ISSN: 2277-9655
Impact Factor: 1.852

IJESRT

INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY

An Approach to Code Generation from UML Diagrams

Harshal D. Gurad[*1], V. S. Mahalle[2]
guradharshal@gmail.com

Abstract

The Unified Modeling Language (UML) has now become the de-facto industry standard for object-oriented (OO) software development. UML provides a set of diagrams to model structural and behavioral aspects of an object-oriented system. Automatic translation of UML diagrams to object oriented code is highly desirable because it eliminates the chances of introduction of human errors in the translation process. Automatic code generation is efficient which, in turn, helps the software engineers deliver the software on time. However, major challenges in this area include checking consistency of UML models, and ensuring accuracy, maintainability, and efficiency of the generated code. This paper represents an approach to generate efficient and compact executable code from UML diagram. By analyzing the characteristics UML diagram, a coding strategy is proposed, and a structure identification and coding algorithm are put forward for code generation from UML diagram. Based on the coding strategy an algorithm is proposed to generate code from UML diagrams using some intermediate steps. The main objective of this paper is to generate the code from UML diagram.

.

Keywords: UML, source, code, generation, optimization, preprocessing.XMI.

Introduction

At present and in the future, the technology development is accompanied by an increase in applications complexity. Code generators are used to increase code quality and decrease envelopment time, since their goal is to generate repetitive source code while maintaining a high consistency level of the generated program code. Code generation assumes the mission of writing repetitive code parts, leaving to programmers more time to concentrate on specific code. The generators provide more productivity; generate great volumes of code, which would take much longer if coded manually. Consistent code quality is preserved throughout the entire generated part of a project. Required coding conventions are consistently applied, unlike handwritten code, where the quality is subject to variation. In case of finding errors in generated code, the errors can be corrected in short time through revising of templates and re-running the process of code generation [4].

Nowadays, there exist many software development tools able to generate source code of basic application skeleton from its formal description unfortunately, in many cases these tools lack an ability to generate complete, production-ready source code defining both the application structure and logic suitable for building without need of any modifications done by a developer..

This paper presents an approach to generate code from UML diagrams. The most important characteristic of this approach is the preserved flexibility towards the target programming language, accomplished by code generation through two transformations; first into an intermediate code and then into the code of a selected target language. Since the complexity of UML model can vary from simple to highly complex, an approach provides an efficient way for generation code from UML diagrams.

Related Work

In this section, we review existing tools and approaches related to UML-based code generation. The tools such as Omondo EclipseUML[6],Altova Umodel[7],Magic Draw[8],Visual Paradigm[9] support code generation from class diagrams. To generate behavioural diagrams, UML behavioural diagrams such as statechart, collaboration and sequential diagrams are used. Visual Paradigm [9] generates the behavioural code from statechart diagrams using templates for different operations and information associated with the transitions in the statechart diagrams.Niaz and Tanaka[10] propose an approach to generate Java code from UML class and statechart diagrams.Tanaka et al [10,11,12] reports a series of work on code generation from UML statechart diagrams.Engles et al.[13] propose a transformation process based on collaboration diagram to generate Java code.HiberObjects[14] generates code from UML 1.x SDs using templates for specific services supported by different types of object.

There are also some existing tools to automatically generate code from UML diagrams such as OCode, JCode, Rhapsody, dcode etc. OCode was developed by Ali and Tanaka [15] to generate Java code

ISSN: 2277-9655
Impact Factor: 1.852

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. JCode was implemented by Niaz and Tanaka [10] 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. Rhapsody was developed by Harel and Gery [16] to generate C++ code from UML object and statechart diagrams. It also takes message sequence charts (MSC) as an additional input. The tool takes STATEMATE representation of statechart as an input and generates C++ code. dCode was created by Ali and Tanaka [15] to generate Java code from UML object, activity and statechart diagrams. The tool follows the same code generation process as described for OCode [9].

Proposed Approach

The code generation process consists of steps as shown in Fig. 1. All these steps are important for error free generation of code

Preprocessor

First of all a source diagram is preprocessed so its structure will change in order to be more suitable for further processing by the code generator.

Verifier

Preprocessed diagram must be verified to find possible inconsistencies in the diagram's topology. If the verification fails the code generation process is aborted.

XMI Representation

After that by using some UML modeling tool we form the XML metadata information (XMI) representation of the given diagram. Proper XMI representation of given UML diagram is very much important for generation of Integrated Graph(IG).

Construction of IG

In this step, we first extract the meta-information of the given UML diagram from its XMI representation by using some standard simple API for XML (SAX) parser. After parsing the XMI representation, the SAX parser returns the attributes of the tagged elements, processing of which we obtain a set of metaobjects . In order to synthesise detail information of a message from these meta-objects, it is necessary to find the relationships among them. For this, we consider the inherent structure of the XMI representation and meta-model of the SD as per UML superstructure specification [2]. After finding the

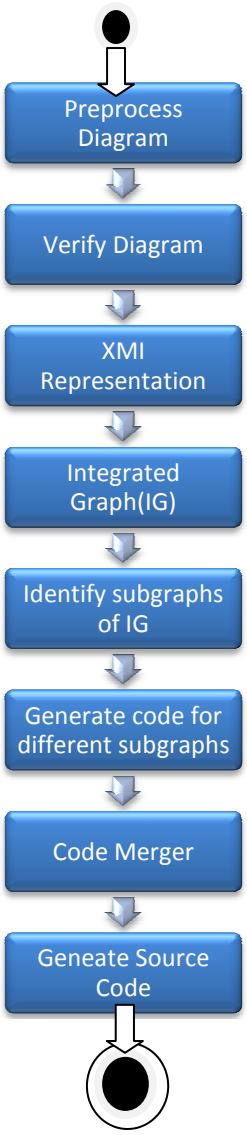

Fig. 1: Code Generation Process

details of a message, we then obtain the set of messages M, set of reply messages R, set of fragments F, message set Mf of each fragment f ∈ F. [17]

Subgraph Identification

Then we identify the subgraphs of given graph by some predefined method. We divide our IG into subgraphs so as it will be convenient for coding.

Generating code for different subgraphs

In this step we are generating code for different subgraphs. Here we are using an algorithm for code

generation. This code generation algoritm writes source code fragments. Here some techniques are used so that proper conversion of given UML diagram into our requires programming language since some language doesn't have to support all command statements produced by the algorithm.

Code Merger

Code Merger takes the code from isolated subgraphs and merges them so as to form the appropriate code representation of given UML diagram.

3.8Generaed source code

After merging all the codes for different subgraphs we are getting final source code for our given UML diagram

Code Optimizations

Generated source code can be optimized at several different levels. Optimizations performed on processed diagrams. lead to production of optimized source codes with reduced extent or better readability due to better source diagram's topology. Also, some low-level optimization can be performed to obtain efficient code. Optimization of generated code helps in obtaining efficient code, eliminating unused code etc.

Conclusion

We have proposed an approach to generate the code from UML diagrams.

By using this approach we are not getting complete executable code since any single UML diagrams doesn't provide all the information required for generation of complete executable code.

References

[1] *Object Management Group (OMG), Unified Modeling Language Specification, Version 2.1.1, (2007-02-07).*

[2] *Booch, G., Object Oriented Design with Applications, Benjamin/Cummings, Redwood, California, 1991. ISBN: 0-8053-0091-0, ISSN: 0896-8438*

[3] *Coad, P., and E. Youdon, Object-Oriented Analysis, Prentice Hall, Eaglewood Cliffs, New Jersey, 1991. ISBN: 0-13-630070-7*

[4] *Herrington, J., Code Generation in Action, Manning, 2003.*

[5] *Michal Bližňák. CodeDesigner RAD homepage. http://codedesigner.org/, 2011.*

[6] *'http://www.ejb3.org/', 2 November 2010*

[7] *'http://www.altova.com/umodel/uml-code-generation.html', 2 November 2011*

[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] *D.Kundu,D.Samnta,R.Mall : 'Automatic code generation from unified modeling language sequence diagrams',pub in IET Softw., 2013, Vol. 7, Iss. 1, pp. 12–28 & The Institution of Engineering and Technology 2013 doi: 10.1049/iet-sen.2011.0080*

[18] *http://www.omg.org/spec/uml/2.2/superstructure', 2 November 2010*