BabelRef: Detection and Renaming Tool for Cross-Language Program Entities in Dynamic Web Applications

Hung Viet Nguyen, Hoan Anh Nguyen, Tung Thanh Nguyen, and Tien N. Nguyen
Department of Electrical and Computer Engineering
Iowa State University, USA
{*hungnv, hoan, tung, tien*}*@iastate.edu*

***Abstract*—In a dynamic web application, client-side code is often dynamically generated from server-side code. Client-side program entities such as HTML presentation elements and Javascript functions/variables are embedded within server-side string literals or variables' values. However, existing tools for code maintenance such as automatic renaming support only work for program entities in a single language on either the server side or the client side. In this paper, we introduce BabelRef, a novel tool that is able to automatically identify and rename client-side program entities and their references that are embedded within server-side code.**

***Keywords*-Refactoring; Web applications; Cross-language.**

I. Introduction

Dynamic web applications have become increasingly popular. Typically, a dynamic web application is written in multiple web languages such as PHP, ASP, and SQL on the server side, and HTML, Javascript (JS), and CSS on the client side. As a client-side request arrives, the corresponding server-side code is executed to *dynamically generate* client-side code, which is transferred to the client-side browser for execution. Program entities on the client side such as HTML web page elements or JS functions and variables are *embedded* within string literals or variables' values in the server code. Moreover, some HTML elements *can be referred to from other languages* such as PHP or JS. When a client-side program entity is renamed, all of its references including those embedded in one or multiple languages must be renamed consistently. Therefore, it is challenging to identify client-side program entities and their references, and provide automatic renaming support for them. Let us illustrate the challenges via a running example.

Figure 1 displays an example of a PHP server page, and Figure 2 shows the source code of the corresponding client page. As seen, the client page contains several HTML and JS program entities. For example, a JS function named loadPage is defined on lines 4-6 (Figure 2), and that JS code is generated from the PHP variable $script (lines 6-10, Figure 1). This function will be invoked on the onload event of the page (line 9, Figure 2), and the corresponding code is generated from line 20 of Figure 1. Similarly, the HTML input field username is defined on line 12 of Figure 2. Its HTML code is assigned to the PHP variable $input on lines 12-15 of Figure 1 using an if statement to specify different presentation styles for the element depending on the runtime input (e.g. the value of the parameter "role").

```
1 <?php
2 if (isset($_REQUEST['username']))
3   include 'Login.php';
4
5 $form_name = 'loginform';
6 $script = '<script type="text/javascript">
7   function loadPage() { // Sets username input's length
8     document.' .$form_name. '.username.setAttribute("maxlength", 20);
9   }
10  </script>';
11
12 if ($_REQUEST['role'] == 'admin')
13   $input = "<input name='username' ... blue;' />";
14 else
15   $input = "<input name='username' ... green;' />";
16
17 echo '<html><head>
18       ' . $script . '
19   </head>
20   <body onload="loadPage();">
21     <form name="loginform">
22       <label>Enter your name here:</label>
23       ' . $input . '
24       <input type="submit" value="Submit" />
25     </form> </body> </html>'; ?>
```

Figure 1. PHP Server-Side Page Example

```
1 <html>
2   <head>
3     <script type='text/javascript'>
4       function loadPage() { // Sets username input's length
5         document.loginform.username.setAttribute('maxlength', 20);
6       }
7     </script>
8   </head>
9   <body onload='loadPage();'>
10    <form name='loginform'>
11      <label>Enter your name here:</label>
12      <input name='username'...style='color:green;' />
13      <input type='submit' value='Submit' />
14    </form>
15  </body> </html>
```

Figure 2. HTML Client-Side Page Example

Suppose that the developer wants to rename these client-side entities: loadPage to load and username to user. To do that, all the references of these entities in the *server-side* PHP code must be renamed consistently. Specifically, in the PHP

code in Figure 1 these references include those referring to loadPage on lines 7 and 20 and those referring to username on lines 2, 8, 13, and 15.

Identifying the references to these *client-side* program entities in the *server-side* code is non-trivial:

1. First, the client-side program entities are *cross-language*. For instance, the JS function loadPage is invoked from an HTML attribute, whereas the HTML input username is accessed from PHP code (line 2) and JS code (line 8). More importantly, these entities and their references are *dynamically generated* from the server-side PHP code.

2. Second, the entities are often *embedded* in PHP string literals, which are usually *incomplete HTML code fragments*. Furthermore, the string fragments are created from *scattered* locations in the server code (different functions or files), and then manipulated and concatenated with other string fragments before being output to the client page.

3. Third, because server code can generate *multiple versions* of the client page depending on different inputs or runtime environments, client-side entities/references may appear in different execution paths in the server program. For example, Figure 1 can produce two versions corresponding to two *roles* of users (line 12). Therefore, although Figure 2 contains only two references to username, four references to it in Figure 1 must be renamed together, including two from different execution paths (lines 13 and 15).

Traditional program analysis tools [1] work on only a single language; thus, they are not able to identify the references to these cross-language entities. Moreover, since these entities are often *embedded* in *incomplete* HTML code, a regular HTML parser cannot be used to detect them. A simplistic approach using text search for the entity name will also be likely to report incorrect results because it does not consider the program's semantics (e.g., the text username in a comment on line 7 of Figure 1 is not a reference to the entity username).

II. BabelRef Approach

We introduce BabelRef, a novel tool for identifying and renaming cross-language, client-side HTML/JS program entities and their references in a PHP-based web application. BabelRef's entity detection algorithm is equipped with two important ideas. The first idea is to symbolically execute each PHP page and represent all possible generated client pages by a single tree-based structure called *D-model* [2], and detect all client-side entities and references directly in that D-model. Figure 3 shows the D-model of the PHP page in Figure 1. As seen, the client page is a concatenation (represented via a **Concat** node) of multiple HTML fragments (represented via leaf nodes). Two versions of the client page corresponding to the execution of the PHP if statement (lines 12-15, Figure 1) are represented by a **Select** node. The details of the symbolic execution on PHP code to create a D-model can be found in our prior work [2].

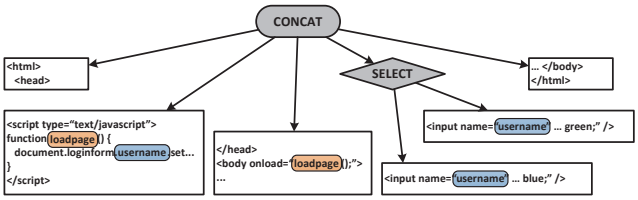

Figure 3. Example of a D-model

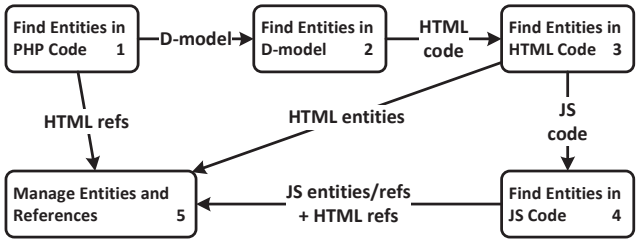

Figure 4. BabelRef's Procedural Overview

The second idea is to develop an HTML *partial parser* and use it to detect entities and references in the D-model by discovering the semantics of the HTML fragments contained in the D-model's leaf nodes. The parser maintains the program semantics across HTML fragments so that two references to the same entity are recognized even when they are contained in different fragments. For example, in Figure 3, the references to the entities loadPage and username are detected at various leaf nodes. Entities and references detected in the D-model are then mapped back to their locations in the PHP code using the location information that was established when the D-model was built.

Figure 4 gives the procedural overview of BabelRef. Given a PHP page, BabelRef generates a D-model representing its output (module 1) and traverses the D-model tree to detect client-side entities (module 2). At the D-model's leaf nodes, HTML fragments are sent to the partial parser where HTML entities will be detected (module 3). For example, given the HTML fragment <input name="username">, the parser parses it into an HTML input with "username" as the value of the attribute "name"; thus, BabelRef recognizes an HTML input entity named "username". Any JS code that is embedded in HTML <script> tags or HTML event handlers such as onload and onclick will also be extracted out for entity detection (module 4). HTML entities can also be accessed via special PHP variables (e.g., the variable $_REQUEST['username'] on line 2, Figure 1 refers to the HTML entity username). Such references are detected directly from PHP code when the program is symbolically executed in module 1. In all cases, BabelRef uses the entity management module for storing detected entities and references (module 5).

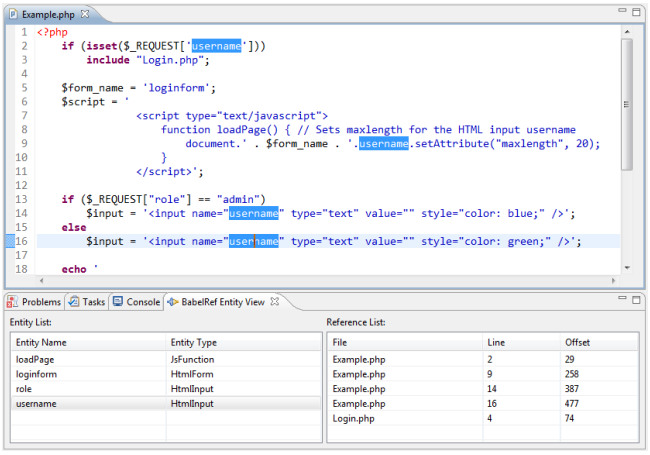

Figure 5. BabelRef's Entity View

III. BabelRef's Functionality

BabelRef is implemented as a plug-in to Eclipse development environment and provides two key features: detecting and displaying cross-language program entities/references and renaming those entities/references on request.

A. Entity Detection

Figure 5 shows BabelRef's entity view with an entity list and a reference list. The entity list displays all the cross-language program entities in the currently-edited PHP file. When an entity is selected, the reference list gives the location information of all the references to that entity, including the source files, line numbers, and offset positions. For example, the PHP file in Figure 5 contains four entities, three of which are HTML entities and the other is a JS function. The HTML entity username has five references located in two different PHP source files. As a user selects an entity in the editor window, all of its references in the file will be highlighted.

B. Entity Renaming

Based on the identified reference locations of the entities, BabelRef provides automatic renaming support on those entities. When the user right-clicks on an entity in the Eclipse editor, the BabelRef Rename command appears in the context menu allowing the user to rename the entity (Figure 6). The user can then enter a new name for the entity and preview the changes before applying the renaming operation (Figure 7).

After a renaming operation or whenever the user edits the source code, BabelRef automatically re-executes the PHP program symbolically, re-detects the entities in the background, and updates the entity and reference lists on the fly. In our experiments on several real-world web applications of size up to 50 KLOC, BabelRef normally took less than 40 seconds to perform symbolic execution and detect entities

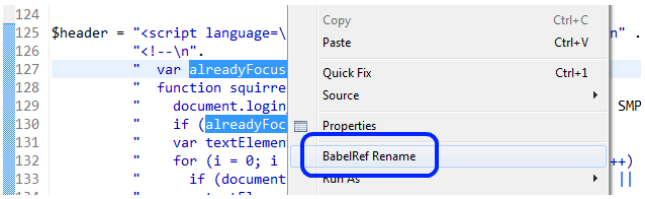

Figure 6. BabelRef's Entity Renaming: Selecting an entity to rename

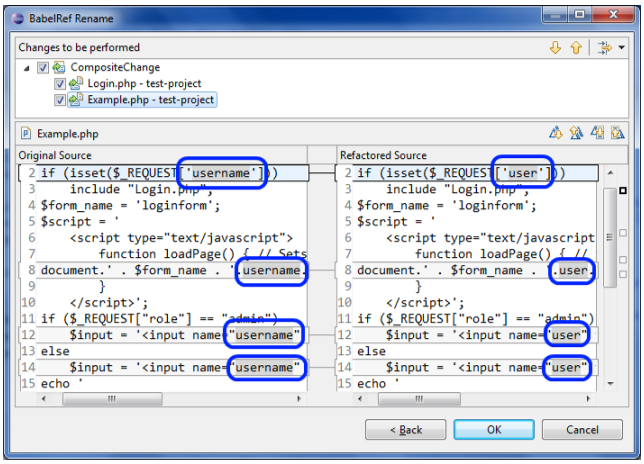

Figure 7. BabelRef's Entity Renaming: Previewing the changes

for a new system (with up to 300 entities and more than 2,000 references), and it took less than one second to re-detect entities and display the updated results when the source code changed.

IV. Interesting Case Studies

In SquirrelMail-1.4.22, alreadyFocused is a JS variable declared and used inside the JS code that is embedded in the PHP string $header (Figure 8a). The string value of $header will then be output to the client page. On the client page, there also exist two HTML elements produced by the PHP functions addInput and addPwField (Figure 8b). Using symbolic execution, BabelRef can construct the HTML code of these elements and detects that inside the event handlers for onfocus, the JS variable is accessed. Therefore, BabelRef recognizes all the four references of alreadyFocused.

In SchoolMate-1.5.4, there are two entities with the same name addstudent, one is an HTML form and the other is an HTML input field (Figure 9). If the user is interested in one of the entities only, BabelRef can help in identifying all the references belonging to the chosen entity. Thus, the user does not have to filter the results returned by a text search for the string "addstudent" to eliminate irrelevant references.

Also, entities may have references located at various locations. As can be seen in Figure 10, the references to the entity logout are scattered across more than 50 source files. If the user selects one of these references to rename, BabelRef will rename all the other references as well.

```
$header = "<script language=\"JavaScript\" type=\"text/javascript\">\n"
        "<!--\n".
        "  var alreadyFocused = false;\n".
        "  function squirrelmail_loginpage_onload() {\n".
        "    document.login_form.js_autodetect_results.value = '" . SMI
        "    if (alreadyFocused) return;\n".
        "    var textElements = 0;\n".
        "    for (i = 0; i < document.login_form.elements.length; i++)
        "      if (document.login_form.elements[i].type == \"text\" ||
        "        textElements++;\n".
```

a) References of alreadyFocused in JS code

```
            html_tag( 'td',
addInput($user, $value, 0, 0, ' onfocus="alreadyFocused=true;"'),
            'left', '', 'width="70%"' )
            ) . "\n" .
            html_tag( 'td',
addPwField($pw, null, ' onfocus="alreadyFocused=true;"').
addHidden('js_autodetect_results', SMPREF_JS_OFF).
```

b) References of alreadyFocused in HTML code

Figure 8. Cross-language entities/references in SquirrelMail-1.4.22

V. Related Work

Code refactoring has been an important part of software development. Mens *et al.* provide a comprehensive survey on various refactoring approaches [1]. However, traditional code refactoring approaches work for one individual language, while the code in a Web application contains embedded program elements in different languages. Moreover, there are very few approaches to cross-language renaming ([3], [4]). Kempf *et al.* [3] built an Eclipse plug-in for cross-language renaming between Java and Groovy code, a dynamically typed code running on the Java virtual machine and interacting with Java code. Sidler *et al.* [4] extended and completed that work to support renaming of elements defined in either language.

Minamide [5] developed a string analyzer to approximate the output client page of a PHP program via a context-free grammar. The technique requires a regular expression describing the input, whereas BabelRef performs symbolic execution for all possible inputs. Based on his work, Wang *et al.* [6] computed the constant strings visible from the browser for translation. In contrast, BabelRef uses an HTML partial parser to identify cross-language entities.

VI. Conclusions

We presented BabelRef, a supporting tool for detecting and renaming cross-language program entities. Our experiments on several real-world web applications showed that BabelRef is able to detect and rename HTML/JS entities and their references with high accuracy and time efficiency. In our future work, we plan to extend BabelRef to detect database entities as well.

Acknowledgements

This project is funded by the US National Science Foundation (NSF) CCF-1018600 grant. The first author was funded in part by a grant from the Vietnam Education Foundation (VEF).

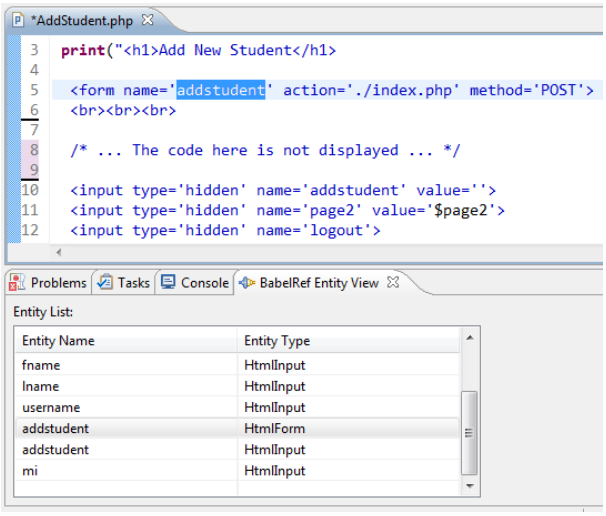

Figure 9. Entities with the same name in SchoolMate-1.5.4

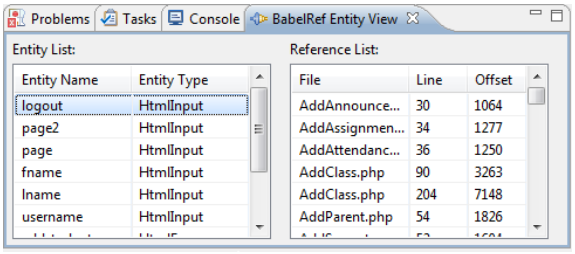

Figure 10. Entities with scattered references in SchoolMate-1.5.4

References

[1] T. Mens and T. Tourwé, "A survey of software refactoring," *IEEE Trans. Softw. Eng.*, vol. 30, no. 2, pp. 126–139, Feb. 2004.

[2] H. Nguyen, H. Nguyen, T. Nguyen, and T. Nguyen, "Auto-locating and fix-propagating for html validation errors to php server-side code," in *Automated Software Engineering (ASE), 2011 26th IEEE/ACM International Conference on*, nov. 2011, pp. 13 –22.

[3] M. Kempf, R. Kleeb, M. Klenk, and P. Sommerlad, "Cross language refactoring for eclipse plug-ins," in *Proceedings of the 2nd Workshop on Refactoring Tools*, ser. WRT '08, 2008.

[4] S. Sidler, S. Reinhard, and P. Sommerlad, "Cross language refactoring for groovy and java in eclipse," in *Proceedings of the 3rd Workshop on Refactoring Tools*, ser. WRT '09. ACM, 2009.

[5] Y. Minamide, "Static approximation of dynamically generated web pages," in *Proceedings of the 14th international conference on World Wide Web*, ser. WWW '05, 2005, pp. 432–441.

[6] X. Wang, L. Zhang, T. Xie, H. Mei, and J. Sun, "Locating need-to-translate constant strings in web applications," in *Proceedings of the eighteenth ACM SIGSOFT international symposium on Foundations of software engineering*, ser. FSE '10, 2010, pp. 87–96.