A Traceability Analysis Method for Event Driven System

KISHIMOTO Yorinori
Kobe Institute of Computing
Department of Information Systems
Kanou-cho 2-2-7, Chuo-ku, Kobe
Japan

Abstract: Web system specially requires high level maintainability because it needs frequent modifications to solving security problems and the latest information release. Corresponding between a program and its specifications is one of the elements of software quality for maintainability. This paper proposed a traceability analysis method between a program and its specifications for the event driven system that permits to formalize the program structure in regular expression considering to the state transition of the event driven system. It also discussed an application method and an example for its application to a HTML document with JavaScript program. As a result it can be checking correspondence between HTML document with JavaScript and its specifications in equations.

Key–Words: Software Engineering, Traceability, Program Structure Formalization

1 Introduction

The software development is required not only to develop the stand-alone system but also to develop the web system because of the widespread use of the Internet. Web system specially requires high level maintainability because it needs frequent modifications to solving security problems and the latest information release. Corresponding between a program and its specifications is one of the elements of software quality for maintainability. We called it ”traceability”[1]. Close traceability means a program is easy to modify and to correct because it can be easily grasped to the change point of source code corresponding to the change point of specifications. What is important in the web system is to keep close traceability between a program and its specifications.

As the method for an analysis traceability between a program and its specifications, we had proposed Program Structure Formalizing Technology(PSF)[2][3]. The basic idea of this technology is the program is consisting of the program structure and the program mechanisms. The program structure is non-deterministic to represent basically the scheme of a program by neglecting the propositions in the selections and the iterations and removing all program constructs which compose the mechanisms for all the propositions. The program structure represents the maximum framework of a program which is specified by the program mechanisms for the propositions. In addition, this technology proposed a program structure could be represented by regular expression and it shows traceability analysis between a program structure represented by regular expression to regular expression derived from its specifications. It has been proposed that an application method for procedural languages, but how to apply to web systems was unknown. In order to develop a close traceability web system, it is necessary to a PSF technology is applicable for this.

Many a web system is developed by the CGI program (e.g. Perl, PHP, and so on). However, these programs main function is just outputting a HTML document, and there is not concerned with behavior of the web browser which interpreted an HTML document including JavaScript programs. Specifications define to not only the server side program specification but also the web browser behavior concerned with the user interface. It requires to a traceability analysis method for the web browser behavior based on event driven framework. Therefore, I focus on the behavior of HTML document including a JavaScript program interpreted by web browser as an event driven framework.

I propose a formalizing method based on PSF which is applicable to the event handler realizing the event driven framework. On the basis of the state transition of event driven framework, the event handler can be interpreted as an iteration including selections. Therefore, PSF is applicable to event driven framework.

This paper proposes a traceability analysis method applicable to PSF for a HTML document including a JavaScript program as an event driven be-

havior. It also presents a practical application example analyzing traceability.

2 PSF

2.1 Basic Idea

The structure of the program is derived by disregarding the restrictions imposed by the conditional statements in the program. Three kinds of operators in regular expression can be used to represent the program construct sequence; $\cdot$ (AND) as concatenation, $+$ (OR) as selection, and Kleene Closures (*,$^+$) as iterations.

Some elementary constructs, such as an assign statement, for a program could have relationship with others. In this case, it is not necessarily true that transformation of the formulae in regular expression should result in possible implementation of the program. The concept of segregating the program structure from the program mechanism has been introduced in order to give solutions to such a problem as "context linkage."[2] The structure is nondeterministic to express the framework of a program, while the mechanism places the constraint to produce a specific program from the structure. In other words, a program comprises two parts; the structure and mechanism. Formalization in regular expression could be applied only to non-deterministic structures.

2.2 Conversion Formulae for Compulsive Control

Compulsive controls such as RETURN in program functions, BREAK in iterations are performed even in the well-structured programs. Compulsive controls shall be formulated in order to enable conversion of compulsive controls. Shown in Table 1 below are the conversion formulae for RETURNs and BREAKs in regular expression. The symbol Δ in Table 1 is an operator valid in compulsive controls.

3 An Idea of PSF Applicable to Event Driven System

The program structure represents the maximum framework of a program by neglecting the propositions in the selections and the iterations and removing all program constructs which compose the mechanisms for all the propositions. On the basis of this idea, the event handler as selections in a program and an occurrence of an event as it its proposition. Thus, the process concatenation path of event driven can be considered as an iteration including nonevent condition and all event handler.

All propositions in these selections and an iteration are not consideration because an application of PSF is neglected all propositions in the selection and the iterations. Therefore, PSF applicable to event driven behavior is set a continuous iteration ($()^\infty$) including all event handler process and keeps nonevent process (ε), and termination of an event handler process is the break of this iteration ($\rfloor$). It also the JavaScript program has the onload or onunload event handler. These event handlers are called by the specified timing of each programming language. Then, an application of PSF for these is put on these event handlers outside an iteration.

For example, JavaScript of an onload event is call at the beginning of a program. So, an application of PSF to this is following:

$$P \equiv a(b + c + \varepsilon\rfloor)^\infty d \tag{1}$$

where, a variable a has been called by onload event, and b ,c and $\varepsilon\rfloor$ has been called by other event handler. Variable d has been called by onunload event.

4 Application Method For JavaScript

4.1 Scope of Application

JavaScript is the prototype based object-oriented language. In JavaScript, a function may be used without instantiate just like as the function of procedural languages. So, a JavaScript program in HTML might be not only use as the object-oriented program but also use as the procedural language, and they have many uses as the procedural language. Then, application scope of this is a JavaScript program uses function as the procedural language, not use as object-oriented program (e.g. not using instantiate, inheritance and so on).

4.2 An Idea About Function Define

Functions of JavaScript program can be classed in three kind of definition. These functions are static definition as define in $\langle$ SCRIPT $\rangle$ tags, created by the function constructor and define the function literal. In the function of the static definition, PSF can be applicable to as the procedural language.

When the function constructer is used, a code is generated dynamically and it is compiled for run time, and it is not applied any static scope. A generated code by the function constructer is considering to the

Table 1 Conversion formulae for compulsive structures

Compulsive Construct		Symbol	Conversion Formulae
RETURN		$\#$	$\Delta_{\#}(f_E + f_R \# f_0) = f_E + f_R$
BREAK	Pre-tested	$\rfloor$	$\Delta_{\rfloor}(f_E + f_B \rfloor f_0)^* = f_E^*(\varepsilon + f_B)$
	Post-tested		$\Delta_{\rfloor}(f_E + f_B \rfloor f_0)^+ = f_E^*(f_E + f_B)$
	Continuous		$\Delta_{\rfloor}(f_E + f_B \rfloor f_0)^\infty = f_E^* f_B$

f_E,f_R,f_B : regular expression without $\#$ and $\rfloor$
f_0 : regular expression contains more than 0 number of $\#$ or $\rfloor$

Table 2 : How to formalize program constructs

JavaScript	PSF	Description
for	$(a)^*$	There are possibility of a continuous iteration by a proposition.
for/in	$(a)^*$	
while	$(a)^*$	
do-while	$(a)^+$	
if	$(a+b)$	
return	$\#$	The label definition is considered to the goto statement and excluded from this method.
break	$\rfloor$	
continue	$\S$	

external functions call because it is defined in a function constructer. Thus, using the function constructer can be applicable PSF as a function defined outside.

A JavaScript program could be defined a function including other functions. In this case, a scope of this function derived from including functions and it can be regarded to an individual function. Therefore, when application to PSF for JavaScript program, function is individual to deepest nested level and it keep each functions relationship.

4.3 Control Statement

In PSF, a program structure represented by the regular expression compound the concatenation, the selection and the iteration. In addition to PSF, it applicable to compulsive controls these are the break, the return and the continue. Table 2 shows corresponding control statements of JavaScript to PSF.

5 Traceability Analysis Method

Step 1:Structure separation

All functions defined in HTML document are separated. A program structure can be separated from the program by neglecting propositions in conditional statements and the element processes constructing the mechanisms.

Step 2:Structure formalization

The structures in functions can be represented by regular expressions with $\rfloor$ and/or $\#$ symbols and each function represented variable as alphabetic character.

Step 3:Event driven behavior formalizing

A function called for any event handlers are extracted from a HTML document. An order of event handlers executed by an explicit timing such as start time or the end is separated. Variables of other event handler processes and a keep nonevent process (ε) are contained in a continuous iteration and event handlers executed by an explicit timing are arrayed to executed timing in a regular expression.

Processes of quit to this event driven are addition to a break symbol ($\rfloor$) after these variables. A break symbol is added as a suffix to variables these are processes of quit to this event driven.

Step 4:Equation development

Some regular expressions including $\rfloor$ and/or $\#$ symbols could be developed applying the conversion formulae in Table 1.

Step 5:Specification arrangement

The specifications must be divided into several single descriptions in a compound one for checking the correspondences. It might be preferable to be applied such a formal method, as the state transition diagram.

Step 6:Traceability analysis

All of the terms in the regular expression can be checked to find the correspondences to the specification description. The more direct the correspondence would be detected; while the higher traceability would be.

6 An Example

A JavaScript program traceability analysis has been shown here. It is a program to decide the results from a total point of 3-choice question. It is an e-Learning content sample exhibit web site.

Step 1:Structure separation

Figure 1 shows a code of HTML document. The components named of lower case at a left side of this is a program structure element.

Step 2:Structure formalization

The following equations are the program structure of the code in Figure 1, and variables named by upper case are symbol of each functions.

Step 3:Event driven behavior formalizing

Upper case characters shown in the left of Figure 1 are correspondence to a symbol of functions called by event handlers. In this case, W is called by onload event and X is called by onunload event. Thus, W is set to the beginning of an iteration and X is set to after the termination of an iteration. Variable Y is called by onclick event. Thus, Y , a keep nonevent process (ε) and a quit window process ($\varepsilon\rfloor$) are set into a continuous iteration. As a result, this examples program structures are following equations.

$$\begin{aligned} P &= W(Y+\varepsilon+\varepsilon\rfloor)^{\infty}X \\ d &= (a\# + b\# + c\#) \\ W &= d(e+f) \\ X &= (g+\varepsilon) \\ Y &= (h(i+\varepsilon))^{*}(jk+lm)(nop+\varepsilon)q \end{aligned}$$

Here, d have $\#$ symbols. d applied the conversion formulae leads to the following.

$$d = (a\# + b\# + c\#) = (a+b+c)$$

A program structure of this example can be led by the equation development that focused on the function call relationship as follows.

$$\begin{aligned} P = {} & (a+b+c)(e+f)((h(i+\varepsilon))^{*}(jk \\ & +lm)(nop+\varepsilon)q+\varepsilon+\varepsilon\rfloor)^{\infty}(g+\varepsilon) \quad (2) \end{aligned}$$

```
                <html>
                <head>
                <script type="text/javascript">
                <!--
                //------------各設問の正解( 選択肢の番号) を設定----------------------
                var correct_response = new Array();
                correct_response[0] = 1;
                correct_response[1] = 2;
                correct_response[2] = 3;

                //------------各設問の配点を設定----------------------------------------
                var weight = new Array();
                weight[0] = 30;
                weight[1] = 30;
                weight[2] = 40;

                //------------合格点を設定----------------------------------------------
                var mastaryScore = 60;

                //------------APIアダプタオブジェクトを見つける----------------------
                var API = null;
                function FindAPI(win){
                  if((typeof(win.API)!="undefined")&&(win.API != null)){
a                   return win.API;
                  }else if(win.location == top.location){
b                   return null;
                  }else{
c                   return FindAPI(win.perent);
                  }
                }

                function MyInit(){
                  //APIフレームを見つける
d                 API = FindAPI(window);
                  if(API != null){
e                   API.LMSInitialize("");
                  }else{
f                   alert("APIが見つかりません。");
                  }
                }

                function MyFin(){
                  if(API != null){
                    //--------LMSFinish("")を呼び出す-----------------------------
g                   API.LMSFinish("");
                  }
                }

                function MyEvaluate(){
                  var myScore = 0;
                  var student_response;
                  var myStatus;
                  var msg;

                  for(var i=0; i<document.forms[0].elements.length; i++){
h                   student_response = document.forms[0].elements[i].selectedIndex+1;
i                   if(student_response == correct_response[i]){
                      myScore = myScore + weight[i];
                    }
                  }

                  //------合格点以上なら合格passed、未満なら不合格failedをmySourceに代入---
                  if(myScore >= mastaryScore){
j                   myStatus = "passed";
k                   msg = "合格です。";
                  }else{
l                   myStatus = "failed";
m                   msg = "不合格です。";
                  }

                  if(API != null){
                    //-----LMSにlesson_status、score.rawを送信する---------------
n                   API.LMSSetValue("cmi.core.lesson_status",myStatus);
o                   API.LMSSetValue("cmi.score.raw".String(myScore));
p                   API.LMSCommit("");
                  }

q                 alert(msg+"¥n得点 "+myScore);
                }

                //-->>
                </script>
                </head>
W,X             <body onload="MyInit();" onunload="MyFin();">
                <form name="question">
                <p>設問1． オーストラリアの首都はどこですか？<br>
                <select name="list0">
                <option>キャンベラ</option>
                <option>シドニー</option>
                <option>ローザンヌ</option>
                </select>
                </p>
                <p>設問2．ドイツの首都ベルリンとタイの首都バンコクでは
                どちらが人口が多いですか？<br>
                <select name="list1">
                <option>ベルリン</option>
                <option>バンコク</option>
                <option>ほぼ同じ</option>
                </select>
                </p>
                <p>設問3． フィリピンの首都マニラはどの島にありますか？<br>
                <select name="list2">
                <option>セブ島</option>
                <option>ミンダナオ島</option>
                <option>ルソン島</option>
                </select>
                </p>
Y               <p><input type="button" value="採点" onclick="MyEvaluate()"></p>
                </form>
                </body>
                </html>
```

Figure 1: 3-choice question program

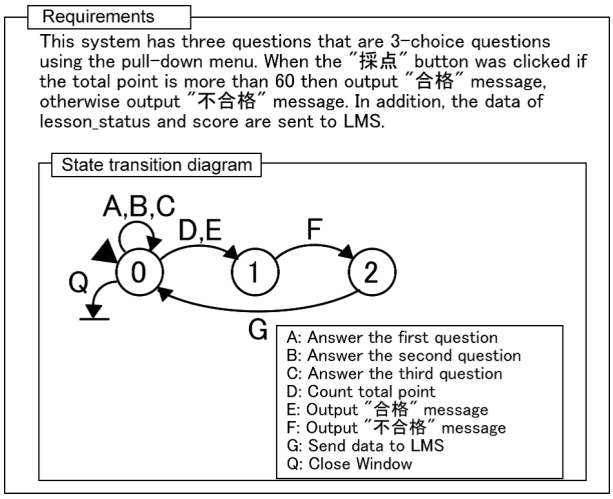

Figure 2: Specifications of 3-choice question program

Step 4:Equation development

The equation (2) have a brake and a continuous iteration. This equation applied the conversion formulae leads to the following.

$$\begin{aligned} P &= (a+b+c)(e+f)((h(i+\varepsilon))^*(jk \\ &\quad +lm)(nop+\varepsilon)q+\varepsilon+\varepsilon\rfloor)^\infty(g+\varepsilon) \\ &= (a+b+c)(e+f)((h(i+\varepsilon))^*(jk \\ &\quad +lm)(nop+\varepsilon)q+\varepsilon)^*(g+\varepsilon) \end{aligned} \quad (3)$$

Step 5:Specification arrangement

The abstract of specifications of this program has been shown in Figure 2. The operations for them specify the system status. The specifications for the system can then be considered a sequential machine with the alphabet of the operations. The machine is shown in Figure 2 in state transition diagram. The diagram is led to the following regular expression.

$$Sp = (A+B+C+D(E+F)G)^*Q \quad (4)$$

, where the variables A,B,C,D,E,F,G and Q are denoted in Figure 2.

Step 6:Traceability analysis

It show the traceability analysis between the equation (3) and the equation (4). Correspondences the equation (3) to the equation (4) are listed in Table 3. It identifies how the program structure is reflected its specifications in equations.

Table 3. Correspondences the structure to the specifications

Specifications(4)	Program structure(3)
D	$(h(i+\varepsilon))^*$
$(E+F)$	$(jk+lm),q$
G	$(nop+\varepsilon)$
Q	$(g+\varepsilon)$

The implementation of the output result message process has been separated into the process of deciding result and the process of the output message. It is advisable to concatenate $(jk+lm)$ and q because relationship of these is closeness. In this case, however, the equation $(nop+\varepsilon)$ has been inserted between $(jk+lm)$ and q, and it means the poor traceability point.

7 Concluding Remarks

It has proposed to a traceability analysis method for the event driven system that permits to formalize the program structure in regular expression considering to the state transition of the event driven system. It also presents that the application of the method to an example program appeared web site has made it clear that it has effective abilities in the practical fields.

It analyzes program traceability that is to explain how the program structures in a program reflect the specifications.

References:

[1] Satou Tadamasa, Kishimoto Yorinori, "An Analysis Method of Program Traceablity to the Specification through Program Construct Formalization" Advances in Systems Engineering, Signal Processing and Communications, 2002,pp19-24,

[2] Satou Tadamasa, Kishimoto Yorinori,"Program Code Analysis Focused on its Structure" WSEAS TRANSACTION on COMPUTERS, Issue 1, Volume 2, 2003, pp24-29

[3] Yorinori Kishimoto, Takako Itou, Tadamasa Satou, "Program Structure Formalizing Technology for Static Analysis" WSEAS TRANSACTIONS on SYSTEMS,Issue 4, Volume 3, 2004, pp1691-1698