Proposing an Evaluation Framework for B2B Web Services-based Transactions

Christos K. Georgiadis and Elias Pimenidis

Abstract — Web Services, due to their integration potential, are being enthusiastically utilized for joining business processes in business to business (B2B) transactions. However, aiming at spontaneously integrating services from different providers in ensuring a user the best possible experience, Web Services face considerable challenges in materialising that notion of improved experience. The interrelationships between the concepts of security, performance and interoperability, while critical to any Web Service, they pose the biggest test and constitute the key features of the proposed framework. The key objective of this framework is to establish a means of evaluating how close a Web Service can get to achieving best practice without one of the three key features overriding the others.

Index Terms — E-Commerce Technology, Evaluation, Transactions, Service-oriented Architecture, Web Services

1 INTRODUCTION

Service-oriented architecture (SOA) provides the proper infrastructure to handle successfully the interactions among a large number of heterogeneous components. In fact, the Web Services (WS) infrastructures have grown-up: first-generation systems are already widely adopted and WS providers are currently trying to interconnect their offerings in unpredicted manners, giving rise to more sophisticated WS-based functions. Thus, modern WS infrastructures may be considered as logical collections of WS whose publication and use depend on limitations characteristic of business service delivery. That is, current WS architectures aim to make explicit the notion of service procurement, separating it from that of conventional service supply and delivery [1].

The business environment in which WS evolve constrains service delivery. From the provider side, suppliers require regulations on how services are published, brokered, and repurposed through composition with other services. On the other hand, from the consumer side, business requirements restrict the way services are discovered, authenticated, monitored, and mediated.

This situation is of critical importance regarding e-commerce supporting technologies. From the technological aspect of B2B interactions, the flexibility of WS-based solutions provides a major advantage: strong integration between transaction activities and complementary enterprise processes. WS integrate applications owned by different organizations (even if they are developed in different programming languages and deployed on different platforms) to provide a loosely coupled architecture for building distributed systems with universal interoperability. As a result, WS have been widely adopted in the industry as a standard platform-independent middleware technology [2]. Cooperative services are capable of intelligent interaction and are able to discover and negotiate with each other, mediate on behalf of their users and compose themselves into more complex services. These exact properties of WS are the ones that pose most challenges and raise the issue of guaranteeing a given “quality” of services to final users. This concept of quality is expressed in terms of functional and non-functional requirements, such as performance or security and it is these two particular attributes that attract researchers in their quest for means of evaluating WS.

2 BACKGROUND — RELATE WORK

In reviewing the work of other researchers in the field of WS evaluation, the authors find some common threads in most published work. Most researchers focus on a common belief is that interoperability of WS must come along with considerable performance penalty [3], [4]. Thus almost all of the proposed models or frameworks used to evaluate some aspects of WS concentrate on the following three factors:

- C.K. Georgiadis is with the Department of Applied Informatics, University of Macedonia, Thessaloniki, Greece. E-mail: geor@uom.gr.
- E. Pimenidis is with the School of Computing and Technology, University of East London, UK. E-mail: E.Pimenidis@uel.ac.uk.
- Interoperability,
- Security and
- Performance.

One common finding is that all three could be affected by the automatic choice of partners in forming a WS and that all three could mutually affect each other.

Casola et al. [5] argue that although a service provider is able to guarantee a predefined service level and a certain security level, the mere nature of service integration and interoperability and the utilisation of up to date technologies does not allow for automatic measurement of such attributes. The most common approach amongst service providers is the “Service Level Agreement” (SLA). At the state of the art, SLAs are primary related to the quality of service and not to security. To reach the aim of SLA dynamic management to support the interoperability among services, a formalized approach both for defining the different quality factors of a service in a SLA, and for evaluating them automatically is required. Policy languages are starting to gain popularity, but they do not specify how to automatically evaluate and compare the related security and quality provisions [5], [6].

Casola et al. [5] propose a two component framework for evaluating WS security. Their framework consists of a SLA policy meta-model providing a flexible approach to define quality criteria and formalize them in an unambiguous way by policies. The policy meta-model can be instantiated by Customers or Providers in order to define the specific Quality Models on which their policies will be based. While Chen et al. [7] remain sceptical in that while WS security enhances the security of WS, it may also introduce additional performance overheads to standard WS due to additional CPU processing and larger messages transferred. They propose to extend their existing WS performance model by taking the extra WS security overheads into account.

The most attractive feature of WS is its interoperability in a heterogeneous environment, and exposing existing applications as a WS increases their reach to different client types. A further advantage is that WS have flexible communication patterns, as an invocation to a WS can be either synchronous or asynchronous. Liu and Gordon [8] focus on the combination of component technologies with WS which further extends the set of architecture choices, with each providing different levels of performance. Thus it is important for an architect to understand these performance implications in the early stages of the architecture design. They analyse the performance of four alternative architectures using J2EE and WS, concluding that WS-based architecture has significant performance degradation due to the large amount of service time spent on parsing XML information in a WS Description Language (WSDL) file.

Han et al. [9] argue that as a consequence of the rapid growth of WS applications and the abundance of service providers, the consumer is faced with the predicament of selecting the ‘right’ service provider. They conclude that there are no common evaluation criteria for WS because WS can be applied to many domains and the end user’s desire for the quality of WS is different. They propose a new evaluation model for WS, which supports customizing evaluation factors dynamically. It computes the weight of each evaluation factor using a machine learning algorithm based on samples from domain experts. This model can be used in WS management, WS selection and WS composition.

3 Web Services Coordination and Transactions

3.1 Web Services Infrastructures as Integration Middleware

In order to understand in depth transacting issues over WS, a closer look into WS environments as integration middleware systems is helpful. A technological resemblance to WS environments is application servers, where middleware functions such as messaging, transaction management, etc. assist interoperation of application components. In the same way, WS environments may provide similar middleware functions to service consumers. Yet, unlike application servers, the middleware itself consists of services that are potentially outsourced in the WS environment [1].

Current WS environments usually have at least one broker who is responsible for delivering services in compliance with the providers’ constraints, such as authentication, and quality in collecting input and returning output. To confront existing and future business model unpredictability, WS environments have to let third parties become brokers and mediators. Service providers require methods to integrate their services with other services and brokers [10], [1]. Thus, WS infrastructure has to proceed in
two major directions: a) long-running, conversational, and multiparty interactions have to be supported (while first-generation WS technology focuses on two-sided request-response dealings), and b) developers need to elevate mediation to a first-class concern. Unlike conventional mediators, intermediaries in WS environments will not be limited to format data representation differences. Instead, they will deal with divergences between long-running interactions.

For both previous considerations, is obvious that transacting over WS requires initially the establishment of a generic WS coordination and composition architecture. Developing a composite WS requires a specification that provides details about the component WS that it contains: for example, it identifies and triggers the component WS, it describes their execution order, it manages the data dependencies among them, and it triggers corrective strategies for exception handling. When composing WS, the business logic of the consumer is implemented by numerous services. A service consumer invoking a composite service can itself be exposed as a WS. This allows the definition of gradually more complex structures by increasingly aggregating services at advanced level of abstraction [11], [12].

3.2 Service Coordination: Orchestration, Choreography, and Composition

Supporting transactions depends significantly from the approach used to connecting WS together to form meaningful business processes. Terms such as "WS composition" and "WS flow" are used to describe the composition of the WS in a process flow. Additional terms are the following [13], [14]:

- Orchestration describes how WS can interact with each other at the message level, including the business logic and execution order of the interactions. These interactions may cross applications or/and organizations and actually formulate a transactional, long-lived multi-step process model. WS orchestration describes the way WS coordinate complex multi-party business processes.
- Choreography is typically associated with the public message exchanges that occur between multiple WS, rather than a specific business process that is executed by a single party. While for orchestration, the process is always controlled from the perspective of one of the business parties, choreography is more collaborative in nature: each party involved in the process describes simply the part they play in the interaction.

A service composition combines services following a certain composition pattern to achieve a business goal, solve a scientific problem, or provide new service functions in general [15]. Specifications for WS composition include the Web Services Flow Language (WSFL), the Business Process Execution Language (WS-BPEL) and the Web Services Choreography Description Language (WSCDL). Their focus is on tying multiple WS together to create multi-step applications, such as filling a purchase order or resolving an insurance claim. Such specifications provide the mechanism for extending the WS Description Language (WSDL) layer to identify a series or sequence of execution for multiple WS.

But this is not all that is needed. Coordination-oriented specifications define the complementary system layer that is necessary to ensure that the multiple WS achieve the desired results of the application and that the cooperation of multiple WS from whatever source (local or remote) produces predictable behaviour despite system failure and leaves the system in a known state [16].

Currently, there are two major coordination specifications: the WS-Coordination (WS-C) and the OASIS WS-Composite Application Framework (WS-CAF). In these specifications, specific standards aim on the introduction of transactions on WS implementations and attempt to bring the best features of transaction processing to WS [17], [18], [19]:

- Web services Transactions (WS-T) standard (part of WS-C): it supports both short-running atomic transactions which are executed within limited trust domains and business activities which are potentially long-lived transactions and spread in different domains of trust,
- Web Services Transaction Management (WS-TXM) standard (part of WS-CAF): it supports traditional ACID transactions, non-ACID long running actions, and business process transactions structured as a collection of previous type transactions with the responsibility to perform some application specific work.

WS-based (or business) transactions call for an extended transaction model that builds on existing WS standards and supports both long period transaction execution and relaxation of isolation levels. An additional requirement is the ability to negotiate at runtime transaction agreements [20]. Model
checking approaches [21] may be utilized to ensure these transaction-related guarantees.

4 Defining the Proposed Evaluation Framework

Service-oriented enterprises are quite concerned with possible interruption of services. Changing and negotiating with service providers leads to the emergence of service mediators. Transactions between parties might be complicated, because service intermediaries have several responsibilities trying to associate providers, consumers and even other intermediaries. The multiple roles that service intermediaries are asked to play provide a useful set of technical and user-oriented criteria for evaluating the efficiency, usability and effectiveness of WS usage for transactions.

Generally, the effectiveness of WS-based interactions and transactions is influenced by the major particular characteristics of the SOA. Among them, we may emphasize the following [22], [23]:
- Need for interoperability and for ease of integration – WS-based systems wish to support service dynamic substitution by assembling different services developed separately (and often offered by competing service suppliers) into one working application.
- Distributed character – different WS are frequently distributed across distant geographical positions, and this raises all related issues concerning performance.
- Decentralized maintenance process – WS form a distributed environment which is not placed within a single trust area, and this complicates the security issues and makes unrealistic to count on a single authority to coordinate all participating parties (for example, for supporting software modifications in a controlled manner).

The set of design concern subjects which we have selected is obviously not an exhaustive list of evaluation criteria. Still, they are sufficient to cover a broader variety of areas under considerations. We may distinguish two sub-sets of criteria: in technical category we may group all decisive factors related to system behaviour, leaving out user-oriented issues, which are actually grouped in the corresponding category.

4.1 Technical Criteria

Briefly speaking, interoperable and secure services of high-quality are necessary for supporting WS-based B2B transactions. We have to notice here, that especially interoperability across multiple transaction protocols is much more easily achieved when the overall system architecture follows an “open standards” philosophy of specifications, and avoids adopting inconsiderately proprietary and “closed” schemes.

Regarding QoS considerations, the most significant issues are those related with performance. An additional criterion refers to scalability, and to the level of the adoption of best practices about the recommended granularity of transactions, focused especially on achieving the proper level of reuse and composability.

Among the most significant aspects of security regarding transactions are to ensure the confidentiality (secrecy), integrity, and availability of the involved interactions. Moreover, an essential part of security considerations refers to accountability. Based on mechanisms capable to ensure identification-authentication of service consumers (by providing a single definition of users and a single user sign-on for heterogeneous service architectures), and to ensure proper authorizations regarding service invocations (by establishing flexible access control rules), responsibility is promoted and repudiation problems are unlike to take place.

**TABLE 1 Evaluation Criteria**

<table>
<thead>
<tr>
<th>A. Technical Criteria</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Interoperability</td>
</tr>
<tr>
<td>2. Quality</td>
</tr>
<tr>
<td>2.1 Performance</td>
</tr>
<tr>
<td>2.2 Scalability</td>
</tr>
<tr>
<td>3. Security</td>
</tr>
<tr>
<td>3.1 Confidentiality</td>
</tr>
<tr>
<td>3.2 Integrity</td>
</tr>
<tr>
<td>3.3 Availability</td>
</tr>
<tr>
<td>3.4 Accountability</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>B. User-oriented Criteria</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Participant-related</td>
</tr>
<tr>
<td>1.1 Cultivate user trust</td>
</tr>
<tr>
<td>1.2 Cultivate user loyalty</td>
</tr>
<tr>
<td>2. Administrator-related</td>
</tr>
<tr>
<td>2.1 Monitor and Report</td>
</tr>
<tr>
<td>2.2 Isolate and Diagnose problems</td>
</tr>
</tbody>
</table>

4.2 User-oriented Criteria

Users involved in service-based transactions
(such as software developers who either build applications that consume specific services, or build actually service implementations), may appreciate (or may seek for) special characteristics capable to cultivate user trust and loyalty. User trust can be achieved mainly by pre-qualifying service providers, based on past experiences. Additionally, user satisfaction is increased when their interactions are designed taken under consideration the discovery of the best (registered) services based on parameters provided by them.

On the other hand, system administrators (people who are responsible for long-term maintenance) look for abilities to monitor and report on the security and performance of the application in use. Of special interest is to monitor availability issues (e.g., the status of the business commitments – Service Level Agreements), and the capability to isolate and diagnose problems in the SOA environment.

We have to underline that contextual information may play a decisive role for supporting more or less the whole set of the previously mentioned criteria. So, a central issue related to the evaluation criteria, as we will analyze in the next paragraph, is the capability for managing properly the context information.

5 Using the Proposed Criteria

5.1 Need for Context-aware Transactions

In several cases, the involvement of a WS in a composite WS is prearranged and does not depend on its execution state. However, in more flexible composition schemes, a WS, before agreeing to engagement, must consider its internal execution state and question whether it would be rewarded for its involvement, or it must know how to manage its performance if it gets disrupted, how to secure its peer interactions, how to adapt its behaviour in case of failure, when or when not to participate in a composite WS and when to recommend peers that might better satisfy an engagement request [24].

To reduce their current limitations, WS must measure both their existing capabilities and in progress commitments. In addition, they must evaluate their nearby environment before binding to any composition. Actually, WS must be context-aware. Context refers to what surrounds the point of focus, provides additional sources of information “where, who and what”, and increases understanding by giving meaning to it. By developing context-aware WS it would be possible, to consider the facets of the environment in which the WS are to be executed. These facets are multiple and can be related to user profiles, computing resources, time of day, and physical locations [25], [26]. We will examine how to contextualize WS composition, and how to deploy context-aware WS.

The composition of WS using present standards is accomplished at the level of message interactions. This is not satisfactory as composition must also be accomplished at the level of message semantics. To realize semantic composition, context information must be provided. This context information is any metadata that unambiguously describes the significance of the data to be exchanged between WS. When WS’ inputs and outputs are clearly described via metadata, WS can be changed properly and without human intervention from one context to another during WS execution.

Making WS context-aware requires that many issues must be addressed, such as how is context structured or what is the overload on a WS of taking context into account. Researchers [25] have started to organize the context of a WS along three interconnected perspectives. The participation perspective is about supervising the numerous composition settings in which a WS joins in at the same time. This assures that the WS is identified (or even adapted) suitably and so it is ready for execution for every composition scenario.

The execution perspective is about the computing resources on which a WS works. It also deals with watching the capabilities of these computing resources so the WS’s conditions are continuously satisfied. Finally, the preference perspective is about making certain that user preferences are taken under consideration and actually incorporated into the arrangement of a composite WS.

Participation, execution, and preference perspectives are all linked. Deployment joins participation and execution perspectives, and deals with the execution of a WS as it accepts participating in a composition. On the other side, tracking connects execution and preference perspectives, and underlines the importance of monitoring the execution of a WS, so user preferences are suitably accommodated. In the end, customization ties preference and participation perspectives and emphasizes the option of fine-tuning a WS so it can fulfill a variety of user preferences.
5.2 Need for Policies

All these aspects can be mapped onto policies, which would in reality define a WS’s satisfactory behaviour. We may consider policies as external, dynamically adaptable sets of regulations that cause the system to adjust to administrative decisions and changes in the execution environment. Thus, policies denote activities that must be completed in line with context changes and detected incidents, and are indeed particularly appropriate for WS composition.

Adopting policies brings in the prospect of varying WS’ actions without changing a composition specification. Via policies, WS developers may regulate several characteristics such as WS’ conflict-resolution mechanisms. In fact, WS developers can use policies to manage WS at both low and high levels [24]. Examples of low-level policies issues are how to exchange intelligible information among WS, how to replace a WS without disturbing the execution flow, how to postpone a WS execution because of risk that information could be captured, etc. High-level policies deal with composition related issues, such as what user preferences may be integrated into WS, what is mandatory to suit to these preferences, etc.

5.3 Reflections on Exploiting the Proposed Evaluation Criteria

Security measures are not something that can be added in a certain system’s architecture, without having thought of them and design them at the very early stages. Since among the proposed criteria is to encounter successfully a number of security concerns (criteria A.3, B.1.1 and B.2.1), the basic model - specification for describing all major issues regarding the utilization, exploitation and management of the evaluation criteria should operate above all as (or should facilitate the operation of) a rigid security-oriented mechanism, capable to:

- Ensure that the whole WS-based transaction (along with its component WS, the participants of the transaction), make use of state-of-the-art encryption technologies in order to protect its (and their) confidentiality, integrity, and availability in a way that every participating part (whether it is a WS, or other architectural component) is accountable for its actions.

- Grant or deny permissions to WS for participating in WS-based compositions, such as transactions.

Current WS specification on security, trust and agreements (WS-Security, WS-Trust) promote the adoption of policies as the basis on which to interoperate, and policy languages (e.g. WS-Policy, WS-Agreement, WSPL) are already widely available [5], [6].

On the other hand, based on what we have already mention in the previous paragraph, policies are a promising way to support interoperability (criteria A.1) and to solve demanding mediation issues (criteria A.2.2, and B.1.2) in WS composition and transaction scenarios. Service mediation may be defined as the act of repurposing existing WS so they can cooperate in ways unanticipated during their development or deployment [1]. Thus, scalability is undoubtedly benefited by service mediation capabilities.

Mediation is usually achieved by intercepting, storing, transforming, and routing incoming and outgoing messages. Since service interface is the only contact point between service providers and service consumers, designing a suitable service interface is an issue of critical importance. So, a prominent form of service mediation is service interface adaptation, where the goal is to keep interfaces as generic as possible while adapting them to specificities that arise in each usage scenario. This adaptation may be defined to some extent on (explicit or implicit) user preferences supporting in this way customization and personalized interactions. Certainly, this increases user satisfaction and user loyalty.

Policies indeed seem as a proper means to exploit and manage WS-based transactions’ evaluation criteria, because they are capable to support the effectiveness of the security and interoperability technologies. This is done mainly, by handling efficiently the contextual information. The integration of context into WS composition/transaction ensures that the requirements of and constraints on these WS (either security- or interoperability-oriented) are taken into account. Context may support WS in their decision-making process when it comes to whether accepting or rejecting participation in a transaction. Predicting what will happen to a WS would also be feasible in case the past contexts (that is, what happened to a WS) are stored. WS can take advantage of the information that past contexts cater, so they adapt their behaviour for better actions and interactions with peers and users.
Moreover, context permits tracing the execution of WS during exception handling. It would be possible to know what happened and what is happening with a WS at any time. This is of major importance, as it facilitates everyday administrative maintenance tasks (criterion B.2.2).

Last but not least, we have to discuss about the role of the major importance performance criterion (A.2.1). Performance is among the key factors of ensuring that the user is going to continue participating in particular WS-based interactions and/or transactions. Certainly, the quality of WS-based interactions serves as a benchmark to differentiate service providers [9], [27]. So, a further evaluation point during transactions is the effect the combination of interoperability and security would have on performance. This would also link to user-oriented criteria and in particular that of user loyalty as performance would certainly affect the user’s intentions of returning to a particular WS-based transaction. An open issue therefore, is how to establish a means of dynamically evaluating performance in relation to all the other criteria mentioned above (e.g. the performance and effectiveness trade-offs presented in [28]) and how to map such type of calculations onto properly structured policies.

6 CONCLUSIONS

This work proposes a two-dimensional table of evaluation criteria as a first step at establishing an evaluation framework for WS. In this approach interoperability and security are recognised as essential to a user subscribing to a WS.

Thus the proposed framework will be evaluating performance in relation to all those criteria included in table 1 above. The difficulty in establishing such an evaluation framework lies in that all previous attempts of evaluation utilise independent protocols that address security or performance but not both. A new set of protocols or a major revision of current ones is therefore required. It is expected that the evaluation results may help all involved parties (software developers, managers, system administrators, etc.) to become aware of the differentiations between approaches and adopt the various considerations judiciously in their implementations.

This work is currently in development. The authors are also looking at means of capturing data that would be utilised with the new framework and improve the effectiveness of evaluation.

REFERENCES


Christos K. Georgiadis is currently a Lecturer in the Department of Applied Informatics at the University of Macedonia, Greece. He received a Ph.D. in information systems & web databases security and a B.Sc. in mathematics from Aristotle University of Thessaloniki (AUTH), Greece. His research interests include the areas of e-/m-commerce technologies, e-/m-commerce security, and web services technologies. He is a member of Greek Computer Society and professional member of ACM – SIGecom.

Elias Pimenidis is a Senior Lecturer at the University of East London, UK. His research focuses on e-business and e-government projects. Other research interests include the evaluation of web services, knowledge management systems and the use of computer games for educational purposes. He is a Chartered IT Professional Member of the British Computer Society (BCS).