ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

|A Monthly, Peer Reviewed Journal| Impact Factor: 7.089|

Vol. 6, Issue 6, June 2017

Compilation of Single Language GUI to GUI in Multiple Languages Using Multitargeted Program Generation Model

Mr. Bhagwat Shankar Uchale

Assistant Professor, Department of CSE., Karmaveer Bhaurao Patil College of Engineering, Satara, MH, India

ABSTRACT: The MAGIC Script is a simple and easy-to-understand graphical interface description language. It needs to be converted to different target languages based on the user's requirements, making it a Multi-Target Language. The target languages may include JFC, AWT, XUL, HTML, or C. This study focuses on the syntax and semantics of the MAGIC scripting language. The MAGIC script incorporates various components common to all the target languages, making it easier for developers to write GUI code in multiple languages. Additionally, the MAGIC script helps reduce development time, with an input-to-output ratio of around 1 per the number of target languages.

KEYWORDS: MAGIC Script, Program generation Model.

I. INTRODUCTION

The MAGIC script needs to be converted into different target languages based on the user's requirements, thereby making the compiler Multi-Target. The system comprises a compiler that translates MAGIC script into specific target languages, such as Java Swing, Java AWT, XUL, or HTML. The system should be designed to allow for the addition of new languages in the future. An integrated development environment (IDE) for creating the MAGIC Script must also be provided.

The compiler is implemented using the JavaCC parser generator. It identifies all the tokens defined in the MAGIC Script and checks for parsing errors. The grammar is specified in BNF. The compiler takes an input file with a .mgk extension and converts it to the target language based on the specified switch. When processing a token like 'button' with the AWT switch, for example, the referenced class is AWTButtonDataType. Subsequently, a function from the referenced class is invoked based on the specified property or event for that control.

The operation of a compiler is divided into four phases: Tokenizing, Syntax analysis, Semantic analysis, and Translation.

II. RELATED WORK

This chapter provides information about existing tools for this system, such as Jvider and GrafiXML, and also discusses their advantages and disadvantages. Jvider is a GUI builder tool for Java swing applications. With Jvider, you can easily design graphical user interfaces for your Java applets and applications. It provides an understandable interface, draggable and resizable components, and the ability to generate the source code in Java language. Moreover, it can export the source code as a frame or applet. Jvider is platform-independent and runs on all environments that support Java Virtual Machine (JVM).

Jvider features:

- Remember the following text: "Standard Java Swing components."
- GridBagLayout is used for layout (no absolute positioning);
- View and export produced Java™ source code as Frame or Applet;

ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

|A Monthly, Peer Reviewed Journal| Impact Factor: 7.089|

Vol. 6, Issue 6, June 2017

Advantages:

- Easy to get started with tutorial and samples provided;
- Powerful Undo/Redo support
- Disadvantages:
- It is used for creating GUI in one language only.

GrafiXML is a software tool used for creating graphical user interfaces (GUI) and saving them in the UsiXML format language. It functions similarly to other GUI builders but also includes tools for localizing your applications. Unlike traditional user interface builders that manipulate physical widget properties, GrafiXML manipulates more widget properties and saves user interfaces in the UsiXML format instead of a specific code format. GrafiXML is distributed under the terms of the Apache License, Version 2.0, explaining how the resulting UI design can support one or many levels of independence concerning the underlying context of use.

III. PROPOSED COMPILER

1. Developer Related:
   The developer is asked to write the source code called a magic script.
   The developer selects the language in which he wants the GUI code.
2. Compiler related:
   The compiler is provided with MAGIC script as a source
   Detects token and compiles it to the target language.

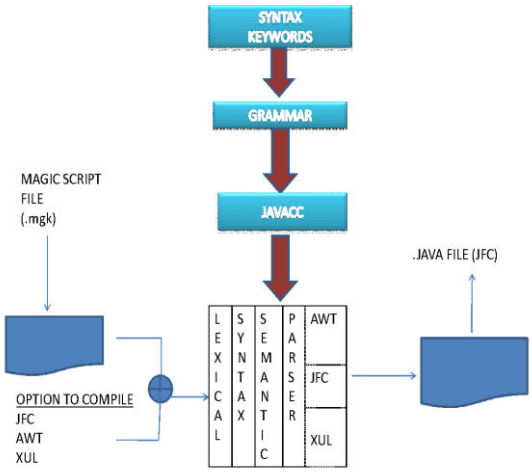

Fig 3.1 Prototype Interface Diagram

The compiler has been implemented using the JavaCC parser generator to detect all the tokens described in the MAGIC Script and check for parsing errors. Each GUI designing language has common GUI components, for which component class files are created. All the classes related to a specific target language are similar to classes in other target languages.

The compiler takes an input file with the .mgk extension and converts it to the target language based on the switch entered. It references the class related to a token for a given target language by adding the language switch before the common class name for that token.

For example, when receiving the token 'button' and switching to AWT, the referenced class is AWTButtonDataType. Then, a call is made to a function written in the referenced class based on which property or event is mentioned for that control. The resulting code is then written in the output file by the called function. This design strategy makes Magic scalable, as adding a new language is easier. Error detection and maintenance are also simplified.

ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

|A Monthly, Peer Reviewed Journal| Impact Factor: 7.089|

Vol. 6, Issue 6, June 2017

MAGIC will be concerned with the following characteristics

- **Follow Object Oriented concepts**: - Building the compiler in Java which is fully objecting oriented.
- **Reliable:-**Developer can rely on software to get code in target languages.
- **Extendable** can add other target languages & and drag & drop features.
- **Flexible:-**It gives output for five languages.
- **Error Free: -** In build JavaCC helps us for detection of errors.
- **Reusability** write the code once & generate code in various target languages.

IV. PSEUDOCODE

Step 1- The user has to open the IDE provided.
Step 2- The user has to write an input script and save the file with the .mgk extension.
Step 3- The file is passed as input to the parser.
Step 5- The parser parses the tokens one by one and passes them to the switching mechanism.
Step 6 The user has to select the target through the command switch.
Step 7 The tokens and command switch is passed as input to the switching mechanism.
Step 8-Switching mechanism generates the class file name by the formula:- Command switch + token + Data type.
Step 9- According to the command switch selected the class will be passed towards the Code Generator.
Step 10- In the code generator the component will be selected as per the class name generated by the switching mechanism.
Step 11- As per the sub tokens generated by the parser the methods in the component are selected.
Step 12 The methods invoked will write the GUI code in the output file.
Step 13- The output file is compiled to get the GUI in the desired language.

V. TEST CASES

Test Case 1: Working of GUI

Table 5.1 TestCasee 1

Test Case	Working of GUI
Description	This test case checks whether the GUI is working or not
Basic Flow	1. Start IDE 2. Click on File ->Open
Acceptance Criteria	Open MAGIC script if it has extension .mgk
Post Condition	The MAGIC script gets opened.

Test Case 2: Checking of "Compile to AWT" Option

Table 5.2 TestCasee 2

Test Case	Checking the "Compile to AWT" option
Description	This test case checks whether AWT is working or not
Precondition	The user should have a MAGIC script in IDE.
Basic Flow	1. Open IDE 2. Open the MAGIC script. 3. Click on Tools → Compile to AWT
Acceptance Criteria	Compile MAGIC script if it has extension .mgk
Post Condition	The user views AWT code.

ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

|A Monthly, Peer Reviewed Journal| Impact Factor: 7.089|

Vol. 6, Issue 6, June 2017

Test Case 3: Checking of "Compile to JFC" Option

Table 5.3 TestCasee 3

Test Case	Checking the "Compile to JFC" option
Description	This test case checks whether JFC is working or not
Precondition	The user should have a MAGIC script in IDE.
Basic Flow	1. Open IDE 2. Open the MAGIC script. 3. Click on Tools → Compile to JFC
Acceptance Criteria	Compile MAGIC script if it has extension .mgk
Post Condition	The user views JFC code.

Test Case 4: Checking of "Compile to XUL" Option

Table 5.4 TestCasee 4

Test Case	Checking the "Compile to XUL" option
Description	This test case checks whether XUL is working or not
Precondition	The user should have a MAGIC script in IDE.
Basic Flow	1. Open IDE 2. Open the MAGIC script. 3. Click on Tools → Compile to XUL
Acceptance Criteria	Compile MAGIC script if it has extension .mgk
Post Condition	The user views XUL code.

Test Case 5: Checking of Generated AWT Code.

Table 5.5 TesCasese 5

Test Case	Checking of generated AWT code.
Description	This test case checks whether the generated code is running
Precondition	The user should have an AWT code.
Basic Flow	1. Open AWT code 2. Run the file.
Acceptance Criteria	Run the AWT code if it has an extension .java
Post Condition	The user views AWT output.

Test Case 6: Checking of Generated JFC Code.

Table 5.6 TestCasee 6

Test Case	Checking of generated JFC code.
Description	This test case checks whether the generated code is running
Precondition	The user should have a JFC code.
Basic Flow	3. Open JFC code 4. Run the file.
Acceptance Criteria	Run the JFC code if it has an extension .java
Post Condition	The user views JFC output.

ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

|A Monthly, Peer Reviewed Journal| Impact Factor: 7.089|

Vol. 6, Issue 6, June 2017

VI. CONCLUSION AND FUTURE WORK

The field of MAGIC is advancing quickly with ongoing research and development. We focused on key issues and existing solutions in this area. One major challenge we continue to encounter is creating genuinely simple, less complicated, and time-efficient methods for building graphical user interfaces (GUI) in various programming languages. MAGIC consists of two components: the compiler and the User Interface Engine (UIE), both of which have room for improvement. The MAGIC compiler can be extended in the following manner:

1. MAGIC can be made to support Java Bean, JSP, ASP, HTM, L, and any other GUI designing language.
2. MAGIC Script can be extended to be a fully-fledged language.
3. Magic UIE can be made to support fully-fledged drag-and-drop functionality.

REFERENCES

WEBSITES

1. https://javacc.dev.java.net/
2. http://www.scifac.ru.ac.za/compilers/conts.htm
3. http://java.sun.com/products/jfc/download.html
4. http://download-llnw.oracle.com/javase/1.4.2/docs/api/java/awt/package-summary.html
5. https://developer.mozilla.org/en/introduction_to_xul
6. http://download.oracle.com/javase/tutorial/reflect/index.html
7. http://java.net: JavaCC [tm]: Grammar Files

BOOKS

8. Alfred Aho and Jeffrey D. Ullman,(1986) Compilers: Principles of Compiler Design.Advanced programming in java NIIT, Prentice Hall of India, ISBN-81-203-2415-3.
9. Raphael A. Finkel, Advanced Programming Language Design.
10. Andrew Appel, Jens Palsberg: Modern Compiler Implementation in Java. Cambridge University Press, 2nd edition, 2003.
11. Software engineering.

PAPERS

12. Benjamin Michotte, Jean Vanderdonckt, “GrafiXML, AMultitarget User Interface Builder based on UsiXML”.
13. Prof. H. H. Patel, Miss Sayali Ahire, Shubhangi Keskar, Shubhangi Devikar, Kalyani Gavade, Multi-Targeted Graphical User Interface Compiler, Vol. 3, Issue 03, 2015 | ISSN (online): 2321-0613
14. Harshad Rane, Brijeshkumar Gupta, Akshay Mhatre, Yogesh Gaikwad, Multi Graphical User Interface Compiler, Volume 3 | Issue 1 | 2017 Print ISSN: 2395-1990 | Online ISSN: 2394-4099
15. Amruta Mukund Talathi, Sarita V. Balshetwar, Dhiraj Vijay Motghare, A Java-Based 4th Generation Multi-Targeted User Interface Compiler (JUICE),2015