Available online at www.ijpe-online.com

International Journal of Performability Engineering

vol. 15, no. 2, February 2019, pp. 637-644
DOI: 10.23940/ijpe.19.02.p28.637644

Approach to Discovering Reusable Components in Java Legacy System

Junfeng Zhao* and Yingjie Li

College of Computer Science Inner Mongolia University, Hohhot, 010021, China

Abstract

With the development of modern society, human beings are more and more aware of the importance of creativity. As a new area of research, creative computing will play a bigger role in life. There is no doubt that a large pool of knowledge is the foundation of creativity. Research shows that legacy systems contain not only a great deal of domain expertise but also many reusable components. From this point, a new component extraction approach is proposed to extract reusable components of the system and provide better foundation and inspiration for creativity. Experiments have shown that this approach can extract reusable components efficiently, which is helpful for creativity.

Keywords: creative computing; legacy system; reusable component

(Submitted on November 15, 2018; Revised on December 17, 2018; Accepted on January 10, 2019)

1. Introduction

The invention of the computer has provided many conveniences for our lives and work. However, as advanced computing technologies continue to emerge, companies are finding that quality environments and facilities are no longer core competitive advantages. In recent years, creativity has begun to attract more attention. As a new research field, creative computing began to play an important role in daily work and life. It can provide suggestions to support users' creativity and improve development efficiency.

There is no doubt that a large pool of knowledge is the foundation of creativity [1]. Only with sufficient knowledge can more creative ideas be generated. Surveys show that traditional software systems contain not only a large amount of domain knowledge, but also many reusable components with high utilization value [2]. Based on this discovery, component extraction of legacy systems will be performed using reverse engineering to achieve knowledge collection and reuse.

In Java legacy systems, different characteristics will affect the extraction process. In this paper, common development frameworks used by Java legacy systems are studied, and a new component extraction method is proposed. First, the system code is parsed to obtain concrete implementation information. Then, the framework usage of the system is analyzed. If the system uses a common development framework, we will use the method presented in this article for component extraction. Otherwise, we will use the improved Bunch clustering algorithm for component extraction [3].

This paper is divided into five parts. The related work is summarized in the second part. The third part describes the implementation of the component extraction approach. The implementation of the tool and the effect evaluation are introduced in the fourth part. The fifth part summarizes the work and puts forward the future research direction.

2. Related Work

For a long time, computers have played the role of assisting other disciplines. Can humans change their thinking and use the knowledge of other disciplines to solve problems in the computer field [4]? After continuous exploration, scholars have confirmed the feasibility of conjecture and represented this emerging field with "creative computing".

* Corresponding author.
E-mail address: cszjf@imu.edu.cn, 2769392302@qq.com

Hugill made a deeper definition of creative computing and put forward a theoretical framework to help researchers distinguish creative computing from computational creativity [5]. He subdivides creativity into historical creativity, psychological creativity, exploring creativity, and international creativity. Yang believes that rich knowledge provides thinking and vitality for creativity and also improves the development efficiency of programmers [6]. Researchers should strive to combine knowledge from different disciplines to help solve problems that cannot be solved during development. In [1], the current specific knowledge discovery approaches are introduced, and new knowledge discovery approaches are developed.

When Rasool studied porting legacy systems to the cloud, he found that legacy systems contained a lot of reusable value [2]. The idea of extracting system components through reverse engineering is popular in the research field. The implementation level entities in the system are divided into a collection and the components are expressed in collections [7-8]. These collections are the archetypes of early clustering. Rigi analyzed the system and extracted highly cohesive subsystems from the original system by means of clustering [9]. Sartipi et al. mined the association relationship and used a clustering algorithm to achieve component extraction [10]. By analyzing the advantages and disadvantages of various algorithms, a new collaborative approach was proposed to realize the mutual cooperation among different clusters [11]. A feature selection technique was proposed to supervise the clustering so as to improve the software quality [12].

In the Bunch method, the search algorithm is adopted to process the cluster, and the legacy system is divided into many subsystems based on the system code entities and relationship diagrams [13]. In an object-oriented implementation system, the cohesion of a class is affected by the different dependencies between classes. However, the module dependency graph (MDG) on which the Bunch approach relies does not distinguish different dependencies. Taking the above problems into consideration, [3] improved the Bunch approach and adopted R-MDG, which can reflect different dependencies, as the input to make the results more practical.

Component extraction is influenced by the characteristics of Java legacy systems. This paper analyzes the hierarchical structure and finds a special case: systems that use popular development frameworks have clear hierarchies, less correlation between functional modules, and easier extraction of reusable components. Based on this discovery, a new component extraction method is proposed and effectively combined with the original method discussed in [3]. Different methods are used to extract reusable components for different scenarios.

3. Component Extraction Approach

The component extraction approach will use two methods to handle different situations, which are shown in Figure 1.

For systems that do not use a development framework, we use the improved Bunch clustering algorithm for component extraction. For systems that use development frameworks, we propose a new method for component extraction. The analysis approach of development frameworks, clustering extraction method, extraction method for using development frameworks, and components generation process will be explained in detail below.

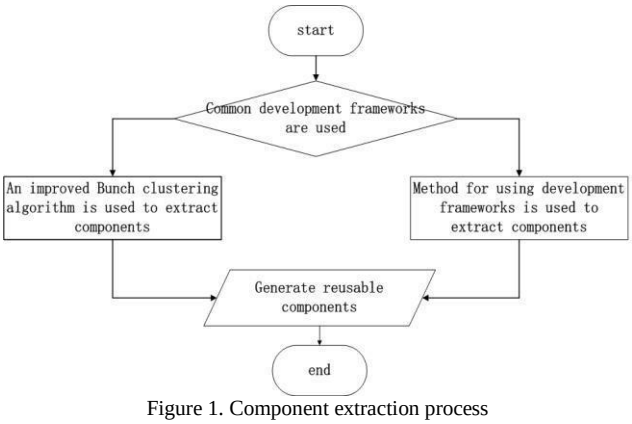

Figure 1. Component extraction process

3.1. Analysis of Framework Usage

The component extraction approach in this article is proposed for Java legacy systems. Therefore, we need to analyze framework usage and decide which methods to use for component extraction.

At the time of Java project development, there will be one classpath file in each project, which will record all the information of the project compilation environment, including the source file path, compiled class file store path, dependent jar package path, running container information, and external project of dependency. The detailed format of the classpath file is shown in Figure 2.

```
<classpath>
    <classpathentry kind="src" path="src"/>
    <classpathentry kind="con" path="org.eclipse.jst.j2ee.internal.web.container"/>
    <classpathentry kind="con" path="org.eclipse.jst.j2ee.internal.module.container">
        <attributes>
            <attribute name="owner.project.facets" value="jst.web;#system#"/>
        </attributes>
    </classpathentry>
    <classpathentry kind="lib" path="WebContent/WEB-INF/lib/antlr-2.7.6.jar"/>
    <classpathentry kind="lib" path="WebContent/WEB-INF/lib/asm-3.1.jar"/>
    <classpathentry kind="lib" path="WebContent/WEB-INF/lib/asm-commons-3.1.jar"/>
    <classpathentry kind="con" path="org.eclipse.jdt.launching.JRE_CONTAINER"/>
    <classpathentry kind="output" path="WebContent/WEB-INF/classes"/>
</classpath>
```

Figure 2. The classpath file format

The contents of the lib field in the file will be extracted and compared to the core Jar packages of the popular development framework. The comparison results will determine which method is ultimately used for component extraction.

3.2. Extraction Method for the System using Development Frameworks

The extraction method for using development frameworks consists of three steps. The first step is to regroup the classes in the system based on their association with each other. The second step divides the new group into modules according to the characteristics of the hierarchy.

3.2.1. Regroup

In object-oriented design, the relationships between classes are inheritance, implementation, composition, aggregation, association, and dependency. Different association relationships will result in different degrees of cohesion. Classes with inheritance or implementation relationships are more cohesive and often occur in the same functional modules. Classes with other associations are less coupled in modules and less related in concrete functions. Based on this discovery, classes with inheritance and implementation relationships are divided into the same cluster, while classes with other associations are arranged in separate clusters.

First, the system code is parsed to get concrete implementation information. For ease of understanding and use, we use nodes to represent classes, and attributes include node id and class name. Edge is used to represent the correlation between classes, and the attributes of edge include edge id, source node id, target node id, and relationship type. Based on the nature of the relationships between classes, use parent to represent inheritance and implementation relationships, and use association to represent other relationships. The detailed format is shown in Figure 3.

```
<graphml>
    <graph id="G" edgedefault="directed">
        <node id="19" name="com.ssh.dao.CommentDao" />
        <node id="20" name="com.ssh.dao.BaseDao" />
        <node id="21" name="com.ssh.dao.MenusDao" />
        <node id="22" name="com.ssh.dao.NoticeDao" />
        <node id="23" name="com.ssh.dao.UserDao" />
        <node id="24" name="com.ssh.daoImpl.BaseDaoImpl" />
        <node id="29" name="com.ssh.serviceImpl.CommentManagerImpl" />
        <edge id="27" source="19" target="20" relationType="parent" />
        <edge id="28" source="21" target="20" relationType="parent" />
        <edge id="29" source="22" target="20" relationType="parent" />
        <edge id="30" source="23" target="20" relationType="parent" />
        <edge id="31" source="24" target="20" relationType="parent" />
        <edge id="44" source="29" target="19" relationType="association" />
    </graph>
</graphml>
```

Figure 3. System implementation information

After the system implementation information is obtained, each node is analyzed. If a node has a father node, it refers to inheritance and implementation relationships. The node needs to be placed in a group with the parent node. The algorithm will recursively look up all the parent nodes of the node, store the found nodes in the linked list, and then ensure that all the ancestor nodes are traversed by constantly comparing the nodes in the linked list. If the node does not have a father node, it is arranged in a group alone. Finally, all the analyzed nodes are stored in a file in a specific form. The algorithm pseudo-code is shown in Figure 4.

```
  //read nodes
information if (Have
father node) {
        saveCurrentNode(currentNo
  de);
  getParent(currentNode,currentArr
  ay); if ( parentNode is not empty)
  {
                    Iterator<String> it =
          currentArray.iterator(); while (it.hasNext()) {
                                        String
                  parentName = it.next();
                                        if
                  (fatherNode is not in the list) {

                              yClass.put(parentNode);
                                        }

                  else
                    {
```

Figure 4. The pseudo-code of regrouping algorithm

After regrouping, in order to facilitate the division of modules, the original system implementation information needs to be modified. Instead of representing a single class, a node represents a set of classes and adds the inNumber and the outNumber to count the use of that node. Each group uses a class to represent the node, with the other classes represented later. Because classes that have inherited implemented relationships are divided into groups, there is no parent relationship between groups, only correlation. The final results are shown in Figure 5.

```
<node id="2" inNumber="2" outIumber="0" name="com.ssh.action.beans.ANotice" />
<node id="3" inNumber="6" outIumber="1" name="com.ssh.beans.Notice" />
<node id="4" inNumber="8" outIumber="0" name="com.ssh.beans.User" />
<node id="5" inNumber="3" outIumber="2" name="com.ssh.serviceImpl.NoticeManagerImpl[com.ssh.service.NoticeManager]" />
<edge id="1" source="1" target="2" relationType="association" />
<edge id="2" source="1" target="3" relationType="association" />
<edge id="3" source="1" target="4" relationType="association" />
<edge id="4" source="1" target="5" relationType="association" />
```

Figure 5. System implementation information after regrouping

3.2.2. Modularize

The new component extraction approach is proposed in particular cases where common development frameworks are used. When a system uses a common development architecture, its hierarchy can be summarized as control layer, logic layer, and data layer. The control layer is responsible for receiving and responding to user requests. The logical layer is called by the control layer and is responsible for executing the business logic. The data layer is at the bottom, called by the logic layer to achieve data persistence.

Based on these characteristics, tree structure is used to partition modules. Classes in the control layer call logical classes to realize business, which act as the root node of the subtree. The class of data layer will be called by other classes only, so it is used as the leaf node of the tree. First, the root nodes of all subtrees are found. Then, each subtree will be traversed from the root to the leaves, and the business components will be discovered. The pseudo-code of the module partitioning algorithm is shown in Figure 6.

```
//Find all subtree rootNodes
   for (Traverse the rootNode of the
subtree) { getParent(rootNode);
getSubclass(rootNode);
      //A HashSet is defined to store the nodes in each layer
HashSet<MyClassUtil> top = new HashSet<MyClassUtil>();
HashSet<MyClassUtil> middle = new
HashSet<MyClassUtil>(); HashSet<MyClassUtil> bottom =
new HashSet<MyClassUtil>();
      //Stratify by inNumber and
outIumber for (Traverse all
subclasses) {
                              if
                  (inNumber==0&&outIumber！=0)
                  top.add(node);
                              else if
                  (outIumber==0&&inNumber！=0)
                  bottom.add(node);
                    else
                                          middle.add(node);
                    }
      //Writes the layered node to a file
            for (Traverse all nodes in
         the top layer)
         { writeNodeToFile(node);
      }
            for (Traverse all nodes in the
         middle layer)
         { writeNodeToFile(node);
      }
```

Figure 6. The pseudo-code of modularizing algorithm

3.3. Extraction Method for the System not using Development Frameworks

The improved Bunch clustering algorithm is used to extract components from systems that do not use frameworks. In this method, the system implementation information is converted to R-MDG, a modular dependency graph that contains information of relational types. Then, the system is divided into modules according to cohesion, coupling, modularization quality, and cluster quantity. The vertical clustering approach is used to classify system functions, and the horizontal clustering approach is used to divide the system hierarchy. Finally, the results of the two clustering methods are analyzed to obtain reusable components. The detailed flowchart is shown in Figure 7.

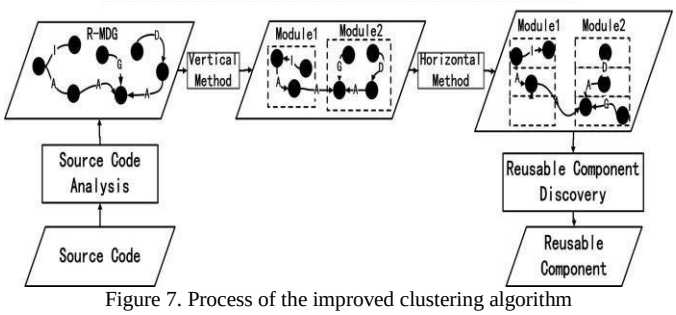

Figure 7. Process of the improved clustering algorithm

3.4. Generate Reusable Components

After the reusable components are obtained, the component-related implementation classes are analyzed. The description of the components can be obtained.

In the process of generating component descriptions, the description is obtained by analyzing the annotation information in the implementation class. However, different systems have different levels of detail. In this case, the annotations need to be analyzed in different ways to generate a more accurate description of the components. For systems

with detailed annotations, the annotations are extracted directly as component description information. For systems without detailed annotations, the identifiers in the implementation class are analyzed to generate component description information.

Reusable component description specifications are defined to describe components as more comprehensible knowledge. Figure 8 shows an example of reusable component description.

```
<reusablecomponent name="productOperation">
xmlns : xsi=http://www.w3.org/2001/XMLSchema-instance
xsi : noNamespaceSchemaLocation="reusablecomponent.xsd">
    <desc>The ProductProxy implemtents the operations related to product</desc>
    <types>
        <type name="Product">
        <attribute name="ID">java.lang.String</attribute>
        <attribute name="desc">java.lang.String</attribute>
        <attribute name="price">double</attribute>
        </type>
    <types/>
    <operation name="getProductByID">
        <note>search product by ID</note>
        <input> java.lang.String : ID</note>
        <output>Product</note>
    </operation>
</ reusablecomponent >
```

Figure 8. Example of reusable component description

As you can see from Figure 8, the component description is divided into three parts. The first part will express the specific functions of the component in words. The second part covers the entity types involved in the component. The third section describes the specific operations that a component contains. Through the introduction of these three parts, the information and knowledge contained in the components are fully interpreted, which provides a basis for further creativity generation.

4. Prototype Tool and Experiment

4.1. Prototype Tool

A prototype tool based on component extraction is presented. The prototype tool is implemented in Java and contains two extraction methods. One method is to extract components for systems that use frameworks. Another method is to use clustering algorithms to extract components for systems that do not use frameworks. The tool will first analyze frameworks used by the system and then select the appropriate methods for component extraction based on the analysis results.

When extracting the components of the system with frameworks, the new method takes less time than the clustering method and produces more accurate results. To verify the advantages of the new method, several case studies were conducted and compared with the clustering method. KBS is one of them.

4.2. Experiment Evaluation

KBS is an open source bulletin board system based on Java, SSH, and web interfaces. Characterized by stability, high performance, and simplicity, it is often adopted by colleges and universities to issue announcements. First, we wrap the project as a Jar package and analyze it using tools. The specific implementation information of KBS is obtained as shown in Figure 9. From Figure 9, we can see that the system contains 32 classes and 53 relationships.

```
<graph id="G" edgedefault="directed">
        <node id="1" name="com.ssh.action.AdminAction" />
        <node id="2" name="com.ssh.action.beans.ANotice" />
         - - - - - -
        <node id="31" name="com.ssh.serviceImpl.NoticeManagerImpl" />
        <node id="32" name="com.ssh.serviceImpl.UserManagerImpl" />
        <edge id="1" source="1" target="2" relationType="association" />
        <edge id="2" source="1" target="3" relationType="association" />
         - - - - - -
        <edge id="52" source="32" target="23" relationType="association" />
        <edge id="53" source="32" target="15" relationType="parent" />
</graph>
```

Figure 9. Realization information of electronic bulletin board system

Then, the improved Bunch clustering method and the new method are used to analyze KBS ten times. The analysis results are shown in Tables 1 and 2. It can be clearly found from the table that the new extraction method can obtain fewer aggregations, and correspondingly, the extracted components have greater granularity. At the same time, the new method consumes less time and is more efficient than the improved Bunch clustering method.

In order to verify the quality of the extracted components, KBS reference decomposition is obtained by analyzing the documents of KBS, which is shown in Table 3. MoJo is a measure of the distance between different clusters, and the similarity between two clusters is evaluated by calculating the minimum number of modules needed to move from one cluster to another [14]. The analysis results are compared with the obtained reference decomposition, and the results are shown in Table 4. According to the comparison results, when dealing with the system with framework, the new extraction method is closer to the original result than the improved Bunch clustering method.

Table 1. Results of KBS by the new method

Experiment times	Final module number	Time consuming (ms)
1	4	249
2	4	372
3	4	259
4	4	239
5	4	291
6	4	270
7	4	234
8	4	227
9	4	237
10	4	235

Table 2. Clustering results of the improved Bunch clustering method for KBS

Experiment times	Initial aggregation number	Final aggregation number	Time consuming (ms)
1	20	11	5858
2	20	10	4849
3	20	10	4448
4	20	10	5903
5	20	10	4255
6	20	11	4774
7	20	10	4727
8	20	10	4090
9	20	11	4126
10	20	11	5995

Table 3. Reference decomposition of KBS

Module name	Functional description	Class ID
Model1	Responsible for the audit and management of announcements. Browse the pending announcement, delete the illegal announcement.	1,2,3,4,5,20,22,24,27,31
Model 2	After the announcement, additional comments can be made. Responsible for review management.	3,4,5,6,7,8,9,19,20,22,24,25,27,29,31
Model 3	Responsible for user management. Including user registration, delete, modify user information, query all.	4,10,11,12,13,14,,20,21,23,24,26,28,30,32
Model 4	Responsible for issuing, editing and auditing announcements.	2,3,4,5,16,17,18,,,20,22,24,31

Table 4. The MOJO distance between the results and the reference decomposition

Result	Improved Bunch clustering method	New method
1st clustering result	13	6
2nd clustering result	14	5
3rd clustering result	12	4
4th clustering result	12	4
5th clustering result	15	6
6th clustering result	15	5
7th clustering result	14	4
8th clustering result	14	6
9th clustering result	15	7
10th clustering result	13	4
Average	13.7	5.1

5. Conclusions

Creativity is essential to everything. Creativity provides people with surprises and improves work efficiency. A wealth of knowledge is the foundation of creativity. Legacy systems contain a great deal of expertise and reusable components, which have high utilization value. Enterprises need to use specific approaches to extract reusable components from legacy systems.

This paper proposes a new component extraction approach. The approach contains two kinds of methods. According to the usage of frameworks, different methods are selected for component extraction. First, the characteristics of the system are obtained by analyzing the system. Then, a new extraction approach is proposed according to the system characteristics. Finally, the prototype tool is developed and combined with the improved Bunch clustering method. A case study is carried out. The results show that the new approach is more efficient and the results are more practical than the clustering method to the system using framework. In addition to the above positive conclusions, future works should also solve some problems. First, an accurate description of components should be added. Second, case studies of large-scale software systems should be carried out to test the effectiveness of complex systems.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (No. 61462066), Inner Mongolia Science and Technology Innovation Team of Cloud Computing and Software Engineering, and Inner Mongolia Application Technology Research and Development Funding Project "Mutual Creation Service Platform Research and Development Based on Service Optimizing and Operation Integrating".

References

1. L. Zhang and H. J. Yang, "Knowledge Discovery in Creative Computing for Creative Tasks," in *Proceedings of 1st Conference on Creativity in Intelligent Technologies and Data Science*, *CIT and DS 2015*, September 15-17, 2015
2. X. Meng and J. W. Shii, "Legacy Application Migration to Cloud," in *Proceedings of 2011 IEEE 4th International Conference on Cloud Computing*, *CLOUD 2011*, pp. 750-751, 2011
3. J. F. Zhao, J. T. Zhou, and H. J. Yang, "Modularizing Legacy System Through an Improved Bunch Clustering Approach in Cloud Migration," *International Journal of Grid Distribution Computing*, Vol. 8, No. 4, pp. 1-10, 2015
4. L. Zhang and H. J. Yang, "Definition, Research Scope and Challenges of Creative Computing," in *Proceedings of the 19th International Conference on Automation & Computing*, *Brunel University*, London, UK, September 13-14 2013
5. A. Hugill and H. J. Yang, "The Creative Turn: New Challenges for Computing," *Int. J. Creative Computing*, Vol. 1, No. 1, 2013
6. H. J. Yang, D. L. Jing, and L. Zhang, "Creative Computing: An Approach to Knowledge Combination for Creativity?" in *Proceedings of IEEE Symposium on Service-Oriented System Engineering*, May 2016
7. S. W. Kang, S. Lee, and D. Y. Lee, "Architecture Reconstruction: Tutorial on Reverse Engineering to the Architectural Level," *Software Engineering*, Vol. 5413, pp. 140-173, 2009
8. T. A. Wiggerts, "Using Clustering Algorithms in Legacy Systems Remodularization," in *Proceedings of 4th Working Conference on Reverse Engineering*, pp. 33-43, 1997
9. H. A. Muller, S. R. Tilley, and K. Wong, "Understanding Software Systems using Reverse Engineering Technology Perspectives from the Rigi Project," in *Proceedings of Conference of the Centre for Advanced Studies on Collaborative Research*: *Software Engineering*, pp. 217-226, 1993
10. K. Sartipi, K. Kontogiannis, and F. Mavaddat, "Design Recovery using Data Mining Techniques," in *Proceedings of European Conference on Software Maintenance and Reengineering*, pp. 129-139, 2000
11. R. Naseem, O. Maqbool, and S. Muhammad, "Cooperative Clustering for Software Modularization," *Journal of Systems and Software*, Vol. 86, No. 8, pp. 2045-2062, 2013
12. Z. Shah, R. Naseem, M. A. Orgun, A. Mahmood, and S. Shahzad, "Software Clustering using Automatedfeature Subset Selection," in *Proceedings of 9th International Conference on Advanced Data Mining and Applications*, pp. 47-58, 2013
13. S. Mancoridis, B. Mitchell, Y. Chen, and E. Gansner, "Bunch: A Clustering Tool for the Recovery and Maintenance of Software System Structures," in *Proceedings of IEEE International Conference on Software Maintenance* (*ICSM*), pp. 50-59, 1999
14. V. Tzerpos and R. C. Holt, "Mojo, A Distance Metric for Software Clustering," in *Proceedings of 6th Working Conference on Reverse Engineering*, pp. 187-193, 1999