International Journal on Recent and Innovation Trends in Computing and Communication ISSN: 2321-8169
Volume: 4 Issue: 3 365 - 368

Automatic Partial Code Generation Using Class and Sequence Diagrams

Mehjabin Pathan, Aakash Patkar, Sayali Surve
Dept. of Information Technology,
K. J. Somaiya Institute of Engineering & Information Technology,
Mumbai (India)
mehjabin.p@somaiya.edu,aakash.patkar@somaiya.edu,sayali.ps@somaiya.edu

Abstract- The main objective is to provide overview of one of the Model Driven Engineering Tool i.e. Transformation tool. Model Driven engineering tool is used to develop, interpret, compare, align, measure, verify, transform, etc. models or Meta models. Model is interpreted as "UML Model". UML based approach provides abstraction to deal with the complex embedded applications and when combined with Model-driven Engineering can also provide automation through automatic code generation. This paper presents an approach to generate structural and behavioral code from UML class and sequence implementation of a code generator.

Keywords- *Model driven Engineering Embedded software; Code generation; UML.*

*****

I. INTRODUCTION

Model-driven architecture (MDA) is a technique to develop software. Model-driven architecture supports model-driven engineering of software systems. Model driven engineering Describes an approach for development of software where models are used as the primitive source for documenting, analyzing, designing, constructing, deploying and maintaining a system[1].

An MDA tool can be one of following types: Simulation, Analysis, Reverse Engineering Transformation, Composition, Test, Creation.MDA tools are used to transform, compare, interpret, align measure, develop, verify, etc. models .A model is a representation or description of system. Model is interpreted as "UML Model"

The UML is a general-purpose, tool-supported, and collection of modeling techniques for visualizing, constructing, and documenting the artifacts of a software system. Amongst multiple MDA Tools, our tool belongs to Transformation tool named "Code Maestro". Model transformation is the technique for transforming one model to another in the same system. The transformation collaborates the platform independent model with some external information to generate a platform specific model.

II. LITERATURE SURVEY

The amount of various software in embedded systems is growing up. Its increase in difficulty combined with timely restrictions has motivated the investigations for software that should be able to reduce costs and accelerate the product delivery. Usually models are used to handle system with complexity through graphical views and abstraction. UML is the standard language for modeling and provides various graphical diagrams to give different views of a system and has been considered significant in the field of model complex embedded systems. When models are used, these must be translated to code to obtain an implementation [4].

Recently, Model-driven Engineering techniques, used by the software organization, have gained attraction in the embedded community, promising automation and abstraction for embedded software development.

III. EXISTING SYSTEM

In Existing System, Model-driven Engineering many transformation tools are available. The free and open source software tools are available to create Unified Modeling Language (UML) class diagram and transform code. The features in Existing tools are:

- Java support with many language specific elements.
- Simple and easy to use user interface.
- Source code generation.

The issues in existing transformation tools i.e. code generation tools are:

- Supports only class diagram of the standard UML diagram types.
- UML Diagrams are able to generate only structural code.
- Does not support Getter and Setter methods.
- Loops and Conditional elements cannot be supported.
- Most of the tools requires internet for the usage.

IV. PROPOSED SYSTEM

The aim of the proposed system is an approach for code generation from UML models able to generate structural and

as well as behavioral code. In our approach, embedded applications are modeled using a UML class diagram to give a structural view and various sequence diagrams to give the behavior view. The main sequence diagram is referred as main method and defines the start point of the behavioral code. From the class diagram, structural code is generated.

From each class, a Java code is generated, describing its attributes and methods, and including the constructor method with attributes initialization passed by parameter. Loops and conditions will be captured from respective diagram generating corresponding Java statements. It will consider relationship between classes or interfaces as well as it generates get and set methods. In case if there is an inheritance including an interface or an abstract class, methods defined by them are generated as methods into the code of its immediate concrete subclass. From the sequence diagrams, the sequence of methods is captured including returns and arguments.

A. BLOCK DIAGRAM

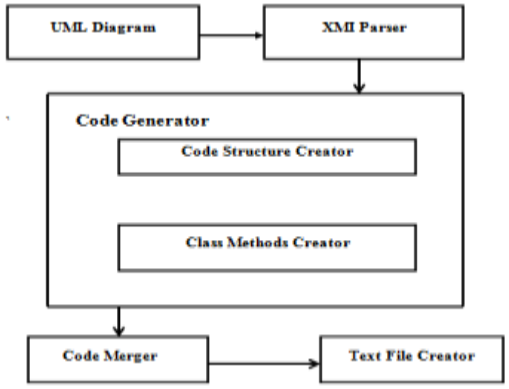

Fig 4.1 Block Diagram of Code Maestro Tool

Fig 4.1 Presents the Block diagram of Automatic Code Generation Tool i.e. Code Maestro tool which implements our proposed code generation approach [2]. The architecture contains three main components known as XMI Parser, Code-Generator, Code Merger, and Text File Creator. Code Generator component is main component responsible for generating Java code. Class Method Creator and Class Structure Creator are the internal components of Code Generator component [2].

A. XMI Parser

XMI Parser starts the execution of our tool. It takes UML models tokens as an input and generates UML models Meta model instances. UML models XMI consists of UML class and sequence diagrams.

Class Diagram Parser class extracts information about UML class diagram from the input XMI and fills UML class diagram meta model instance. It starts with package element contains class and interface elements which referred in relationships like, composition, association, generalization, aggregation etc. Interaction Diagram Parser class extracts information about UML sequence diagram from the input XMI and fills UML sequence diagram meta model instance.

B. Code Generator

Code Generator component is an important Component in our tool.UML Meta model instances are taken as an input and generates Java code. The components' explanation is as under.

a) Class Structure Creator

Class Structure Creator component takes UML class diagram metamodel instance and outputs Java structural code which contains classes, their attributes, their method signatures. It also contains interfaces and their methods. Relationships between classes are included in structural code.

b) Class Method Creator

Class Method Creator component takes UML sequence diagram meta model instance and outputs class methods code which contains control flow, alternative and iterative blocks.

a. Code Merger

CodeMerger component takes the isolated Java code from Code Generator component and includes class methods' code into their corresponding classes. It is responsible for generating the complete Java code for classes and interface which contain full functional class methods.

b. Text File Creator

Text File Creator component takes Java Complete code as input and creates Text file at the defined location by placing the Java code .

B. PROPOSED SYSTEM DETAIL DESIGN

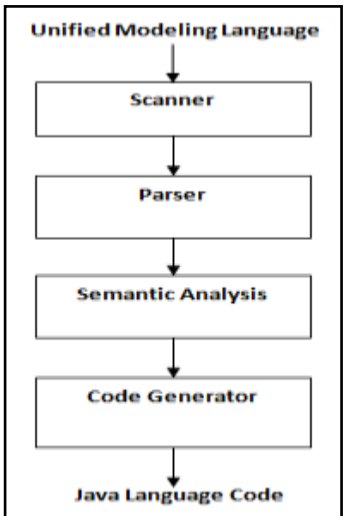

Fig 4.2 System Detail Design of Code Maestro Tool

Fig 4.2 presents proposed system design detail. The proposed transformation tool acts as a compiler where it requires many step such as scanner, parser, semantics analyzer and code generator [5].

a) **Scanner**:

The scanner scans your source code and turns it into atomic units called tokens.

b) Parser:

The tokenized code is then passed through a parser to identify and encode its structure and scope into what's called a syntax tree.

c) Semantic Analyzer:

Semantics help interpret symbols, their types, and their relations with each other. Semantic analysis judges whether the syntax structure constructed in the source program derives any meaning or not.

d) Code generator:

The last phase of transformation is code generation. The output of the code generator is the Java Language program of the specified UML Diagram in text file.

V. CASE STUDY

An ATM machine is chosen as case study, because it is a simple example of embedded software and allows demonstrate the main features of the proposed approach. A UML model consisting of one class diagram and three sequence diagrams, will be build to represent static and dynamic aspects of the ATM Machine system using Code Maestro tool.

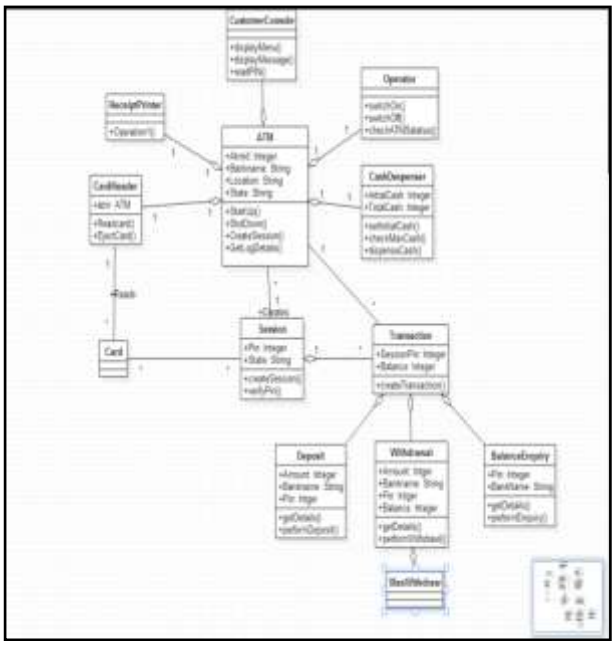

Fig. 5.1 Class diagram of ATM case study

The static view is represented by the class diagram from Fig 5.1. Basically, this model is composed of seven concrete classes (ATM, CustomerConsole, CashDespenser, TransactionSession, CardReader, RecieptPrinter, Operator), the abstract class Transaction, and the interface Customer. The class Deposit, Withdrawal, Balance Inquiry extends Abstract class Transaction. The ATM is the main class, which has the method main, beside of other methods representing the ATM machine operations.

ATM is associated to the classes Operator, CustomerConsole, CardReader, and RecieptPrinter. In the class CustomerConsole has some methods such as readPin, displayMessage, and displayMenu, used to access attributes and whose usages are demonstrated in the sequence diagrams depicted in Fig. 5.2

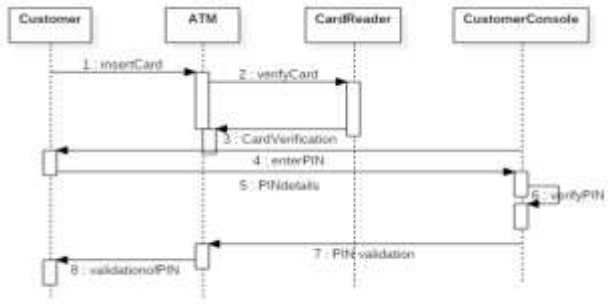

Fig. 5.2 Sequence diagram for ATM case study

Following our approach, a sequence diagram is used to represent the behavior of the method main of the classATM, which is depicted in Fig 5.3.
Applying our approach, it will generate Java code from the ATM machine model using CodeMaestro tool. Listing 1 (a) and (b) illustrate the codes generated for the abstract classTransaction and for the interface Customer, declaring entities, its attributes and methods.

ABSTRACT CLASS

```
public abstract  class Transaction
{
int PIN;
int balance;
public abstract createTrasaction();
}
```

1(a)

INTERFACE

```
public interface Customer
{
void insertCard();
int getPIN();
void ejectCard();
}
```

1(b)

To demonstrate the behavioral code generation, code Fragments of the method main of the ATM,will be generated from the corresponding (Fig 5.2) The generated codes will also include the constructor with initialization of

attributes. It will also include the Getter as getPin() method. To finalize the code, the tool will generate the methods defined in the interface Customer, which must be implemented in this class, since a realization relationship will be defined between these entities. Thus the code for main class will be generated in a following manner in fig 5.3.

MAIN CLASS

```
public class ATM implements Customer
{
int atmID;
String bankName;
String location;
int transTime;

public ATM(int transTime)
{
this.transTime=transTime;
}

public static void main(String args[])
{
Customer.insertCard();
PIN=Customer.getPIN();
valid=CustomerConsole.readPIN(PIN);
if(valid==”TRUE”)
{
option=Transaction.getTransOption();
Switch(option)
{
case 1: performDeposit(); break;
case 2: performWithdrawal(); break;
case 3: performEnquiry(); break;
default: break;
}
}
else
{
System.out.println(“Wrong password!.....”);
System.out.println(“Please reenter Password”);
}
}
```

Fig 5.3 Main Class Generated Code for ATM Machine Case Study.

VI. CONCLUSION

This paper presented an approach for code generation from UML models based on class and sequence diagrams. Our approach named Code Maestro cannot generate complete code, since the sequence diagram granularity is the method invocations. However, it offers abstraction and a reliable automation for the embedded software Development. In future we have planned to generate executable code using this tool and we will enhance it support other languages.

VII. REFERENCES

[1] B. Selic, “Uml 2: A model-driven development tool. Modeldriven software development,” IBM Systems Journal, Riverton,vol. 45, n. 3, pp. 607–620, 2006.

[2] M. Usman and A. Nadeem, “Automatic generation of java code from uml diagrams using ujector,” International Journal of Software Engineering and its applications (IJSEIA), Daegu, vol. 3, n. 2, pp. 21–37, 2009.

[3] I. A. Niaz and J. Tanaka, “An Object-Oriented Approach to Generate Java Code from UML Statecharts”, *InternationalJournal of Computer & Information Science*,vol. 6, no. 2, 2005.

[4] B. Selic, “Models, software models, and uml,” UML for real:Design of embedded realtime systems, vol. Boston: Kluwer Academic Publishers, pp. 1–16, 2003.

[5] “Object Oriented UML Modeling for ATM Systems”:Department of computer technology, VJTI University, Mumbai.