Generation of Java Code Structure from Uml Class Diagram

Prajkta R. Pawde, Vikrant Chole

Abstract — Unified modelling language (UML) is a visual modelling language, which has gained popularity among software practitioners. The significance of automatic generation of object-oriented code from UML diagrams has increased due to its benefits, such as, cost reduction and accuracy. This paper extends our work on a tool for automatic generation of Java code structure from UML diagrams. In this UML class diagram is given as input to the tool to generate java code structure. We can use that structure for any program by coding manually for the desire output. Using already defined structure reduces cost to build the complete executable code. The extension includes a more detailed overview of the code generation tool regarding its architecture and code generation process. The propose approach only uses UML class diagram to build java code structure which is convenient for programmers and for cost reduction.

Keywords:- UML, Unified, Generatio.

I. INTRODUCTION

Software engineering is the one which consists of design, development, operation, and maintenance of software, and the study of these approaches; that is what the application of the engineering to the software. Modern day programs are much large and complex. Further, maintainability of these programs is frequent. Unified Modeling Language (UML) which is a visual modeling language, used for the purpose of support of the modeling of different views of the software. That is the main reason why, it is widely used by the software practitioners. Even though UML is being widely popular among the software practitioners, its potential in helping the software development activities is not fully known. One potential use of Class Diagram is code generation with which the human errors during manual translation of the Class Diagram to equivalent code can be avoided and also improve communication between design team and coding team which results in reduction of coding effort. Even if the different diagrams are used, each diagram has their own elements specification which may results the generated code be structural or behavioural in nature. So that, based on the UML diagrams used the code generated will also get changes. In the field of the software development, there consists of the designing and also coding of the software, which is more or less time consuming based on the complexity of the project or the designing strategy. Coding of the software with respect to the designed component of the project is what happening in the current scenario, which may create the overhead in the development phase.

Manuscript Received on June 2014.

Prajkta R. Pawde, Mtech CSE, G.H. Raisoni Academy of Engg And Technology,Nagpur, Maharashtra,India.

Prof. Vikrant Chole, Professor at CSE department, G.H. Raisoni Academy of Engg And Technology,Nagpur, Maharashtra,India.

Also this type of conversion of the design component to the corresponding code component is time and human resource consuming too. Consider the situation of software development there consists of the design and coding. Here the human resource will be divided among design and coding team, in which the coding will consume the larger amount of it. If the coding human resource can be reduced then explicitly the time consumption also gets reduced. So in order to achieve this the automatic code generation from design model diagrams is introduced. Thus the development phase is integrated with the design phase and coding for each of the application is reduced. UML diagrams are used for the designing of the software concept and the flow of the data and control in the software. UML has different versions, but mainly UML 1.x and UML 2.x. UML 2.x is used, since it supports the superstructure specification.

II. RELATED WORK

In this section, we present a few definitions, basic concepts that would be used to describe our proposed methodology.

A. OCode

OCode was developed by Ali and Tanaka to generate Java code from UML state diagram. The tool takes state diagram represented in Design Schema List language (DSL) as an input. It consists of two components known as interpreter and code generator. The interpreter generates a transition table from the DSL while code generator generates Java code from the transition table.

B. JCode

JCode was implemented by Niaz and Tanaka to generate Java code from UML statechart. The tool takes statechart represented in DSL as an input. The tool consists of three components known as interpreter, transformer and code generator. The interpreter takes DSL as an input and generates intermediate transition table. The transformer generates transformed transition table from the intermediate transition table. The transformer focuses on resolving concepts like state hierarchy, state composition, compound transition, etc in the intermediate transition table. The transformed transition table is used as an input to the code generator to generate Java code.

C. Rhapsody

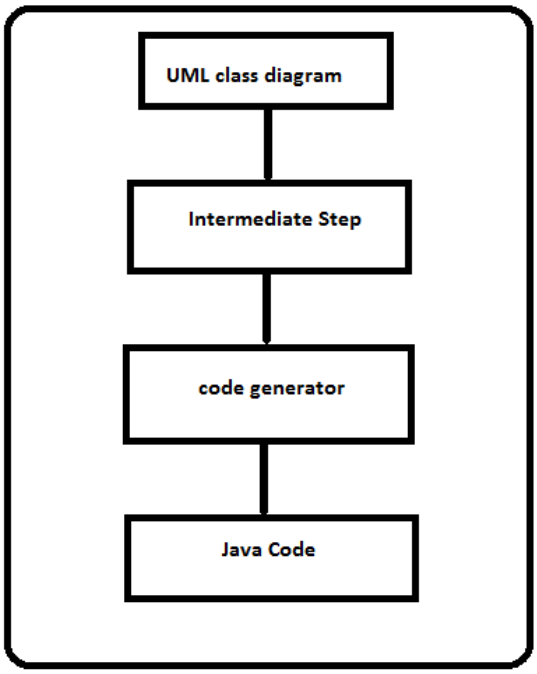
Rhapsody was developed by Harel and Gery to generate C++ code from UML object and statechart diagrams.

D. dCode

dCode was created by Ali and Tanaka to generate Java code from UML object,
activity and statechart diagrams. The tool follows the same code generation process as described for OCode.

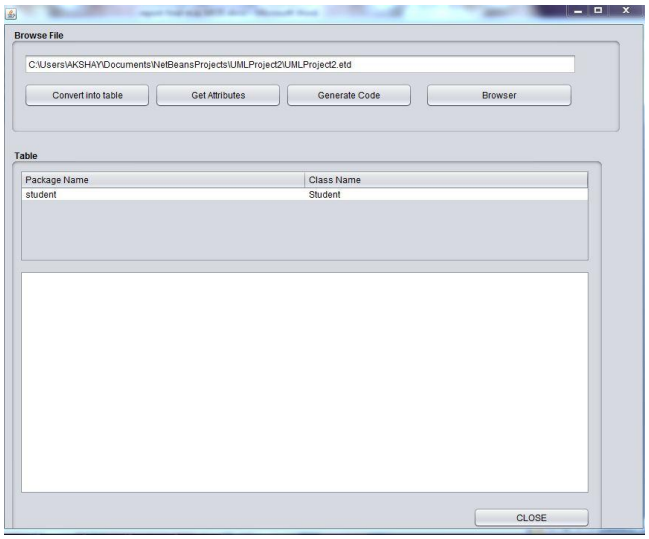
III. PROPOSED RESEARCH

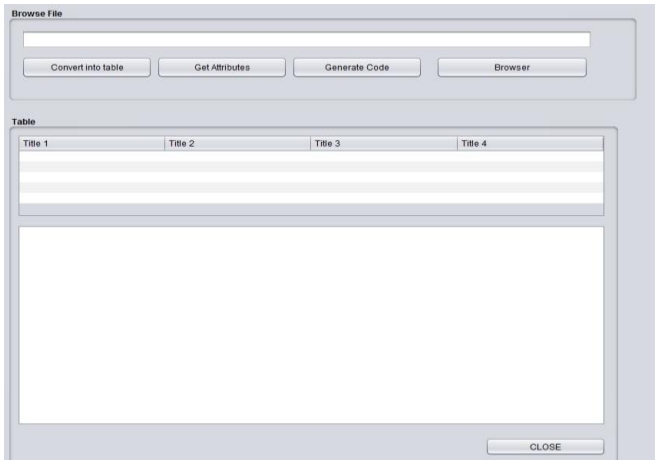
A. Tool Architecture



Above figure presents the overall architecture of the tool which implements the code generation approach described above. The architecture has two main components intermediate stage and code generator. In the intermediate stage we can identify package name and class name and attributes of class. And *CodeGenerator* is the main component responsible for generating Java code.

B. GUI Implemented for System





The GUI implemented for the system is as shown above. The functions of the buttons/panel are described as below.

Browse:
This button is used to select XMI of UML diagram that we have already drawn with the help of Netbeans IDE. That diagram is saved with .etd extension.

- **Convert into Table:**

This button is used for displaying the class and package names in tabular form.

- **Get Attributes:**

This button is used to display all the attributes that are present in the diagram in the tabular form

- **Generate code:**

This button is used for generating the final output for given UML diagram

C. Tool Process Flow

Our code generation tool process flow is divided into three main parts which are discussed below:

Input:
First we draw UML class diagram as shown below in the fig.

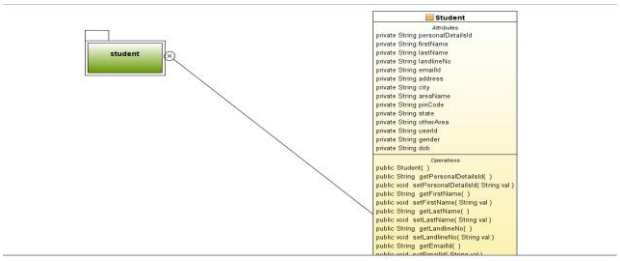


Figure Snapshot of class diagram

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication

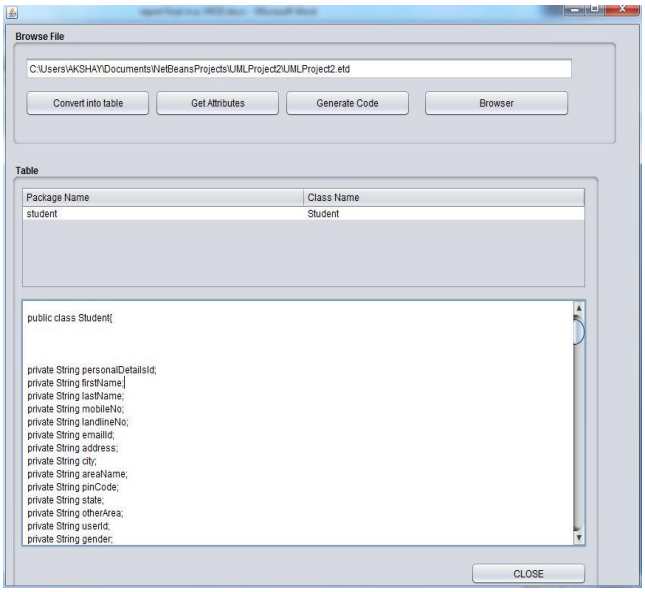
Now in the second step we generate ETD file for this diagram using Netbeans IDE which contains the class, packages, attributes, operators used in the diagram.

Transformation:

Now we generate table for package name and class name as shown below Now we get the table containing the attributes for given diagram with the help of *Get Attribute button* as shown in the below figure

Figure Attribute table

Now in the last step we are generating the final java code for UML Class diagram with the help of Generate Code button



Output:

The tool generates a summary of the generated code as shown in figure . The generated code is presented as tree hierarchy containing packages, the classes in the packages, and the methods in the classes. The generated code for any class or class method can be viewed by clicking on any class or class method.

IV. RESULT AND ANALYSIS

In this section, we analyze consistency between UML class diagram and Generated Java code, and completeness of the generated code

A. Consistency Between UML Class Diagram and Generated Java Code

To establish consistency between UML Class diagram and generated java code, we need to check whether model elements in Class diagram have correspondence with the java code. Many of the UML concepts correspond to Java language features, although UML has several concepts such as multiple inheritance and associations that are not part of the Java language. The UML concepts that are not in the Java language can be implemented in Java though. Table below lists the concepts in UML that are relevant to XMI and the corresponding Java concept, if there is one.

Table- UML and Java Comparison

UML CONCEPT	JAVA CONCEPT
Class	Class
Attribute	Field
Association	None
Association end	None
Single inheritance	Inheritance
Multiple inheritance	None
Instance	Instance
Package	Package
Datatype	Primitive type

B. Completeness of Generated Code

Code generation from the class diagram generates a limited skeleton code consisting of class attributes and method signatures. It provides the framework code for the object structure of a system. The generated code is not complete and cannot be executed. Based on the partial models of object dynamics, we have to explicitly program object behavior and communications in the target language to make it executable. We analyze the code with respect to 'completeness' attribute, which measures how much code is to be written manually after code is generated. Considering the basic constructs of Java language, we find that total five possible types of statements that may be contained within a class method and they are 'local variable declaration', 'variable initialization', 'conditional/unconditional method calls', 'variable definition/modification', 'exception handling constructs'.

V. CONCLUSION

In this paper, an approach capable of generating a structural java code from UML class diagram has been proposed. An approach involve converting a given UML class diagram into its ETD representation This generated ETD file contains information required for generating the source code for a given diagram. Then this generated ETD file is finally converted into structural java code. The proposed approach generates the Java code within class methods from UML 2.x Class diagrams. The explicit method scope information present in our XMI file helps to identify different attributes, operations for which the code is to be generated. Our study shows that the code generation from UML Class diagram is most effective for generation of java code. The results show that the generated Java code is consistent with UML diagrams. From our experimental studies, we observe that approximately 50% line of Java code is generated with our approach. That means with every generated program 50 % cost reduction will be there.

FUTURE SCOPE

The above approach works only on UML class diagrams.

- This approach can be further modified to work all existing UML diagrams.
- This approach can further be modified for complete executable code generation from the combination of UML's structural and behavioral diagram

ACKNOWLEDGMENT

I extend my sincerest gratitude to PROF. VIKRANT CHOLE, Professor, Department of CSE, GHRAET, Nagpur, under whose guidance I carried out this research. Without Prof. VIKRANT CHOLE's advice and supervision my research would never have reached its conclusion. I also thank my family, teachers, colleagues and friends for all their support.

REFERENCES

1. Booch, G. , J. Rumbaugh, and I. Jacobson, The Unified Modeling Language User Guide, Addison-Wesley, 1999.
2. 'http://xerces.apache.org/xerces2-j/', 2 November 2010
3. Kundu, D., Samanta, D., Mall, R.: 'An approach to convert XMI representation of UML 2.x interaction diagram into control flow graph', ISRN Softw. Eng., 2012, 2012, pp. 1–22
4. 'http://www.omg.org/spec/uml/2.2/superstructure', 2 November 2010
5. Michal Bliž˘nák. CodeDesigner RAD homepage. http://codedesigner.org/, 2011.
6. Niaz, I.A., Tanaka, J.: 'An object-oriented approach to generate Java code from UML statecharts', Int. J. Comput. Inf. Sci., 2005, 6, (2)
7. Jakimi, A., El Koutbi, M.: 'An object-oriented approach to UML scenarios engineering and code generation', Int. J. Comput. Theory Eng., 2009, 1, (1), pp. 35–41
8. 'http://www.magicdraw.com/', 2 November 2010
9. 'http://www.visual- paradigm.com/product/vpuml/', 2 November 2010
10. Niaz, I.A., Tanaka, J.: 'An object-oriented approach to generate Java code from UML statecharts', Int. J. Comput. Inf. Sci., 2005, 6, (2)
11. Ali, J., Tanaka, J.: 'Converting statecharts into Java code'. Proc. Of Fourth World Conf. on Integrated Design and Process Technology (IDPT99), Dallas, Texas, USA, 2000
12. Niaz, I.A., Tanaka, J.: 'Mapping UML statecharts to Java code'. Proc. Of IASTED Int. Conf. on Software Engineering (SE 2004), Innsbruck, Austria, 2004, pp. 111–116
13. Engels, G., Hucking, R., Sauer, S., Wagner, A.: 'UML collaboration diagrams and their transformations to Java'. Proc. of the Second Int. Conf. on the UML, 1999, (LNCS 1723) pp. 473–488
14. 'http://objectgeneration.com/eclipse/04-sequencediagrams.html', 2 November 2010
15. J. Ali, and J. Tanaka, "An Object Oriented Approach to Generate Executable Code from the OMT-based Dynamic Model", Journal of Integrated Design and Process Science (SDPT), vol. 2, no. 4, 1998, pp.65-77.
16. D. Harel, and E. Gery, "Executable Object Modeling with Statecharts", 18th International Conference on Software Engineering (SE'96), IEEE Computer Society Press, Berlin, Germany, March 25-29, 1996, pp.246-257.
17. Donald E. Knuth. The Art of Computer Programming Vol 1.Boston: Addison-Wesley, 3rd edition, 1997.

AUTHORS PROFILE

Prajkta R. Pawde, student of Mtech CSE, G.H.Raisoni Academy of Engineering and Technology, Nagpur.

Prof. Vikrant Chole, Professor at G.H. Raisoni Academy of Engineeing and Technology, Nagpur