
Ghita Kouadri Mostefaoui, University Campus Milton Keynes, Buckinghamshire, UK
Zakaria Maamar, Zayed University, Dubai, United Arab Emirates
Nanjangud C. Narendra, IBM India Software Lab, Bangalore, India

ABSTRACT

This paper discusses Aspect-Oriented Programming (AOP) as an efficient way to handle security concerns in Web services. Without AOP, the necessary security code would be mixed with the business logic that a Web service implements. This renders the maintenance of both code and business logic tedious and prone to errors. AOP allows confining codes of non-functional concerns like security and self-healing into specific modules so that they do not cross-cut with the Web service’s business logic. The proposed aspect-oriented approach in this paper is built upon three levels referred to as user, component, and resource, and adopts three types of context, one context per level. The contexts contain various details on the environment of Web services, which permits activating the necessary aspects in response to these details. A set of experiments validating this approach are also reported in this paper.

Keywords: Aspect-Oriented-Programming, Aspectj, Cross-Cutting Concern, Security, Spring AOP, Web Services

INTRODUCTION

Web services are hailed for their capacities in developing loosely-coupled business processes that can spread over organizations’ boundaries (Papazoglou et al., 2007). Over the last few years several efforts have been put into boosting the acceptance of Web services among the IT community by making them the technology of choice when developing such processes. However several still have doubts about Web services’ capacities when it comes to addressing critical concerns like security (Singhal, 2008) and fault tolerance (Mougouei et al., 2012). In this paper we focus on security as part of our SC-WS project standing for Security Concerns of Web Services.

For a proper handling of security we advocate for a clear separation between the business logic that underpins a Web service operation.

DOI: 10.4018/ijoci.2014040102
and the security requirements that restrict this operation. This handling is rendered possible using Aspect-Oriented Programming (AOP, (Kleinschmager, 2012)). AOP allows confining a concern code into one single module without making it scatter over other modules of a system. This helps minimize maintenance efforts in the case of any new requirement arises. Examples of initiatives adopting AOP for Web services include Azzam et al. (2012) and Shanmuganeethi et al. (2012).

The proposed SC-WS approach to handle security concerns is built upon three levels referred to as resource, component, and composite. The status of each level in terms of actions taken and techniques adopted is reported in a specific structure know as context. In this approach these concerns are defined as aspects. On top of the three contexts, a dedicated context associated with security is also defined. It is a state of the working environment that requires taking one or more actions. The rest of this paper is organized as follows. In the next section the security-based, aspect-oriented approach is presented in terms of architecture, security aspect configuration, and proof-of-concept. Prior to drawing some conclusions a set of related work is discussed.

THE SC-WS APPROACH

General Architecture

Taking into account the context of Web services has been proven mandatory when taking into account the characteristics of the environment in which these Web services operate (Maamar et al., 2006). These characteristics are multiple and can refer to users (e.g., stationary versus on the move), level of expertise (e.g., expert versus novice), computing resources (e.g., fixed versus mobile), time of the day (e.g., in the afternoon versus in the morning), to mention just a few.

Figure 1 presents the way we handle aspects in the SC-WS approach. Three levels of abstraction exist: user, Web service, and resource. The constituents of each level are tracked using specific contexts, for instance U-context, W-context, and R-context. The connection between user, Web service, and resource levels is implemented with “invokes” and “operates upon” relationships, respectively. Some key features of the SC-WS approach are as follows: multi-level concern separation using aspects and contextual tracking of the security requirements of Web services.

The Web service level refers to the context-aware Web service. A Web service is split into two parts: business logic and aspects. On the one hand the business-logic part refers to the actions that a Web service carries out as part of the functionality it offers (e.g., query data). On the other hand the aspect part refers to non-functional requirements that manifest themselves as cross-cutting concerns affecting the actions and interactions of the Web service such as security and logging.

The resource level is about context-aware resources. Resources represent the computing means upon which Web services operate. The scheduling of execution requests of Web services is prioritized when enough resources are not available to satisfy them all at once.

Configuring Security Aspects

Figure 2 illustrates the operation of the SC-WS approach. This operation consists of selecting the necessary security aspects that will protect the whole Web services environment (these aspects are referred to as active in Figure 2). The selection process combines context and policies. The former offers details on the environment including users, Web services, and resources, and policies trigger the appropriate security aspects based on these details. In addition to W/R/U-contexts we suggest a, security context (S-context in Figure 2). It is fed with details from W/R/U-contexts and is involved when triggering policies for weaving active security aspects. According to Kouadri Mostefaoui, a security context is a state of the working environment that requires taking one or more security actions (Kouadri Mostefaoui, 2004). In the following and relying on a previous work in (Sattanathan
An Autonomous Agent Approach to Query Optimization in Stream Grids
Saikat Mukherjee, Srinath Srinivasa and Krithi Ramamritham (2012). Intelligent and Knowledge-Based Computing for Business and Organizational Advancements (pp. 276-299).
www.igi-global.com/chapter/autonomous-agent-approach-query-optimization/65799?camid=4v1a