Effort Reduction by Automatic Code Generation

Sukhvir Singh
Asst. Professor, Department of Computer Science & Engineering
NC College of Engineering
Israna, Panipat
boora_s@yahoo.com

Neeraj Kumar
Research scholar Department of Computer Science & Engineering
NC College of Engineering
Israna, Panipat
neerajadonis@gmail.com

Abstract—**The emergence of Unified Modelling Language (UML) as a standard for modelling systems has encouraged the use of automated software tools that facilitate the development process from analysis through coding. In our approach UML class diagram is used to generate XML schema and generated xml schema is used for code generation.**

In our approach UML class diagram is used to generate XML schema and generated xml schema is used for code generation. We have implemented a system for XML Schema generation using .net platform. .net is helpful in fastly generating the tool. A tool named JIBX is used for code generation which is developed by IBM and also a open source tool.

Keywords- UML, XML Schema, Code generation, XSD, modelling, transformation

I. INTRODUCTION

Design through modelling, which is a norm in the engineering domains, enforces a careful investigation of the structure, behaviour and architecture of a system in the early stages of development and promotes documentation and reuse. This activity generates a number of models to represent different perspectives of a system and/or at different levels of detail. If not done with proper care, the implications of incorrect designs could be minor to severe. On the one hand, it could result in minor defects in the final implementation of the system, which cause frustrations for the user, but do not affect the functionality in a major way. On the other hand, it could lead to costly problems, such as failures or worse yet, harmful to its users. In software development, it has been studied that it costs ten times as much to fix a problem during implementation that which could have been corrected in the design phase of the development life-cycle.

XML was originally envisaged as a language for defining new document formats for the Web and can be considered a meta-language: a language for defining markup languages. XML is a text-based format that provide mechanisms for describing document structures using markup Tags. This document aims to define a correspondence between the class diagrams of Unified Modelling Language (UML) and XML. We propose a model that will generate code from Design i.e. UML class diagram. UML is being used as de-facto standard for software development, including web applications that exchange XML documents. Therefore a need arises to integrate XML schemas, i.e., schemas written in XML Schema, into UML-based software development processes. Not only the production of XML schemas out of UML models is required, but even more the integration of XML schemas as input into the development process, because standard data structures and document types are part of the requirements.

Since both the technologies UML and XML are just beginning to grow, not much work has been done in the domain of mapping between these two techniques. Grady Booch et al[1], describes a graphical notation in UML for designing XML Schemas, in this paper the authors describe the relationship between UML and XML Schema. Carlson ([2]) describes an approach based on XMI rules for transforming UML to XML Schema. [2] also defines a UML profile which addresses most XML Schema concepts, except of simple content complex types, global elements and attributes, and identity constraints. Regarding semantic equivalence, the profile has some weaknesses in its representation of model groups, i.e., sequence, choice, and all.In [3], Provost has addressed some of the weaknesses of [2], addressing representation of enumerations and other restriction constraints, and of list and union type constructors, although the latter doesn't conform to UML.

Wu and Hsieh [5] used a technique for mapping UML to XML. They created XSD from class diagrams, but this technique is very complex to generate the XSD file for each class diagram. Mikael R. Jensen et a[l4], present an algorithm for conversion but this time in the opposite direction of our approach, they have developed

algorithms for automatically constructing UML diagrams from XML schema. Section 2 gives correspondence between UML and XML Schema and also highlights similarities and difference between UML and XML Schema. Sections 3 describe the proposed work in detail.

II. UML VERSUS XML SCHEMA

The Unified Modelling Language (UML) is a graphical language for visualizing, specifying, constructing, and documenting the artefacts of a software-intensive system. The UML offers a standard way to write a system's blueprints, including conceptual things such as business processes and system functions as well as concrete things such as programming language statements, database schemas, and reusable software components. The important point to note here is that UML is a 'language' for specifying and not a method or procedure. The UML is used to define a software system; to detail the artifacts in the system, to document and construct - it is the language that the blueprint is written in. The UML may be used in a variety of ways to support a software development methodology (such as the Rational Unified Process) - but i n itself it does not specify that methodology or process.

XML stands for eXtensible Markup Language. XML is a markup language much like HTML. XML tags are not predefined. You must define your own tags. XML uses a Document Type Definition (DTD) or an XML Schema to describe the data. XML with a DTD or XML Schema is designed to be self-descriptive. XML Schema is an XML based schema description language. An XML schema describes the structure, contents and semantics of an XML document. It is an W3C recommendation. It can be referenced through any standard XML parser. The XML Schema language is also referred to as XML Schema Definition (XSD). The purpose of an XML Schema is to define the legal building blocks of an XML document, just like a DTD. An XML Schema defines elements and attributes that can appear in a document, which elements are child elements, their order and number, whether an element is empty or can include text, data types for elements and attributes, and default and fixed values for elements and attributes. Both XML and UML have object oriented characteristics but still there some issues[6] that arise at time of transformation. One issue to be addressed is the difference in the conceptual level between XML Schema and UML. UML is primarily architectural and conceptual. XML Schema, on the other hand, is more programmatic and implementation oriented. XML Schema includes programmatic and syntactic details that are not generally captured in UML. The next issue is that of ordering. In UML, attributes of a class are unordered. On the contrary, in XML Schema, ordering information is very important. In general, XML Schema can specify whether a set of elements is ordered or unordered, i.e. specific XML Schema constructs are used to indicate whether a set of elements is ordered or not.

Another issue is that XML Schema is textual, whereas UML is graphical. Incorporating excessive textual information in a graphical model causes an overloaded and complex diagram. Finally, the different treatment of derivation in XML Schema and UML must be considered. Class derivation in UML only allows attributes to be added, not removed, from an existing class definition. This is modelled using generalization in UML. In contrast, derivation in XML Schema allows attribute addition (by extension) or elimination (by restriction).Therefore, although generalization in UML can be used to represent XML extension, another method is required to represent XML restriction in UML.

III. PROPOSED METHOD

1. With ARGO UML modelling tool create UML MODEL for the application.

We are using Argo UML for generating a UML diagram. Argo UML is drag and drop interface in which we pick a particular element like class and drop it onto editing pane.

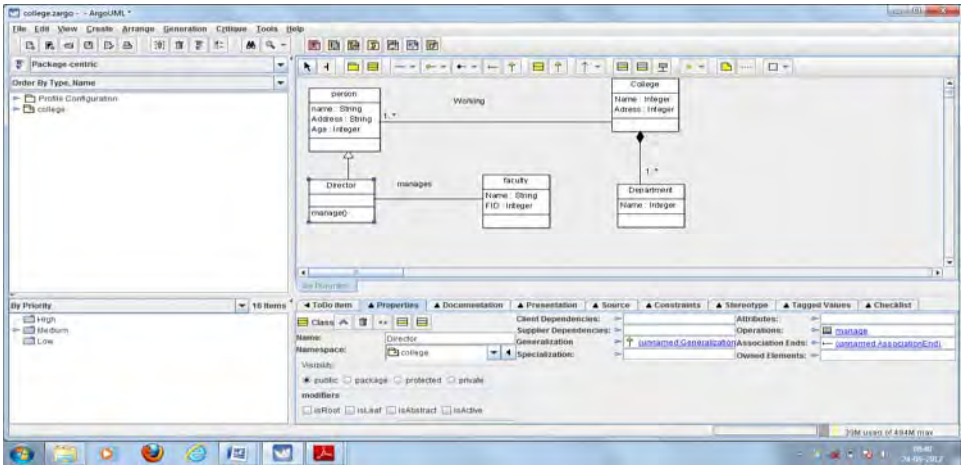

Fig 3.1 UML Class Diagram Generation

2. Read the UML MODEL and convert the model in Textual form.

UML generated by ARGO UML in previous step is codified in textual form with help of following rules:

i) Class is represented with help of square brackets "[ ]" like in above example class person is represented as [person]
ii) Class attributes are represented by parentheses "( )" and access specifier are represented using standard terminology of UML i.e. public is specified by symbol "+" private is specified by symbol "-" and protected is specified by symbol "#"like in above example attributes of person is represented as (+name: string ; +address: string; +age: string) and different attributes are separated by semicolon ";".
iii) Class methods are represented by curly braces "{ }" and name of method is followed by its return type and arguments access specifier are represented using standard terminology of UML i.e. public is specified by symbol "+" private is specified by symbol "-" and protected is specified by symbol "#"like in above example method of director class is represented as {+working: void : void}and different attributes are separated by semicolon ";".
iv) Association between two classes is represented by "*-*" like in above example person class is associated with class company it is represented as [person] *-* [college].
v) Inheritance relationship of classes is represented by "->" like in above example person class is inherited by director class so it is represented like [director] -> [person].
vi) Aggregation relationship is represented by symbol "<>" like in above example college class and department class have aggregation relationship so it is represented as [college] <> [department].

```
[person(+name: string; +address: string; +age: Integer)]
[person] *-* [college(+name: string; address: string)]
 [director{working: void : void}] -> [person].
 [director] *-* [faculty(+name: string; +FID: integer)]
[college] <>  [department(+name: string)]
```

Fig 3.2 Text UML

3. Apply Transformation rules to text UML and generate XSD file[11].
   Once a text UML file is generated we are ready to generate XML Schema. For transforming XML Schema from UML text we have generated a tool[26] which will use following mapping rules:
   i) A class in a class diagram is represented with XML element and a matching complex type.
   ii) Attributes of class is represented XML element with type attribute for data type and enclosed within <xs:sequence>
   iii) Assuming all built-in data types in UML have equivalent data types in XML. Built-in data types like int, double, string, are mapped to <xs:int>,<xs:double><xs:string>
   iv) If an existing data type is not found, it is treated as an user-defined data type. User-defined data types of UML mapping is derived through generalization from existing simple types
   v) The UML representation is either a UML attribute preceded by '/' or a UML method preceded by <<XSDDerivedAttribute>> stereotype. Derived Attributes XML element with name and type attributes for data type
   vi) A general association is represented in both classes connected by the association. reference element, with IDREF attribute referencing the associated class and keyref for type safety (key/keyref references)
   vii) Inheritance by extension is assumed and not inheritance by restriction complex type of the subclass is defined as an extension of the complex type of the super class
   viii) OCL constraints are represented using invariant, precondition and post condition. OCL constraints are mapped to restriction element.
4. Use XSD file for Code generation with help of JiBX: XSD generated in previous step is input to this step. [13] JiBX is used for code generation from XML Schema. JiBX is open source tool from IBM. JiBx generate object oriented code that is java code. This java code is our final output and can be used directly during project development hence reduce human efforts.

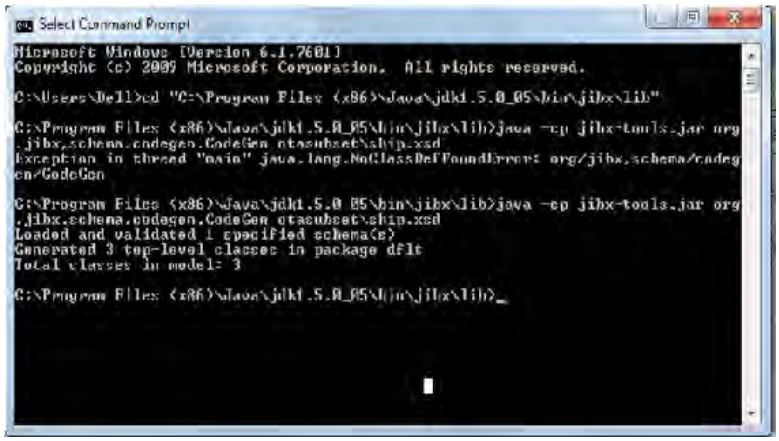

Fig 3.3 JIBX running Screenshot

IV. CONCLUSION

In this work, we have presented a new approach that produces automatically code from UML class diagram with XML Schema transformation. We have implemented this approach through a prototype. This automatic code generation can reduce efforts and development time by reducing coding efforts.

V. REFERENCES

[1] Noreddine GHERABI, Mohamed BAHAJ. Robust Representation for Conversion UML Class into XML Document using DOM. International Journal of Computer Applications (0975 – 8887) Volume 33– No.9, November 2011

[2] A. Jakimi, M. El Koutbi. An Object-Oriented Approach to UML Scenarios Engineering and Code Generation. International Journal of Computer Theory and Engineering, Vol. 1, No. 1, April 2009 1793-8201

[3] Grant, E. S., Chennamaneni, R. and Reza, H.: Towards Analyzing UML Class Diagram Models to Object-Relational Database Systems Transformations. In Databases and Applications, pp. 129{134, 2006.

[4] David Carlson. UML2 Profile for XML Schema Generation http://www.xmlmodeling.com/documentation/specs/. July 24, 2007

[5] Choi Mun-Young, Jongseon Lim, Joo Kyung-Soo. Developing An Xml Schema Generator Based On Uml Class Proceedings of the Fourth International Conference on Machine Learning and Cybernetics, Guangzhou, 18-21 August 2005

[6] Eladio Dom ınguez, Jorge Lloret, Ange L. Rubio, and Marıa A. Zapata. Evolving XML Schemas and Documents Using UML Class Diagrams. Springer-Verlag Berlin Heidelberg 2005

[7] Iftikhar Azim Niaz and Jiro Tanaka, An Object-Oriented Approach To Generate Java Code From UML Statecharts, *International Journal of Computer & Information Science*, Vol. 6, No. 2, June 2005

[8] Augustin Yu, Robert Steele. An Overview of Research on Reverse Engineering XML Schemas into UML Diagrams. Proceedings of the Third International Conference on Information Technology and Applications 2005 IEEE.

[9] Marchal, B.Working XML: UML, XMI, and code generation, Part 1, IBM Developer Works, 31 March. Available:http://www106.ibm.com/ developerworks/xml /library

[10] R. Eckstein and S. Eckstein. *XML und Datenmodellierung*. dpunkt.verlag, 2004.

[11] Salim, F.D., Price, R. Krishnaswamy, S. and Indrawan, M. 2004, 'UML documentation support for XML Schema', Proceedings of 2004 Australian Software Engineering Conference 2004, pp. 211-220

[12] Krumbein, T. and Kudrass, T. 'Rule-Based Generation of XML Schemas from UML Class Diagrams, Leipzig University of Applied Science, Department of Computer Science and Mathematics 2004.

[13] I. A. Niaz and J. Tanaka, "Mapping UML Statecharts to Java Code", in *Proc IASTED International Conf. on Software Engineering (SE 2004)*, Innsbruck, Austria, Feb. 2004, pp. 111-116.