AN EFFICIENT AND RELIABLE PARAMETRIC APPROACH FOR WEB SERVICE COMPOSITION

Dheeraj Kumar¹, Vivek Jaglan², S.Srinivasan³

¹Research scholar (CSE)  Suresh Gyan vihar University, Jaipur, India.
²Research scholar (CSE)  Suresh Gyan vihar University, Jaipur, India.
³Dept. of Computer Applications, PDM  Bhadurgarh, India

ARTICLE INFO

Corresponding Author: Vivek Jaglan
Research scholar (CSE)  Suresh Gyan vihar University, Jaipur, India.
jaglanvivek@gmail.com

Keywords: Webservice, Composition, Deadlock, Efficient, Selection

INTRODUCTION

Web services have emerged as a major technology for deploying automated interactions between heterogeneous systems. They possess certain key properties, viz., independent from specific platforms and computing paradigms, developed primarily for inter-organizational situations, and composable into composite Web services. Web service composition primarily concerns requests of users that cannot be satisfied by any available Web service, whereas a composite service obtained by combining a set of available Web services might be used. The dynamic nature of the business world highlights the continuous pressure to reduce expenses, to increase revenues, to generate profits, and to remain competitive. This requires Web services to be highly reactive and adaptive. In particular, composite Web services should be equipped with mechanisms to ensure that their constituent component Web services are able to adapt to meet changing requirements. Figure 1 is showing the basic web service architecture.

From the viewpoint of the users of the composite Web service, such an approach enables on-demand adaptation with minimal disruption. To the best of our knowledge, this is the first non-intrusive distributed AOP mechanism, especially applied to Web services. Hence our main contributions are the following:

1. A distributed system, built on the well-known PROSE architecture, for non-functional properties adaptation of Web services via AOP methods.
2. A relationship model - expressed as a relation function - between the non-functional properties of the composite web service and those of its component web services, via a relation function.
3. Service-specific and non-intrusive concern extraction and manipulation implementation for component

Figure 1 : Web Service Architecture
A) **DIFFERENCE BETWEEN SERVICES AND WEB SERVICES** Though services and Web services are commonly used interchangeably in many situations, there exists basic distinction between the two. A service is the observable set of behaviors of a system accessible via a prescribed interface. "A service is a mechanism to enable access to one or more capabilities, where the access is provided using a prescribed interface and is exercised in consistent with constraints and policies as specified by the service description". A Web service is a specific type of service, describing the interface using the WSDL, using SOAP over HTTP as a transport protocol for example. A service is provided by an entity called the service provider for use by others and can be accessed by means of a service interface where the interface comprises the specifications of how to access the underlying capabilities. Services, much like components, are intended to be independent building blocks that collectively represent an application environment. But different to traditional components in the sense that, services have a number of unique characteristics that allow them to participate as part of a service-oriented architecture. One of these distinguishing features is complete autonomy from other services which means that each service is responsible for its own domain. This design approach results in the creation of isolated units of business functionality loosely bound together by a common compliance to a standard communications framework. As a result of the independence of services in this framework, the programming logic they encapsulate does not need to comply with any particular platform or technology set.

B) **WEB SERVICES INTERACTION PROBLEM**
Web services promise to allow businesses to adapt rapidly to changes in the business environment, and the needs of different customers. However, the rapid introduction of new services paired with the dynamicity of the business environment also leads to undesirable interactions that negatively impact service quality and user satisfaction. The root causes for feature interactions are:

- **Conflicting goals** (services with the same preconditions but incompatible goals are in conflict)
- **Competition for resources** (services compete with each other for limited resources, which need to be partitioned among the services)
- **Changing assumptions on services** (services make implicit assumptions about their operation, which can become invalid when new services are added)
- **Design evolution** (services need to be added to meet new customer needs, and the system will need to interoperate with other vendors' systems)

As the number of web services increases, their interactions will become more complex. Many of these interactions will be desirable, but others may be unexpected and undesirable, and we need to prevent their consequences from occurring. As noted by , many of them are related to security and privacy.

C) **WEBSERVICE COMPOSITION**
When running composite web services, each sub service can be considered autonomous. The user has no control over these services. Also the web services themselves are not reliable; the service provider may remove, change or update their services without giving notice to users. The reliability and fault tolerance is not well supported; faults may happen during the execution. Exception handling in the context of web services is still an open research issue.

II **LITERATURE SURVEY**
Mohammed Ketel defined WS which are limited and cannot satisfy some practical user requirements. The true potential can be achieved if WSs are used to dynamically compose some existing/new WSs. Moreover, WS technology is to be applied in various environments, such as wireless networks of relatively narrow bandwidth and unstable connections. In response to the mentioned problems Author adopt Mobile Agent technology and present a Mobile Agent framework for WSs integration. Author exploit the capabilities offered by Mobile Agents to invoke WSs without the need for simultaneous presence of the service requestor[1]. A formal service agent model is proposed, which integrates the web service and software agent technologies into one cohesive entity. Based on the service agent model, a distributed planning algorithm for web service composition called DPAWSC is presented. DPAWSC formalizes web service composition into a graph search problem according to the dependence relations among service agents[2]. A way to automatically compose web service workflow that uses component web services. The web services workflows are described using our transactional workflow ontology. The workflow ontology can be used to describe both component web service workflows and master web service workflows. Author have also implemented a workflow engine that runs the workflow instances[3]. An agent-based and context-oriented approach that supports the composition of Web Services. A Web service is an accessible application that other applications and humans can discover and invoke to satisfy multiple needs. To reduce the complexity featuring the composition of Web services, two concepts are put forward, namely, software agent and context[4]. The development of a distributed multi-agent workflow[5] enactment mechanism from a BPEL4WS[1] specification. This work demonstrates that a multi-agent protocol (LCC protocol)[10] can be derived from a BPEL4WS specification to enable business workflows using web services[2] composition computing paradigm[6]. The functional model and the architecture design of a platform for services composition and delivery. Mobile agents are able to access and compose services accessing them by standard interfaces. User authentication, service discovery and publication, personalization are other provided facilities. Technological interoperability is provided by Web Services technology[7]. Automatic SWSC enabling method for AgentSpeak agent. Firstly, conversion algorithm from OWL-S web service description to agent’s plan set (OWL2APS) is presented. Target service is converted to agent’s goal and related services are converted into agent’s plan set[8]. Based on semantic web extension, pragmatic web, Author propose pragmatic agent based QoS aware web service composition architecture. The detail structure and design process is elaborated and it is argued that this approach provide a new perspective on web service composition[9]. Proposes a multi-agent based semantic web service composition process. The process presented deals with some untouched issues in composition process. The work also presents a
CONCLUSION
We propose an approach for dynamic service selection and, which has the following advantages in comparison with previous approaches:

1. It hides the system’s complexity from the clients.
2. It provides a transparent service selection from the client’s point of view.
3. It assures a level of security, since the clients do not have direct access to the Web Services.

REFERENCES


[4] Zakaria Maamar,” Toward An Agent-Based And Context-Oriented Approach For Web Services Composition”, IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING 1041-4347/05 © 2005 IEEE


[8] Rocco Aversa,” Web Services Composition And Delivery Using A Mobile Agents Based Infrastructure”, Proceedings Of The Fifth International Symposium On Parallel And Distributed Computing (ISPDC’06) 0-7695-2638-1/06 © 2006 IEEE

[9] Xu Qixing,” Web Service Composition Based On Mobile Agent And Active Network”, 0-7803-9584-0/06 © 2006 IEEE
