Semantic Cloud: Building Dynamic Mashup in Cloud Environment

Abdelhamid Malki, Djillali Liabes University of Sidi Bel Abbes, Sidi Bel Abbes, Algeria

Sidi Mohammed Benslimane, Djillali Liabes University of Sidi Bel Abbes, Sidi Bel Abbes, Algeria

ABSTRACT

Mashups allowed a significant advance in the automation of interactions between applications and Web resources. In particular, the combination of web Application Programming Interfaces (APIs) is seen as a strength, which can meet the complex needs by combining the functionality and data from multiple services within a single Mashup application. Automating the process of building Mashup based mainly on the Semantics Web APIs which facilitate to the developer their selection and matching. In this paper, we introduce reference architecture with six layers representing the main functional blocks for annotating, combining and deploying Web APIs in Cloud environment. We introduce Semantic Annotation for Web Application Description Language (SAWADL), an extension of the Web Application Description Language (WADL) that allows the semantization of the REST Web Service. The proposed architecture uses the Cloud Computing technology as a promising solution to increase the number of public API and therefore making the engineering process of Mashup applications more agile and more flexible.

Keywords: Application Programming Interfaces (API), Cloud Computing, Matching, Representational State Transfer (REST), Semantic Annotation for Web Application Description Language (SAWADL), Semantic Annotations for Web Services Description Language (SAWSDL), Semantic Mashup, Simple Object Access Protocol (SOAP)

INTRODUCTION

Dynamicity, agility and efficiency are concepts of the future. The World Wide Web is undergoing an evolution from a static environment to a dynamic world in which Mashups will play a central role. The Mashups are web applications developed by the combination of data, business logic, and/or user interfaces of web sources published and reused via APIs. Thus, Mashups are designed to reduce the cost and development time of web applications. Despite these advantages, engineering of Mashups applications requires the intervention of the developer which needs not only programming skills but also to understand the structure and semantics of APIs that wants to integrate. Currently, several tools Mashup (e.g., IBM WebSphere1, Convertigo2, Yahoo-pipes3, etc.) are used by end-users (i.e. with less programming skills) to facilitate the building of Mashup applications. However, the intervention of the

DOI: 10.4018/ijitwe.2013100102
professional developer is required when the application Mashup is complex.

In addition, the current Mashups applications are essentially static, i.e., the ingredients of Mashup application constitute a predefined set of APIs and services making Mashups applications less extensible, less dynamic and less fault-tolerance, especially if these software components become inaccessible for one reason or another. To remedy this problem, developers of Mashup are forced to manually seek and integrate APIs offering similar features to those that have become inaccessible.

Moreover, even if the problems of engineering and dynamicity of Mashups applications described above are resolved, the availability of a large number of APIs that meet the needs of a Mashup application is missing, this amounts at the cost of investment required by the owner of APIs on his infrastructure (i.e., server, storage and computing power, etc.) in order to ensure proper functioning of his business service (i.e., ERP, CRM, data warehouse, etc.) in the case of a utilization charge. One thing which prohibits service owners to open their business beyond their local environment. For example Google-Map and Amazon support their APIs (the most used in Mashup applications) by powerful data-center in order to ensure the quality of service (QoS) and remedy the load and complexity of exploitation, which is not feasible for medium and small businesses.

In order to address these deficiencies that impede agility and flexibility in the building of Mashups applications, we propose a new approach based on the use of semantics in order to make the building of Mashup more automatic and more dynamic, while strengthening the process of selection, discovery, and combination of APIs, the so-called “Semantic Mashups”. Semantic Mashup is a Mashup whose combined APIs are supported (or annotated) by a semantic layer that allows to select them and compose them in an automatic way (unambiguous).

However, the key element in the Semantic Mashup approach is that the semantization of APIs cannot be realized in a simple way. This is due to the heterogeneity of different types of APIs that build the Mashup application (e.g., SOAP, REST, JavaScript, etc.). The semantization of SOAP APIs is supported by a large number of approaches (e.g., OWL, WSMO, SAWSDL) satisfying all the problems related to the web services semantization. However, there are few works dealing semantization of REST APIs that are the most commonly used as an ingredient for applying Mashup. Therefore, we propose in this work SAWADL, a novel language for the semantization of REST web services. SAWADL uses WADL (Hadley, 2006) description to enrich RESTful APIs with a semantic layer that allows the discovery and automatic superposition of APIs in order to automatically build Mashup applications. SAWADL is more flexible and adaptive with respect to other approaches of semantization such as SAWSDL (Kopecký et al., 2007) which is used to annotate the WSDL description (Christensen et al., 2001) of SOAP web services (Box et al., 2000) with ontological concepts.

The semantization process is insufficient for the creation of Mashup Application, which requires other tasks that allows to select, and to compose relevant APIs (i.e., SOAP, and REST services) satisfying the user query. In our architecture the user query is represented by a set of user’s needs describing by their Inputs/Outputs. The selection of relevant APIs is performed by computing the similarity’s degree between the user query and each available APIs, then, the APIs that have the best degree are selected as relevant APIs. Based on these new set of APIs the composition process integrates and combines them based on the predecessor and successor relationship (i.e., an API $A_i$ is the predecessor of $A_j$ (or $A_j$ is the successor of $A_i$) iff the outputs of $A_i$ can be matched with the inputs of $A_j$. Moreover, we propose in our architecture a set of rules that are used to resolve the syntactic heterogeneities between REST and SOAP services semantized by SAWADL and SAWSDL language, respectively.

Thus in order to increase the number of APIs available on the web while reducing their operating costs and consequently enhance the process of building Mashup application by more

www.igi-global.com/e-resources/library-recommendation/?id=2

Related Content

Class Level Test Case Generation in Object Oriented Software Testing
www.igi-global.com/article/class-level-test-case-generation/2644?camid=4v1a

An Effective Recommender System Based on Clustering Technique for TED Talks
www.igi-global.com/article/an-effective-recommender-system-based-on-clustering-technique-for-ted-talks/241775?camid=4v1a
Get Out of My Sandbox: Web Publication, Authority, and Originality
www.igi-global.com/chapter/get-out-of-my-sandbox/137338?camid=4v1a

Farming 4.0: Technological Advances That Enable Smart Farming
www.igi-global.com/chapter/farming-40/233278?camid=4v1a