AOJS: Aspect-Oriented JavaScript Programming Framework [1]

Hironori Washizaki, Atsuto Kubo, Tomohiko Mizumachi,
Kazuki Eguchi, Yoshiaki Fukazawa[2]

Dept. Computer Science and Engineering, Waseda University
3-4-1, Okubo, Shinjuku-ku, Tokyo, 169-8555, Japan

Nobukazu Yoshioka, Hideyuki Kanuka, Toshihiro Kodaka,
Nobuhide Sugimoto, Yoichi Nagai, and Rieko Yamamoto

National Institute of Infomatics, Hitachi, Ltd., Fujitsu Laboratories ltd.,
Toshiba Solutions Corporation, NEC Corporation, and Fujitsu Laboratories ltd.

JavaScript (ECMAScript[1]) is a popular scripting language that is particularly useful for client-side programming together with HTML/XML on the Web. In JavaScript programming, there are many concerns (such as logging and Ajax-based functions) that cannot be encapsulated and separated into independent modules due to the limitation of JavaScript's modularization mechanism. For example, when conducting a beta-test of a typical web application with JavaScript program, it might be necessary to log all value changes of specific variables and send each log to remote sever at runtime because of variety of web client environments. Embedding remote-logging function codes into each location where variable substitutions will take place is the traditional way for realizing such logging function; however since the additional codes scatters and tangle with other concerns' codes, the maintainability of the program will decrease significantly.

To encapsulate and separate realizations of such crosscutting concerns into independent modules, there are several Aspect-Oriented Programming (AOP[2]) frameworks for JavaScript, such as Aspectjs[3] and Google Ajaxpect[4]. However, regarding all of conventional frameworks, it is necessary to modify the target program to include extended library and/or describe aspects in itself. Moreover none of conventional frameworks can specify the location where variable substitutions take place as joinpoints for weaving JavaScript codes.

[1] This paper has been accepted for demonstration at AOSD 2009. The extended content is under review for 8th AOSD Workshop on Aspects, Components, and Patterns for Infrastructure Software.

[2] Contact to Hironori Washizaki, Email: washizaki@waseda.jp

To solves these problems, we propose an Aspect-Oriented JavaScript programming framework, named "AOJS". AOJS is based on the proxy architecture for weaving aspects and the joinpoint model[5] for specifying program locations where the aspects will be weaved.

Firstly, AOJS realize the complete separation of aspects and target programs by the proxy-based runtime weaving. This design leads to the fact that AOJS always ensures the consistency between programs and the ones with aspects weaved by the proxy. Moreover the client does not have to consider AOP nor the weaving process when requesting web pages with JavaScript programs; it is only necessary to know the proxy's URL. Therefore it is easy to weave/remove aspects at runtime by only changing the URL for accessing. Figure 1(a) shows the architecture of AOJS. AOJS is realized as a server that communicates with web clients that request web pages via HTTP and web servers that store/provide original web pages and JavaScript programs. AOJS server consists of two parts: the reverse proxy for judging necessity of weaving and redirecting requests/outputs, and the weaver for weaving.

Secondly, AOJS allows programmers to specify any variable substitution, function execution and file initilization as a joinpoint by using corresponding pointcuts: `<var>`, `<function>` and `<initializeFile>` written in an aspect file in the form of XML. For the specified joinpoints, AOJS can weave both of before and after advices that will be performed before/after the target joinpoint's execution. Figure 1(b) shows the example of weaving when a variable substitution has been specified as a joinpoint. In the figure, the code portion surrounded by the dashed line will be replaced by the code for replacement based on the template. These replacements add new behavior into the target program while keeping the original functionality.

We conducted some experimental evaluations regarding the functionality and runtime performance of AOJS, and confirmed that AOJS has enough ability to specify joinpoints including variable substitutions. Moreover, we also confirmed the runtime performance can be improved to a practical level by adding a cash proxy in front of the reverse proxy.

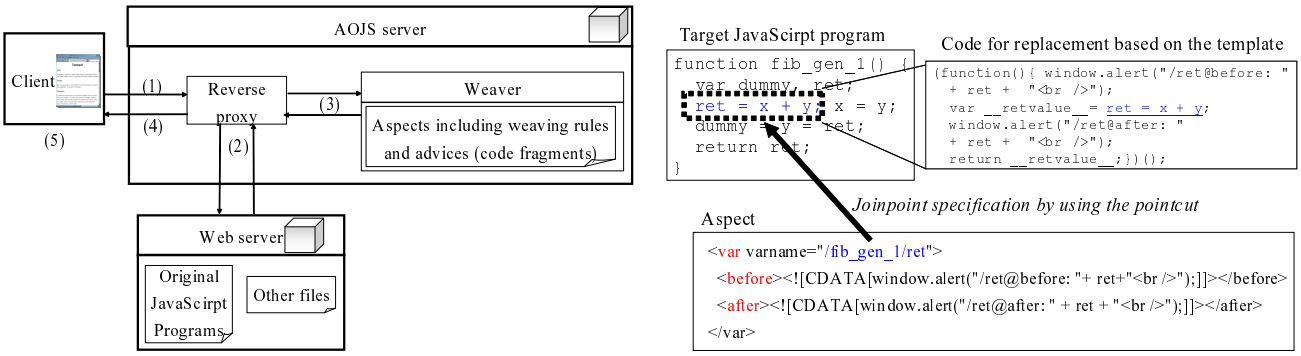

Fig. 1. (a) Architecture of AOJS (b) Weaving mechanism for variable substitution

References

[1] ISO/IEC 16262:2002, Information technology - ECMAScript language specification, 2002.
[2] Gregor Kiczales, et al.: Aspect-oriented programming, Proc. European Conference on Object-Oriented Programming (ECOOP), pp.220–242, 1997.
[3] LECACHEUR Sebastien: Aspectjs, `http://zer0.free.fr/aspectjs/`
[4] Google: Ajaxpect: Aspect-Oriented Programming for Ajax, `http://code.google.com/p/ajaxpect/`
[5] Gregor Kiczales, et al.: An Overview of AspectJ, Proc. European Conference on Object-Oriented Programming (ECOOP), pp.327–353, 2001.