. Combination operation involves combinations of these 3 atomic operations. These combination will generate new semantic changes such as migration of element to another element where involves removal and additional at different level of class diagram.

Differences between these two class diagrams from the preceding class diagram are linked to newer class diagram. This link is called as traceability link due to the changes occurs.

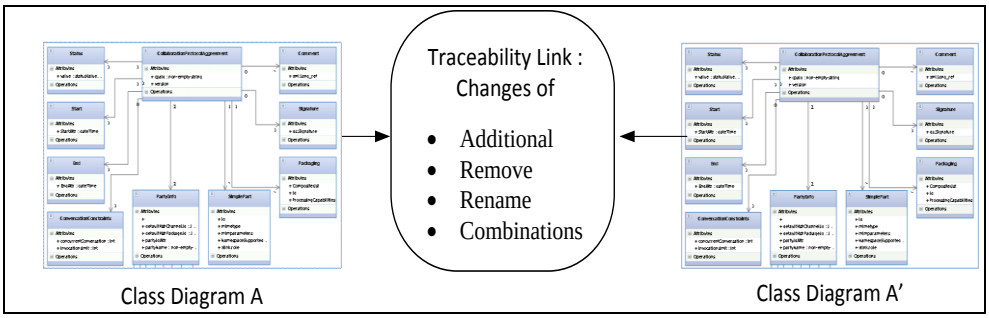

Figure 3. Traceability Link Process

Our work proposed a new combination of atomic operations. It involves additional, removal and modification of element where the purpose of combination shows the evolution of an element. Based on proposed changes, Versioning Rules (VR) have been introduced as follows in Table 1:

Table 1. Versioning Rules (VR)

VR	Description
VR1	Each class of both class diagrams has been compared - level by level of the diagrams.
VR2	Each Added classes and Removed classes are compared.
VR3	Classes which have the same attribute and same relationship are tested with string similarity. High result of similarity tests the changes of new versions of class diagram.

VERSONING FACTORS

From the traceability link, version factors are generated. It may consists of additional, removal, rename and the combination of them. These factors are then being used for the transformation of a new version of requirement changes as a new version of software requirements document.

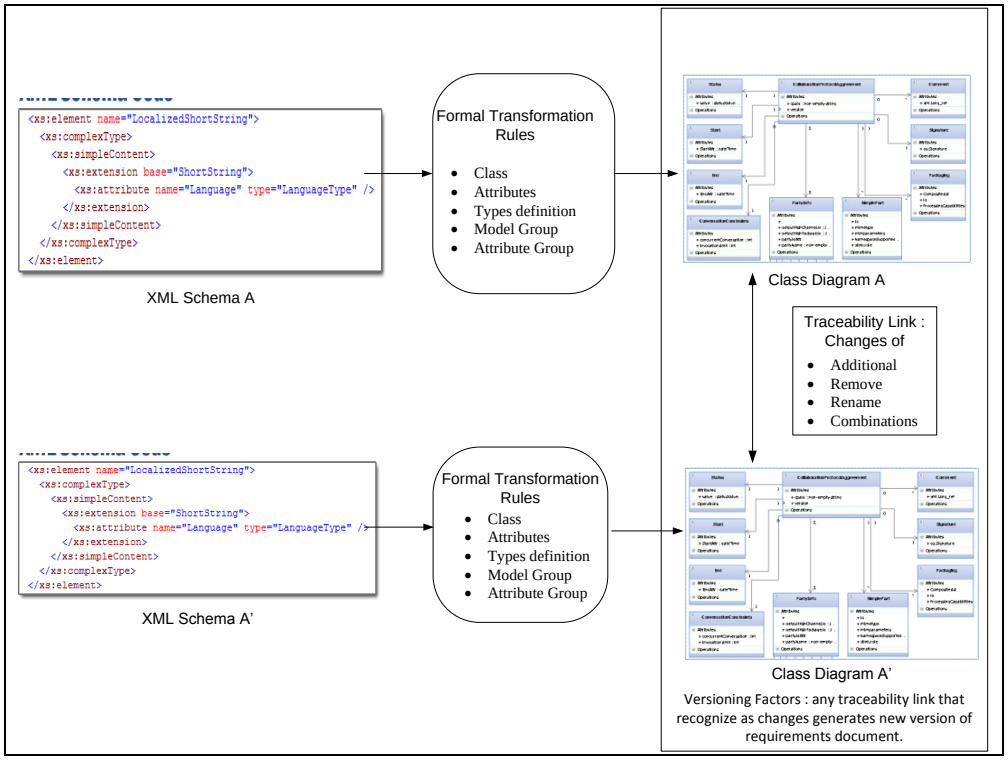

Figure 4. Versioning Factors Process

CONCLUSION

Agile developer maintains application with versions. These versions need to be documented properly. XML Schema is one of the components in application that also need to be documented due to changes. Therefore, this paper proposed a framework using a reverse engineering approach to detect document changes on XML schema.

In our work, the approach contains 2 important rules which are transformation rules for reverse approach and versioning rules to detect changes on semantic of XML application. Therefore, the proposed approach is an alternative approach to assist agile developer in XML application domain to generate document changes. The version factors in document changes may be used as a control document for future work.

ACKNOWLEDGMENTS

The authors would like to thanks Malaysian Ministry of Education (MoE) for supporting this research under the Fundamental Research Grant Schema (FRGS).

REFERENCES

Aman, H., & Ibrahim, R. (2014). Formalization of Transformation Rules from XML Schema to UML Class Diagram. *International Journal of Software Engineering and Its Application*, *8*(12), 75–90.

Brahmia, Z., Grandi, F., Oliboni, B., & Bouaziz, R. (2012). Versioning of Conventional Schema in the tXSchema Framework. *2012 Eighth International Conference on Signal Image Technology and Internet Based Systems*, 510–518. doi:10.1109/SITIS.2012.153

Cavalieri, F., Guerrini, G., & Mesiti, M. (2011). Updates on XML documents and schemas. *2011 IEEE 27th International Conference on Data Engineering Workshops*, 308–311. doi:10.1109/ICDEW.2011.5767672

EbXML Specifications. (2006). Retrieved from http://www.ebxml.org/specs/index.htm

Kaisti, M., Rantala, V., Mujunen, T., Hyrynsalmi, S., Könnölä, K., Mäkilä, T., & Lehtonen, T. (2013). Agile methods for embedded systems development - a literature review and a mapping study. *EURASIP Journal on Embedded Systems*, *2013*(1), 15. doi:10.1186/1687-3963-2013-15

Klettke, M. (2007). Conceptual XML Schema Evolution. In *BTW Workshop "Model Management und Metadaten-Verwaltung", Aachen. 2007*.

Klímek, J., Maly, J., Mlynkova, I., & Necasky, M. (2012). Evolution and change management of XML based system. *Journal of Systems and Software*, *85*, 683–707. doi:10.1016/j.jss.2011.09.038

Polák, M., Necasky, M., & Holubová, I. (2013). DaemonX : Design , Adaptation , Evolution , and Management of Native XML (and More Other) Formats. In *IIWAS '13: Proceedings of International Conference on Information Integration and Web-based Applications & Services*, 484. doi:10.1145/2539150.2539159

Rich Site Summary (RSS) by Netscape. (2014). Retrieved from http://www.sigmaxi.org/programs/international/news.082004.shtml

Tan, M., & Goh, A. (2004). Keeping Pace with Evolving XML-Based Specifications, 280–288.