International Journal of Modern Engineering Research (IJMER)
www.ijmer.com Vol.3, Issue.2, March-April. 2013 pp-924-927 ISSN: 2249-6645

XML Based Reverse Engineering System

Soon-kak Kwon[1], Hyungchul Park[2]
[12]*Department of Computer Software Engineering, Dongeui University, Korea*

Abstract: *Reverse engineering that gets data such as an original document or design technique by tracing reversely from certain software system is required in the sides of maintenance and reuse. This paper describes how to implement a system which can get class from one source file, member variable and method within class, and association of between different classes by using reverse engineering, and then can display to XML document and save the document. Also, it analyzes the number of declared variables and functions used in reverse engineered project and displays it as GUI form to give convenience to user.*

Keywords: *Reverse Engineering, Package, xml, Class*

I. Introduction

Reverse engineering [1, 2] is one field of software engineering. It means the process that gets data such as a document or design technique by reversely tracing an already made software system. It must be essential in software maintenance process that understands the system and changes it appropriately, and has significant meaning of reuse from expansion through the analysis of the implementation of the system.

Software programmers as well as consumers who buy the software has tendency to ignore inner structure and attach importance to result of the project. So, as many team members make one software project, individual of team members makes one by using own method, and assemble them. As a result, unused resources waste resources, and independent modules are not implemented optimally, also association among modules may not be correctly defined.

Therefore, if we use reverse engineering, we can extract information such as class, member variable and method within class, and association of between different classes from one source file, and save them in a specific document. Through it, we can extract the number of declared variables and functions used in target project for reverse engineering to be useful for maintenance, and draw class diagram in detail which is drawn on basic tool.

This paper describes how to implement a system which can do reverse engineering from source file made by JAVA, and save the result as an XML document.

II. XML Based Reverse Engineering

The flowchart of proposed reverse engineering is shown in Fig. 1.

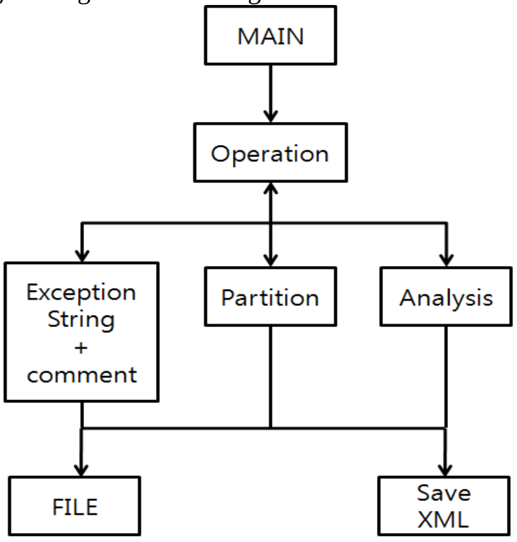


Fig. 1. Flowchart of proposed reverse engineering system

If the square is one class, the main class starts program and the operation class maintains classes by dividing three classes of string+comment exception, partition, and analysis.

The string+comment exception class removes comments and string variable values. The partition class makes a file separately and saves it when many classes are declared in one class file.

In the analysis class, information are saved in sense of class types (class, interface), variable types(data variable, instance variable), and function types(general method, abstract method). Also, information of relationship whether it is inherited or built through interface are extracted and saved.

The file class, used as API, has functions of file open, close, write, save, find, etc.

Then, the usage count class makes a report by extracting usage count of declared functions and variables, and draws a class diagram to see relation between classes easily.

A class is analyzed by five steps of removing risks, partition, sentence analysis, saving XML document, extracting association.

1) Removing risks

For example, when ‘String str = "class";’ is declared, it may be risky because 'class' word partitions the class. So, we need to remove risky elements when a sentence is analyzed.

Also, because we need not to analyze a single line comment (//~~~\n) and a multiple comment (/* ~~~ */) when a project is analyzed, the comments are first removed.

2) Partition

Class division divides

Many classes can be declared by inner classes in one JAVA file or many classes within one class. The partition divides into individual classes. Partitioned classes are saved to temporary files and they are used into sentence analysis.

3) *Sentence analysis*

Fig. 2 shows algorithm which explains flow of sentence analysis. In Fig, 'public class A{' sentence is presented as an example, it reads each one character, then if there is a non-character like a blank or special character, it recognizes one word to before non-character.

Recognized words are analyzed by finding out meaning of each word through a file that defines reserved words and keywords. And the analyzed words are saved to matched values, then the sentence analysis catches properties about the line, that is, class, interface, method, abstract method, data variable, instance variable, or association as existence of special characters((), {}, ;). Then, analyzing current line is complete.

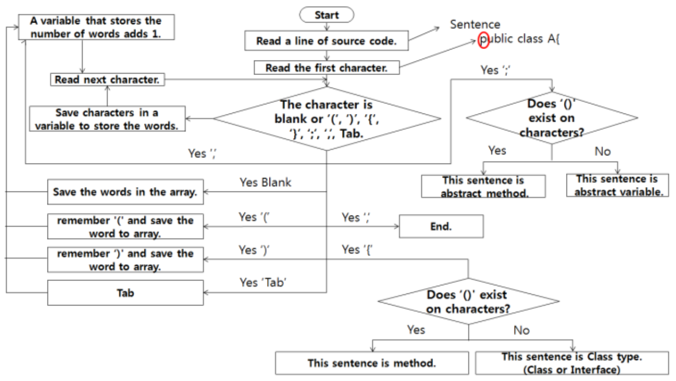

Start
Read a line of source code.
Sentence
public class A{
Read the first character.
A variable that stores the number of words adds 1.
Read next character.
Save characters in a variable to store the words.
The character is blank or ‘(’, ‘)’, ‘{’, ‘}’, ‘;’, ‘,’, Tab.
Yes ‘;’
Does ‘()’ exist on characters?
Yes
No
This sentence is abstract method.
This sentence is abstract variable.
Yes ‘,’
Yes Blank
Save the words in the array.
Yes ‘(’
remember '(' and save the word to array.
Yes ‘;’
End.
Yes ‘)’
remember ‘)’ and save the word to array.
Yes ‘{’
Yes ‘Tab’
Tab
Does ‘()’ exist on characters?
Yes
No
This sentence is method.
This sentence is Class type. (Class or Interface)

Fig. 2. Algorithm of sentence analysis

4) Saving XML document

The values which get through sentence analysis are saved in order as XML structure. The values which are saved through node's name and sentence analysis are constructed into four depths by condition statements.

```
<Package>
     <Package Name>
     </Package Name>
     <Class Type>
        <Class Type Data>
        </Class Type Data>
        <Method>
           <Method Data>
               <Parameter Data>
               </Parameter Data>
           </Method Data>
         </Method>
         <Variable>
            <Variable Data>
            </Variable Data>
         </Variable>
         <Relation>
             <Relation Data>
             </Relation Data>
         </Relation>
     </Class Type>
</Package>
```

Fig. 3. XML structure

Xml document is saved as structure following depth.
Depth 1 is package.
Depth 2 is package name, and class type (class, interface).
Depth 3 is class type information (extends, implement, visibility, static, final, etc.), method, and variable.
Depth 4 is method information, variable information, and relation information.
Depth 5 is parameter information (visibility, return type, static, final, etc.)

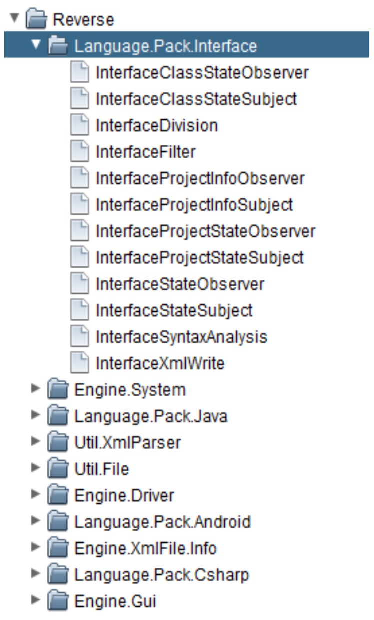
5) Extracting association

The information whether the value saved in XML is function built to override abstract method through interface, or whether it is function or variable inherited from parents class are extracted from class name built by 'implements' and class name inherited through 'extends', and then saved.

III. Implementation

It is easy to extract and find particular information, but it is hard to see information easily when saved as XML.

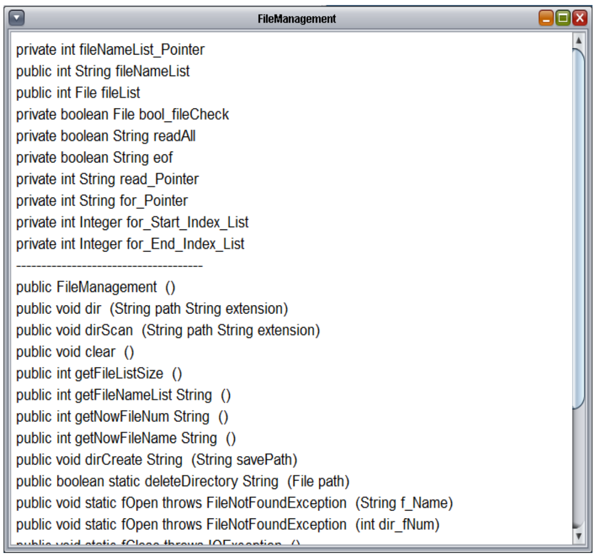
So, the XML based GUI is implemented. A method which is implemented to input directly path is more comfortable and accurate. Fig. 4 shows the classes saved as tree form after analysis.



```
▼ Reverse
  ▼ Language.Pack.Interface
      InterfaceClassStateObserver
      InterfaceClassStateSubject
      InterfaceDivision
      InterfaceFilter
      InterfaceProjectInfoObserver
      InterfaceProjectInfoSubject
      InterfaceProjectStateObserver
      InterfaceProjectStateSubject
      InterfaceStateObserver
      InterfaceStateSubject
      InterfaceSyntaxAnalysis
      InterfaceXmlWrite
  ► Engine.System
  ► Language.Pack.Java
  ► Util.XmlParser
  ► Util.File
  ► Engine.Driver
  ► Language.Pack.Android
  ► Engine.XmlFile.Info
  ► Language.Pack.Csharp
  ► Engine.Gui
```

Fig. 4. Classes categorized by tree form

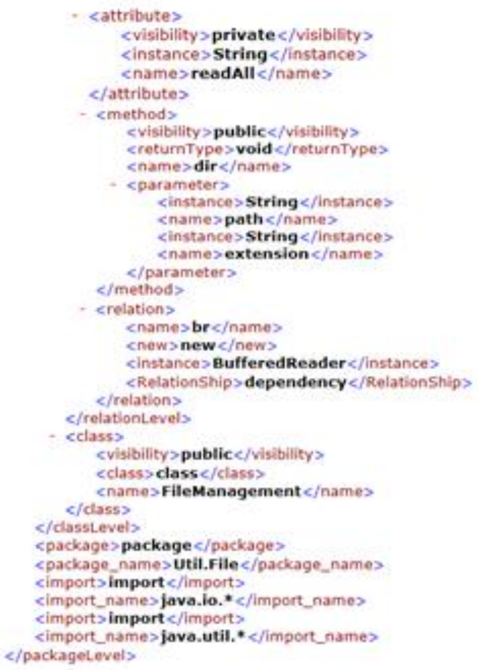
It looks more easy because it is classified package specifically, and when we click on the package we can see classes of package shown in Fig. 6.because it is stored in the tree form.
When we click certain class in Fig. 4, the analyzed contents are displayed as variables and methods shown in Fig. 5.



```
FileManagement
private int fileNameList_Pointer
public int String fileNameList
public int File fileList
private boolean File bool_fileCheck
private boolean String readAll
private boolean String eof
private int String read_Pointer
private int String for_Pointer
private int Integer for_Start_Index_List
private int Integer for_End_Index_List
----------------------------------------
public FileManagement  ()
public void dir  (String path String extension)
public void dirScan  (String path String extension)
public void clear  ()
public int getFileListSize  ()
public int getFileNameList String  ()
public int getNowFileNum String  ()
public int getNowFileName String  ()
public void dirCreate String  (String savePath)
public boolean static deleteDirectory String  (File path)
public void static fOpen throws FileNotFoundException  (String f_Name)
public void static fOpen throws FileNotFoundException  (int dir_fNum)
```

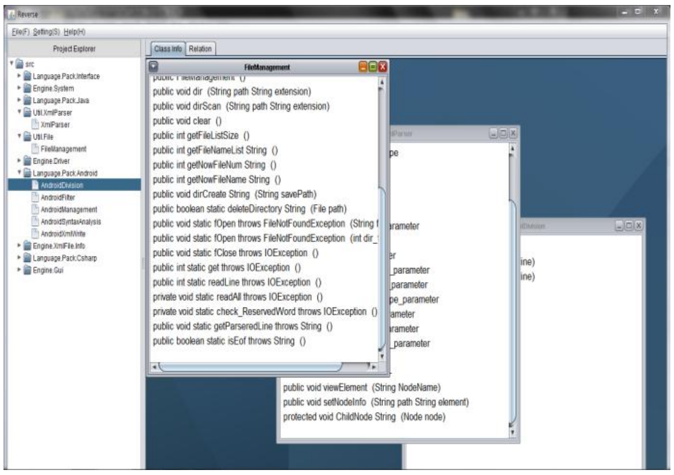
Fig. 5. Displayed variables and methods

Fig. 6 shows the saved XML values. Fig. 7 shows the GUI type of proposed reverse engineering system.



```
- <attribute>
      <visibility>private</visibility>
      <instance>String</instance>
      <name>readAll</name>
  </attribute>
- <method>
      <visibility>public</visibility>
      <returnType>void</returnType>
      <name>dir</name>
    - <parameter>
          <instance>String</instance>
          <name>path</name>
          <instance>String</instance>
          <name>extension</name>
      </parameter>
  </method>
- <relation>
      <name>br</name>
      <new>new</new>
      <instance>BufferedReader</instance>
      <RelationShip>dependency</RelationShip>
  </relation>
</relationLevel>
- <class>
    <visibility>public</visibility>
    <class>class</class>
    <name>FileManagement</name>
  </class>
</classLevel>
<package>package</package>
<package_name>Util.File</package_name>
<import>import</import>
<import_name>java.io.*</import_name>
<import>import</import>
<import_name>java.util.*</import_name>
</packageLevel>
```

Fig. 5. Values saved as XML



```
public void dir (String path String extension)
public void dirScan (String path String extension)
public void clear ()
public int getFileListSize ()
public int getFileNameList String ()
public int getNowFileNum String ()
public int getNowFileName String ()
public void dirCreate String (String savePath)
public boolean static deleteDirectory String (File path)
public void static fOpen throws FileNotFoundException (String f
public void static fOpen throws FileNotFoundException (int dir_
public void static fClose throws IOException ()
public int static get throws IOException ()
public int static readLine throws IOException ()
private void static readAll throws IOException ()
private void static check_ReservedWord throws IOException ()
public void static getParseredLine throws String ()
public boolean static isEof throws String ()
```

```
public void viewElement (String NodeName)
public void setNodeInfo (String path String element)
protected void ChildNode String (Node node)
```

Fig. 6. GUI type

IV. Conclusion

Reverse Engineering has an object to facilitate maintenance and increase reuse of software by extracting maximum of information necessary to source code through tracking reversely system. Also, it can decrease unnecessary waste of resource that consumers who buy a program and developers who develop the program. Reverse Engineering system which is proposed in this paper is composed of some parts, removing comments and strings to facilitate distinguishing reversed words, dividing up inner class and many of classes declared within one class into independent classes, analyzing a received line through hash map, saving value made by the analyzing part as XML document, and saving association through extracting saved XML. In addition, it can display such as GUI form.

V. Acknowledgements

The authors would like to thank uBASIC, Ltd for developing this system. Corresponding author: Soon-kak Kwon (skkwon@deu.ac.kr).

References

[1] T. Panas, J. Lundberg, W. Lowe, “Reuse in Reverse Engineering,” International Workshop on Program Comprehension, pp. 52-61, June 2004.

[2] M. Garg, M. K. Jindal, “Reverse Engineering – Roadmap to Effective Software Design,” International Journal of Recent Trends in Engineering, Vol 1, No. 2, pp. 186-188, May 2009.