Applying Input-Output Tree to the Implementation of a Rapid Prototyping Tool for Java Web Applications

CHUN-FENG HSIAO1,2 AND CHIH-PING CHU1
1Department of Computer Science and Engineering
National Cheng Kung University
Tainan, 701 Taiwan
2Department of Applied Informatics and Multimedia
Chia Nan University of Pharmacy and Science
Tainan, 717 Taiwan

A prototyping tool which facilitates the design of Java applets based on existing Java classes in the repository is developed and embedded into an eLearning platform. An input-output tree is proposed to graphically show the data flow from the input to output of a specific application. In the future, the possible direction of using such similar application as the proposed tool will assist the programmer to assemble the reusable software components from the repository, in which the source codes are generated automatically by means of automatic programming.

Keywords: automatic programming, CASE, input-output tree, rapid prototyping, software reuse

1. INTRODUCTION

Learning platforms adopted at most universities conform to SCORM (Sharable Content Object Reference Model) [1-3], which is a collection of standards and specifications for eLearning. Over the past 10 years, many eLearning courses have been developed. The assets of these online courses, including web pages, image files, sound files, video files, FLASH objects, and Java applet files, are stored on web servers.

A recent study by the authors [4] found that many simulations for eLearning are designed as special web services, and have been developed by expert programmers. In order to reduce the cost of developing web applications, existing assets and software components can be reused. Java and FLASH are primarily used for developing web applications for eLearning. Therefore, it is desirable for an authoring tool to provide the reuse of Java and FLASH code.

Many programmers are familiar with software revision control systems such as CVS. The process of reusing software components to accomplish a specific function is similar to the assembly of Lego™ building blocks. Knowledge management systems or configuration systems are required for managing the products of the projects and facilitating software reuse. Although a learning platform can be regarded as a knowledge management system, SCORM-compliant learning platforms do not support software reuse. Many universities make learning platforms accessible 24 hours a day. This yields the following questions. Is there any possibility to merge the learning platform with the function of software reuse? Can the learning platform automatically generate the prototype of
Java applet by assembling the Java classes in the repository? Our previous study [4] showed that a learning platform can be merged with software reuse. Automatically generating prototypes of Java applets by assembling the Java classes in the repository will help form a community that shares and reuses software components via the learning platform. For another, the related department of computer science does not need to purchase software configuration system installed on a stand-alone server. Furthermore, the goal of power saving can be accomplished as well.

In the present study, a prototyping tool, the Visual Online Prototyping Authoring Tool (VOPAT), which adopts input-output tree and facilitates the development of Java applets based on the existing Java classes in the repository, is proposed. For developers who lack coding experience, VOPAT helps them easily reuse Java applets to accelerate the development of their own Java applet. VOPAT generates a webpage based on Wikipedia, which helps rapidly generate a description of the project or application. VOPAT also offers the rapid prototyping of Java applets in a way similar to automatic programming to take advantage of reusable Java classes. The present study focuses on the development of VOPAT and its integration into a learning platform for automatically generating prototype Java applets, while the effect of the learning outcome is far beyond the topic since the users are likely to flexibly develop many kinds of Java applet prototypes through the VOPAT. To conveniently and easily prove the feasibility of JAVA class reuse, the range of the VOPAT is confined in the mathematical computation related domain in this study.

The rest of this article is divided into four sections as follows:

1. The related literature and the relevant work are reviewed.
2. The methodology of the VOPAT is described.
3. Implementation and test results of the VOPAT are presented.
4. Some final conclusions and suggestions for future work are elaborated in the closing part.

2. RELATED WORKS

This section briefly introduces software reuse, rapid prototyping, and RPC and CORBA. Then existing automatic programming systems are reviewed and compared to VOPAT.

2.1 Software Reuse

Software reuse enables developers to leverage past accomplishments and facilitates significant improvements in software productivity and quality [5, 6]. Software developers derive private benefits from writing software and sharing their code, and collectively contribute to the development of software [7]. Such private benefits include enjoyment, fun, learning, reputation, and community membership [8, 9]. In this study, software reuse is center on VOPAT. Consequently, we will survey the user’s satisfaction of improvement in software productivity, the users’ willingness of sharing their code, and the interaction among developers.
2.2 Rapid Prototyping

Rapid prototyping is a design methodology that has been successfully used in software engineering. Tripp and Bichelmeyer [10] indicated that rapid prototyping is a viable model for instructional systems design in a computer-based instruction context. They concluded that there are striking similarities between software engineering and instructional systems design. Two of the five potential advantages of rapid prototyping they stated are listed below:

1. Prototyping can increase creativity through quicker user feedback.
2. Prototyping accelerates the development cycle.

Shih [11] stated that many teachers at the secondary high school level might lack sufficient knowledge or skills to install or maintain a programming language in their own computers. For this reason, we introduce the VOPAT in this study that not only facilitates the reuse of the Java applet and but also generates the Java applet automatically based on the repository for developers lacking in the experience in coding.

2.3 RPC and CORBA

RPC (Remote Procedure Call) and CORBA (Common Object Request Broker Architecture) are used to develop the object-oriented distributed applications. CORBA enables separate pieces of software written in different languages and running on different computers to work with each other like a single application or set of services [12].

The paper in [13] described that there exist two main disadvantages to this code development paradigm in RPC and CORBA. First, it increases the code development time and cost. Second, it limits the development of distributed applications. The code development in RPC and CORBA is troublesome, time-consuming and error-prone.

2.4 Automatic Programming

In computer science, the term automatic programming identifies a type of computer programming in which some mechanism generates a computer program rather than have human programmers write the code [14]. Besides the generation of code from a wizard or template, IDEs (integrated development environment) such as Eclipse, Interface Builder and Microsoft Visual Studio can also generate and manipulate code to automate code refactorings that would require multiple (error prone) manual steps, thereby improving the developer’s productivity. The 15 automatic programming tools listed in [14] are either model-driven or template-based, and all of them are stand-alone software tools.

Barstow [15] used the following informal definition of an automatic programming system, which implicates that an automatic programming system must be domain-specific.

An automatic programming system allows a computationally naive user to describe problems using the natural terms and concepts of a domain with informality, imprecision and omission of details. An automatic programming system produces programs that run on real data to effect useful computations and that are reliable and efficient enough for
routine use.

In [16], Budinsky et al. presented a tool for generating design pattern code automatically from a small amount of user-supplied information. In addition, they also describe how the tool incorporates a hypertext rendition of Design Patterns to give designers an integrated online reference and development tool.

Bassil and Barbar [17] indicated that modern computer programming languages are governed by complex syntactic rules. Unlike natural language, they require extensive manual work and a significant amount of learning and practicing for an individual to become skilled at and to write correct programs. Bassil and Barbar proposed a new programming language and an environment for writing computer applications based on source-code generation. It is mainly a template-driven automatic natural imperative programming language.

In [18], Reformat et al. conducted an experiment on automatic programming using GP (Genetic Programming) algorithm for software clones. The experiment proved the possible usability of GP-based approach to automatic generation of clones. Kang et al. [19] investigated the representation of program for program reuse. They indicated that gene expression programming (GEP) [20] may have great significance and deep influences on the research of automatic programming in the future.

Fertalj and Brcic [21] presented an application generator based on UML specification and on templates written in XML/XSL. The generator accomplished the preserved flexibility towards the target programming language by code generation through two transformations; first into an intermediate code and then into the code of a selected target language by the specific template. Only the templates for C# and MSSQL had been produced in the application.

In the related works of automatic programming mentioned above, the user is required to download the specified software tool and install it on a PC, while the VOPAT, which works as the web service just like cloud computing, is working on the Internet. Furthermore, the VOPAT is not a source code generator but utilizes the Java classes in the repository as building blocks for assembling the Java applet prototype.

3. METHODOLOGY

This section will introduce the system architecture and the scenarios of usage. The Visual Online Prototyping Authoring Tool, as illustrated in Fig. 1, consists of two function units, the reusable class assembly function unit and the webpage editor.

3.1 System Architecture

Reusable class assembly function unit assembles the reusable Java classes. In order to assemble the software building blocks through the VOPAT, we need four basic components below.

(1) Repository: it is used to store the Java class or API.
(2) The Plug-in Repository Management System: it handles basic operations of the repository. More detailed description can be found in our previous study [4].
(3) The guidance document of the Java class or API: it offers the usage guide of the Java class and it follows the format of Java API documentation defined in [22]. In order to facilitate the search of reusable Java class, the source code should properly add the comments on the basis of the format in Table 1, in which this format of comment is defined by Sun Microsystems [22]. Then the PRMS will generate the corresponding Java doc as shown in Fig. 2. A well-formed Java doc is the critical factor for the successful operation of the reusable class assembly function unit.

(4) The transformation XML file: it describes the user requirements of the Java applet
which include the number of user interface template (input and output), the specification of input and output, the name of the label text, and the description of the process. The XML file serves as the input of the reusable class assembly function unit and its format is defined in Table 2. Because parsing the natural language sentences into an input-output tree is difficult to work in general if the requirements are getting more complicated, the user instead needs to type the separated keywords by semicolon representing the description of the process into the system user interface. In this way, it will be simpler for the VOPAT to construct the input-output tree and search for the reusable JAVA class.

3.2 The Input-Output tree

The reusable class assembly function unit will retrieve the keywords from the transformation XML file and then try to denote the program as an input-output tree illustrated in Fig. 3. The left side of Fig. 3 is the standard expression tree [23, 24] which behaves the deficiency of the ambiguous data type of each node. Without modification, the VOPAT will face the trouble to tell the data type of the node “number” which will cause the failure of finding the right method in the reusable class. Therefore, we propose the input-output tree as shown in the right side of Fig. 3 which will clearly show the data flow from input to output of a specific application with single GUI. In the input-output tree, the root node of the tree represents the output of the system, such as a parameter of the basic data type, an object, an array, and etc. The node in the $2^n$ level (where $n > 0$) of the tree represents the processing method or an operator (+, -, *, /, OR, AND, NOT…) or a subsystem. The node in the $(2^n - 1)$ level (where $n > 1$) of the tree represents the operand and is equivalent to the input parameter of a specific method. It reveals that from the modified expression tree at most three methods within some reusable classes in the repository will be required to assemble the user application in Table 2. For instance, the method “is prime” will need the input parameter of type integer to generate the output parameter of type boolean. Fig. 4 shows that the system user interface is composed of one or many input-output trees. The leaf node of the input-output tree represents the input parameter which is offered by the user or generated by the system (example: constant). We can regard the input-output tree as the representation of data flow from bottom (input) to top (output), which might be helpful to automatic programming in software reuse.
Table 2. The transformation XML file.

```xml
<requirement>
  <project_name> Prime and Odd</project_name>
  <templateUI_input>1</templateUI_input>
  <templateUI_output>1</templateUI_output>
  <main_class>
    <name> prime;odd </name>
    <attributes_list>
      <name> number </name>
      <datatype> integer </datatype>
    </attributes_list>
    ...
    <operations_list>
      <input_parameter>
        <parameter_name> InputNumber </parameter_name>
        <datatype> integer </datatype>
      </input_parameter>
      ...
      <output_parameter>
        <parameter_name> IsPrime </parameter_name>
        <datatype> boolean </datatype>
      </output_parameter>
    </operations_list>
    ...
    <button> check </button>
  </main_class>
  <process>
    <step id="1"> Test;number;prime;odd </step>
    ...
  </process>
</requirement>
```

3.3 The Set of Domain Characteristics for the VOPAT

It is sometimes difficult to judge if a potentially reusable component can be put into practice in a particular situation. To make it clarified, it is necessary to define a set of domain characteristics that are shared by all software within a domain [25]. Therefore, in this study, we define the set of characteristics in Table 3, which is used to identify the

Table 3. The set of domain characteristics for the VOPAT.

<table>
<thead>
<tr>
<th>Characteristics</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application domain</td>
<td>Does the reusable Java class belong to the application domain?</td>
</tr>
<tr>
<td>Method conformance</td>
<td>Does the method in the reusable Java class conform to the specification of input and output?</td>
</tr>
<tr>
<td>Process conformance</td>
<td>Does the method in the reusable Java class conform to the required process?</td>
</tr>
</tbody>
</table>
reusable component. After setting the adequate number of weight of each characteristic, the VOPAT will calculate the weighted number of the reusable component.

The second unit of VOPAT, the webpage editor, provides the user to edit the text of the application. In the unit, we use the open source software, the CKEditor [26], to achieve the goal of visual online and the system will first get access to the Wikipedia according to the keyword of the application as the original text, which the user can edit and refine later.

4. RESULTS AND CASE STUDIES

The user interface depicted in Fig. 5 is the entry of the rapid prototyping tool, which first lists all the reusable Java applets in the repository. Then the developer types the keyword to search precisely for what he/she demands and the result is shown in Fig. 6. The two buttons, “download class” (to download the class for reuse and extending) and “See API” (to view the Java API documentation for the Java applet class), support the advanced user to add functionality to existing Java applets. More discussion about them is elaborated in our recent study in [4]. The button “Class Information Template” offers the entry into the process of rapid prototyping. A new pop-up window, shown in Fig. 7, will be generated when we press this button.

In Fig. 7, the image (a) is the top half of the template and the image (b) is the bottom half. We can preview the existing Java applet and edit the learning content through the template.

Fig. 5 shows the scenario in which the user attempts to create a template-based application. VOPAT adopts the architecture of Model–View–Controller (MVC) and each image stands for each one of the three user interface templates. In order to prove the feasibility of reusing software components as the assembly of Lego™ building blocks, four test cases were prepared to test the VOPAT. Theses test cases offer the verification and validation of VOPAT and the results are shown as follow.
Fig. 7. (a)-(b) show the template for reusing the existing Java applet.

Fig. 8. The user interface templates for generating the Java applet prototype.

Fig. 9. The Applet Prototype Factory using template 2 (test case 1).

When the user selects the “Template 2” in Fig. 8, the user interface of the Applet Prototype Factory will appear on the screen as shown in Fig. 9. In test case 1, the user wants to calculate the area of a sector which the result will be rounded to a closest integer. The user interface will offer the input of radius and degree of a certain sector. After typing all the information in the Applet Prototype Factory and pressing the button named “Generate the prototype”, VOPAT will transform the requirement into an XML file. When the user chooses to know the detail information of the VOPAT, the system will present the information (package, method, return parameter, input parameter, the description of the method, and the weight of this method) as shown in Figs. 10 (a) and 10 (b). Fig. 11 (a) shows the simple information of the reusable Java class. When the user clicks the hyperlink of “Click to see the applet in a new window”, an applet prototype will later appear in a new pop-up window as shown in Fig. 11 (b). Eventually, the user can use the applet to calculate the area of a sector.

From the test case 1, it is proved that the VOPAT successfully assembles the reusable Java classes with the Java classes offered by JAVA JDK. Fig. 12 shows the test
Fig. 10. (a)-(b) show the detail information of the reusable Java class.

Fig. 11. (a) shows the simple information of the reusable Java class; (b) shows the result of the applet (test case 1).

Fig. 12. The Applet Prototype Factory using template 3 (test case 2).
case2 in which the user wants to calculate the surface area of a cylinder \((2\pi r^2 + 2\pi rh)\). The height of the cylinder is defined as 10 and therefore the input of radius is essential in this case. Fig. 13 (a) shows the simple information of the reusable Java classes and Fig. 13 (b) shows the input-output tree of this test case which is internally used in the VOPAT.

From the result of test case 2 shown in Figs. 14 (a)-(b), it is again proved the proper work of the VOPAT and this test case will represent the more complicated case in this study. Then the Fig. 15 shows the test case 3 in which the user needs to input two integers at random and the output will show the ratio of the maximum and the minimum. Fig. 16 (a) shows the simple information and Fig. 16 (b) shows the result.

The Fig. 17 shows the test case 4 in which the user wants to calculate the third power of an integer. Fig. 18 (a) shows the error message in this case. When the error message occurs, the user knows that the suitable Java class which can meet the requirement is not existent in the repository. Then the user may read the guide shown in Fig. 18 (b) to modify the requirement. VOPAT is not an automatic source code generator and as the error message emerges repeatedly, the user should code by self or call someone good at programming for help.

---

**Fig. 13.** (a) shows the simple information of the reusable Java class; (b) shows the input-output tree of this test case

---

**********Best Recommend Class: 1**********;
package edu.math.Area
method: float circleArea(float radius)
Description: Calculate the area of a circle when the radius is given.

**********Best Recommend Class: 2**********;
package edu.math.shapeLength
method: float circleLength(float radius)
Description: Calculate the length of a circle when the radius is known.

****.Linked List Node:
edu.math.Area
 circleArea
 float
 (float radius) Calculate the area of a circle when the radius is given.
 190.0

****.Linked List Node:
edu.math.shapeLength
circleLength
 float
 (float radius) Calculate the length of a circle when the radius is known.
 175.0

**Click to see the applet in a new window!**

---

**Fig. 14.** (a)-(b) show the result of the applet (test case 2).
Fig. 15. The Applet Prototype Factory using template 3 (test case 3).

Fig. 16. (a) shows the simple information of the reusable Java class; (b) shows the result of the applet (test case 3).

Fig. 17. The Applet Prototype Factory using template 1 (test case 4).
Fig. 18. (a) shows the error message of test case 4; (b) shows the guide of the Applet Factory.

5. CONCLUSION

From the four test cases and three templates, it is proved that the proposed prototyping tool VOPAT allows developers to reuse and assemble existing Java classes in the domain of mathematical computation. Although VOPAT is applicable to only small-scale applications and three templates in this moment, it can be used as a service of the cloud computing to develop small Java-applet-based projects and used as a knowledge management system of reusable Java classes. VOPAT is different from the approaches such as RPC and CORBA, because the operation of VOPAT is similar to automatic programming and it aims to help the users in school experience the benefit of software reuse.

The input-output tree proposed in this paper plays an important role in assembling the methods of reusable Java classes. It clearly presents the data flow from input to output in a specific user interface and let the VOPAT choose suitable methods in reusable Java class repository to fulfill the requirement of the user interface. Though the set of weighted number of domain characteristic is a subjective judge and may cause errors in some conditions, it is essential for the VOPAT to assemble Java building blocks. The study of suitable weighted number of domain characteristic will be conducted in the future.

In order to enhance the ability of assembling complicated reusable Java classes, some approaches of program representation are required, such as graph-based individual structures [27] or grammatical evolution [28]. In the future, VOPAT may allow the drawing of UML [29] diagrams online, giving users a convenient method of describing their applications. Besides, more templates can be developed and allow VOPAT to apply to other domains such as physics, chemistry, and so on. On the other hand, the input-output tree may offer the software developers more obviously information in the design of the user interface.

Since VOPAT is independent of eLearning platforms and eLearning standards, it is easy to deploy for supporting Java class or API repositories on eLearning platforms and for providing a software configuration system infrastructure that is embedded in the eLearning platform. In conclusion, the possible direction of using such similar application as the VOPAT will assist the programmer to assemble the reusable software components from the repository, in which the source codes are generated automatically by means of automatic programming.
REFERENCES


Chun-Feng Hsiao (惠淳豐) received the B.S. degree in Department of Computer Science and Information Engineering from National Chiao-Tung University, Taiwan, in June 1993 and the M.S. degree in Graduate Institute of Information and Computer Education from National Kaohsiung Normal University, Taiwan, in July 2001. Since September 2005, he had been studying towards the Ph.D. degree and he got a Ph.D. degree in the Department of Computer Science and Information Engineering, National Cheng Kung University, Taiwan in 2012. He is now an Assistant Professor in the Department of Applied Informatics and Multimedia, Chia Nan University of Pharmacy and Science Taiwan. His research interests include e-learning and software engineering.

Chih-Ping Chu (朱治平) received a B.S. degree in Agricultural Chemistry from National Chung Hsing University, Taiwan, an M.S. degree in Computer Science from the University of California, Riverside, and a Ph.D. degree in Computer Science from Louisiana State University. He is currently a Professor in the Department of Computer Science and Information Engineering of National Cheng Kung University, Taiwan. His research interests include parallelizing compilers, parallel computing, parallel processing, internet computing, DNA computing, and software engineering.