Advanced Web Services

Jonathan Dale (ed.)
Fujitsu
jonathan.dale@fla.fujitsu.com

David Levine
IBM Corporation
dwl@watson.ibm.com

Francis G McCabe
Fujitsu
fgm@fla.fujitsu.com

Geoff Arnold
Sun Microsystems
geoff.arnold@sun.com

Margaret Lyell
The MITRE Corporation
mlyell@mitre.org

Harumi Kuno
Hewlett Packard Company
harumi_kuno@hp.com

2nd September, 2002
Version 0.9a

1 Introduction

Business is about the evolution of relationships in which organisations collaborate to complete tasks. Today this collaboration is largely defined by human interactions, often mediated by computer-based technologies. The promise of Web Services, as defined by organizations such as the W3C and Oasis, is that many of these interactions can be realized using a common set of Internet technologies. In this white paper we envisage a world in which Web Services can model short-term and long-term business relationships and in which business relationships include automated collaboration between software systems acting as representatives of business organisations.

The REST architecture defines a framework in which today's Web infrastructure can support several forms of business interaction. These are achieved by the use of technologies such as XML, SOAP messages and service descriptions such as WSDL and UDDI. However, the operations thus supported represent only a fraction of the interactions which make up business relationships. We suggest that to model the full range of business activities requires additional work in a number of areas. Specifically, we propose that the Web Services Architecture (WSA) should be enhanced to include:

- peer-to-peer and multi-party interaction;
- the dynamic composition of services;
- message patterns which go beyond the simple request-response paradigm;
- long-term identity, roles and relationships;
- contracts and service-level agreements; and,
- service accountability, auditability and verifiability.

The aim of this white paper is to ensure that the WSA requirements document and subsequent W3C specifications should support not only simple Web Services but also this richer and more complex view of how Web Services are likely to evolve. It is not our intent to disrupt or nullify any of the existing work in WSA; instead we want to ensure that the W3C specifications do not preclude the emergence of what we are calling "business-friendly" Web Services. In order to do this, it is necessary for us to provide an account of the kinds of business processes that we expect to see Web Services providing.

2 Motivation

The current state of Web Services is based on a set of technologies that support various types of business interactions. For example, the XML standards support data exchange and interoperability through the use of a verifiable, platform-neutral representation. Public listings of business services are available through the UDDI Registry, with the services themselves described in WSDL. However, the current state of Web Service technologies and their extrapolations can represent only a fraction of the full range of business activities.

The parties involved in a business relationship are better modelled as peers than as clients and servers; even if some are suppliers and others are customers. For example, consider the case in which a consumer interacts with a Travel service. The Travel service, acting on behalf of the consumer, interacts with Hotel services, Airline services, Entertainment services and Car Rental services. The consumer has placed a maximum dollar limit on the fees that they is prepared to incur for lodging and plane fare and has also expressed specific requirements regarding the quality of the hotel and travel times. By linking the two items (the airline ticket and the hotel reservations) into a single requirement, the consumer causes a multi-party interaction to take place among the Travel service, Airline service and Hotel service business peers in an effort to meet the consumer's requirements. Furthermore, each of these services must then interact with the business services of specific airlines and hotels that serve the city which the consumer desires to visit. Each airline and hotel represents another business peer.

Further complicating the interactions is the fact that long-standing business relationships often exist among specific pairs of hotels and airlines. A specific airline and a hotel that are part of such a long-standing relationship might engage in a side interaction and prepare to bid as a team for the consumer's business. The complex set of interactions that may take place among these peers and sets of peers is not easily modelled by request-response message patterns; it more closely resembles a series of conversations in which the various parties take on different roles. Also, the participants (the collection of business peers, as represented by their respective Web Services) may vary from one request to the next and may be composed dynamically depending on the consumer requirements. The next consumer might only be interested in the availability of hotel rooms, thus obviating the need to involve Airline services or individual airlines.

Suppose at some time before the trip is to occur, the consumer learns of a new theatre production. Realizing that they will be in the city on some of the performance dates, they re-contact the Travel service to request tickets. The Travel service now contacts the specific Hotel Web Service directly and conveys the request. The hotel sends the request to its in-house Concierge service, which chooses whether to satisfy the request itself or use an Entertainment service. However, the in-house Concierge service also participates as a business entity in the Entertainment service. When the in-house Concierge service satisfies the request for tickets, its identity and its role in the transaction must be known. In this case, the Concierge service's role is as a department of the hotel business entity. The Travel service needs information on both the identity and the role of the entity that satisfied the theatre ticket request. With increased flexibility comes the increased need for audit and verification services. These services must be supported by the WSA and attendant technologies.

3 Realisation

3.1 Business on the Web

Central to managing business-oriented relationships is the creation of shared understandings: Shared understanding of the task being undertaken, shared understanding of the terms being exchanged and ultimately the creation of contracts which capture shared agreements. This process is aided by having an architecture (see Figure 1) which accounts for the all the roles, obligations and permissions associated with a business oriented relationship.

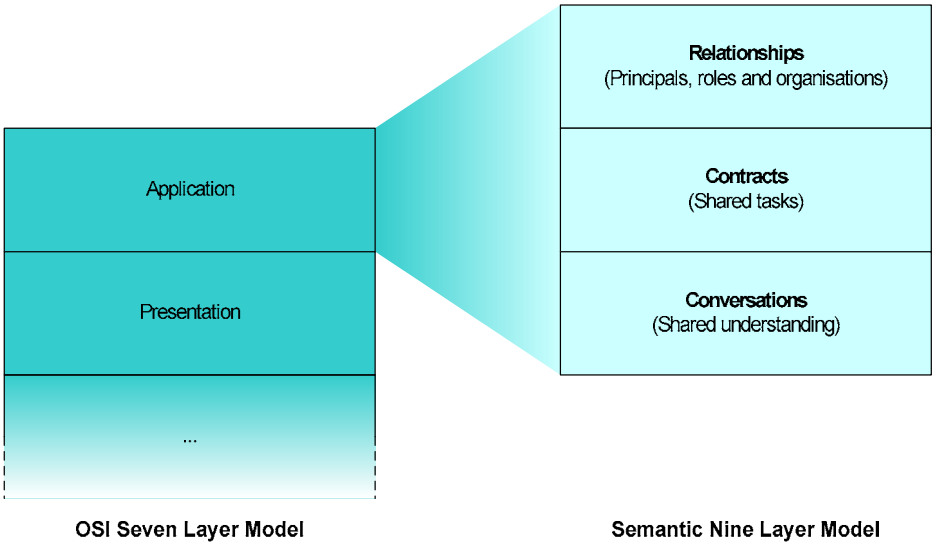

Figure 1: Extensions to the OSI Seven Layer Model

3.1.1 Shared Understanding: Core Plumbing Interoperability is Crucial

Business interactions are founded on shared understanding. Automated systems need clarity to define an explicit description of conversational elements, such as the types of messages, the meaning of messages and the expected ordering of messages, as well as explicit descriptions of the meaning of conversational patterns.

Technology support requirements include standard message content languages, orchestration and choreography and ontology descriptions.

3.1.2 Shared Tasks: Achieving the Task as Advertised

Business activities revolve around tasks shared between different partners. Automated systems need clarity to define an explicit description of tasks in terms of the expected effect of actions and the dependencies between tasks. Additionally, there needs to be an explicit description of responsibilities for principals involved in the task which defines their rights and obligations and the conditions which can apply.

Technology support requirements include standard languages to express process flow, service-level agreements and policies.

3.1.3 Principals, Roles and Organisations: Ensuring the Task is Achieved by the Right Person

All business activity takes place in the context of relationships between principals. The constitution of organisations consists of definitions of roles, rights and obligations of member principals of the organisation. Automated systems need clarity to define an explicit description of the constitution of organisations to give an accounting of the roles, rights and obligations of member principals, as well as an explicit means of recording and managing trust relationships and authority relationships.

Technology support requirements include standards for describing constitutions, legal entities, privacy, trust and reputation.

3.2 Technology Roadmap

3.2.1 Peer-to-Peer

The Peer-to-Peer (P2P) communication model is characterized by dynamic symmetry: each party exposes a set of comparable functionality and any party can initiate a communication session at any time. The P2P paradigm can be seen as distributing the traditional client/server responsibilities evenly between the partners in the interaction. However, the essence of P2P from a business perspective is that it reflects the reality that each partner in a business relationship is 'equal' and that a simplistic approach to customer/supplier relationships cannot be accurately modelled in a strict client-server architecture.

This evolution of the client/server model is reflected by the Web Service paradigm. The first Web Services are distinctly traditional in their treatment of the client/server implementation; each Web Service acts as an independent server to programmatic or human clients. However, the loosely-coupled nature of the Web Service model is extremely compatible with P2P computing and we foresee that adopting P2P and multi-party interaction capabilities will enable Web Services to engage in a richer and more dynamic set of business interactions. In particular, the fact that peers can independently invoke each other means that a Web Service peer could both volunteer information as well as perform inquiries on another Web Service. This capability enables asynchronous modes of interaction, where a supplier can notify a customer of an event or a prospective client can submit a request for bids to multiple service providers.

3.2.1.1 Issues and Requirements

A number of issues must be addressed in order for Web Services to realise the symmetry of P2P peers. In particular, both the Web Service and its peer client may have some sort of permanent presence, that it must be possible to model extended conversations between peer Web Services and that it must be possible for peers to volunteer information as well as invoke methods asynchronously.

In addition, the WSA must not preclude multi-party interactions, such as auctions, escrow services, proxy services, broker services, etc. In order to implement proxy and broker services, it must be possible to quote, verbatim and modified, messages within top-level messages to an arbitrary depth, as well as to send anonymous messages that elide the intended sender and recipient. Furthermore, it must be possible to express multiple receivers and to express 'wait' points in service orchestration. Finally, there should be models and mechanisms to assign identity independently from the principal's attributes and it should be possible for one peer Web Service to discover another using purely publicly observable semantics.

3.2.2 Identity

The notion of identity is a key part of the enhanced Web Services world which this document describes. Identity is a well-understood and important feature of closed system architectures (for example, in mechanisms such as user login, access control lists for files, and, more recently, role-based access control). However, it has not figured prominently in distributed computing and, instead, it has been treated as part of the application content and has been handled in many different ways by various applications. Recent initiatives such as the Liberty Alliance and Microsoft's Hailstorm have sought to provide a unified model for authentication and identity, but these still amount to a single sign-on for short-lived Web transactions.

In a world of complex, long-term and multi-party Web Service interactions, identity takes on a new importance. It becomes an element by which multiple service interactions can be correlated as part of a larger business relationship. If we imagine a cluster of Web Service interactions between an individual, their health-care provider, their insurance company and the local pharmacist, the key elements that allow these interactions to be recognized as part of a single business relationship are the identities of the principals. These interactions may evolve over time, with new

services being incorporated into the overall relationship; identity is an important element of such orchestration.

3.2.2.1 Issues and Requirements

A complete model of identity must take into account the notion of role: the idea that a principal may play a particular function or part within some interaction. The relationship between identities and roles is an important aspect of Web Service choreography, particularly in complex services involving multiple parties such as workflows. Consider the case where a particular transaction requires the approval of either the Chief Financial Officer or two members of the board of directors. If Alice is the CFO and also a member of the board, it is not sufficient to simply say "Alice approves the transaction." She can approve in either of her roles, but if she does so in her role as Director, then the transaction will still require another approver. Of course, the precise rules surrounding her right to approval are important here – she may be able to assume both roles simultaneously or not, depending on the constitution of the organization she represents.

Another example is where an individual might interact with a company in the role of customer or shareholder and as such, the access to information and range of possible actions depend on the role chosen.

3.2.3 Content Language

A content language is a language used to express the content of messages that are exchanged between principals involved in transactions. Given such a definition, it is clear that XML is a content language since it too can be used to express messages in a conversation. However, there are benefits for adopting a stronger content language; a content language with a stronger opinion about the form and semantics of the messages exchanged. The primary benefit of a semantically rich content language is that it facilitates a higher-level of interoperability between systems: by agreeing on how meaning is conveyed, it makes it simpler for applications to share meaningful content.

3.2.3.1 Issues and Requirements

Any strong content language must be strong enough to convey the kinds of meaningful content that we envisage; in addition, it must be engineered to be efficiently processed and fit well with existing technology, applications and best software engineering practice. In the context of Web Services, that implies a strong relationship to SOAP, UDDI, WSDL and a strong connection to standard implementation technology such as Java and .NET.

Some of the kinds of objects that are required to be communicated between applications across ownership boundaries are:

- assertive statements: *the price of that widget is $X*;
- requests and commands: *open that file*;
- negotiable contracts: *if you pay me $X, we will send you widget Y*;
- policy statements: *to use this service you must have a valid certificate signed by Z Inc*; and,
- declarations: *I hereby agree to the terms and conditions of the contract*.

Since the meaning of terms is often at issue when two separately owned entities are collaborating, the ontologies referenced need to be clear: this can, in turn, lead to a requirement that definitions themselves be communicated, for example, this widget is a car, as opposed to being a pet animal (say).

In addition to purely technical requirements, it is also clear that any content language designed for use by Web Services must be consistent with evolving industry choices for technologies in Web Services. It is also extremely helpful if any impedance mismatch between content languages and implementation technologies is minimized: this suggests that the use of strong typing in the content language. Traditionally, logic-based languages have either been untyped, such as XML,

RDF and KIF, or weakly typed, such as many-sorted logics and Conceptual Graphs, where weak typing here means that there is some difficulty in using traditional type inference technology to enforce type safety. However, to fit in with programming languages such as Java, it is very helpful to use strong typing in the content language to provide a framework for development tools to automate the integration with programming technology. However, the representational requirements for content languages are significantly different from those of regular programming languages; in addition to values of various forms, it is also necessary to be able to represent contracts, business rules, conversations and so on. These requirements impose strong constraints on any type system used.

We are of the opinion that neither RDF nor the prospective OWL (which is based on DAML+OIL) meet the requirements outlined above for expressing content: from a technical perspective they are not strong enough to represent the richer forms of content and from an engineering perspective they have too high an impedance gap between their expressive power and the capabilities of common programming languages[1].

At its core, a content language is required to express the messages between application systems. Therefore, it must contain elements that can describe the content of such messages: values and actions. In addition, it is necessary to be able to represent more propositional information: predicates and rules. It may also be necessary to support other features such as type definitions themselves.

At a higher level, it is also necessary to represent elements such as conversational descriptions, contracts, role definitions, policies, fragments of ontologies and even organisational constitutions. Such elements should, in principle, be layered on top of a more basic logical framework. We envisage a content language being composed of the following elements: a type system, a term or value system, a rule system, an action description system and a recursive structuring system that allows hierarchic combinations of these elements to be composed into coherent wholes.

3.2.4 Interaction Patterns

An interaction pattern is a *graph* that represents possible sequences of messages and that guides two or more principals as they attempt to carry out a shared task. Interaction patterns structure interactions between principals by providing conversational templates with desirable (often proven) properties. For example, an interaction pattern may assure its participants that it will either converge to an agreement or terminate within ten message exchanges. Interaction patterns are closely related to the choreography and orchestration of Web services, but focus on the public declaration of interaction paths between peers, rather than fully dictating the interaction.

As interaction patterns are intended for public consumption by principals, they are structured as XML documents available for interpretation. Interaction patterns are marked up to indicate the roles of participants, the messages which progress participants through the interaction and points at which the pattern concludes, either with success or failure.

3.2.4.1 Issues and Requirements

In order for the WSA to support interaction patterns, roles and identity must be properly supported and P2P interactions accounted for.

3.2.5 Security

Web Services will naturally take advantage of emerging security architectures. However future features of Web Services will emphasize particular aspects of security and may well introduce new security requirements.

[1] Of course, neither was designed to be a strong content language.

At the heart of the advanced Web Services concept are the ideas of modelling long-lasting multi-party business relationships in terms of dynamically composed Web Services and of Web Service interactions representing verifiable performance against a contract. From this perspective, the security of Web Services must support mechanisms such as the verifiable capture of message exchange patterns, contracts and execution traces to support non-repudiable performance verification. It must address issues of long-term identity and roles, including the kind of identity transformations associated with business life-cycle events. Finally, it must define ways to support legally and commercially appropriate forms of anonymity and delegation.

3.2.6 Contracts

A contract is an agreement between two or more organisations which binds the actions of the principals under an agreed set of terms and conditions. Contracts are formed between organisations by authorised roles whose capabilities are enabled and restricted by the rights and obligations delegated to them by their organisation. For example, a researcher, by definition of their role, might not be allowed to form a contract with a supplier to purchase inventory for a company. However, that task would be allowed by the role of a purchaser. The actions in the contract define the task details and the conditions describe how to account for contract fulfilment, violation, repudiation, etc.

3.2.6.1 Issues and Requirements

Of the essence for contracts relating to automatic systems are *machine-readable* contracts – contracts that can be parsed and comprehended by software systems. This requires not only a standard `syntax' for contracts but also implies constraints about the logical form of contracts. In addition, since contracts often imply commitments there is normally a relationship between such machine-processable contracts and human law.

One of the key aspects of contracts is the ability to identify the principals that are agreeing to the contract to ensure its integrity. Additionally, the nature of the tasks to be achieved and the conditions surrounding them must be defined and established. Contracts themselves, however, can often be expressed in relatively simple terms – of the rights and obligations of the parties involved – typically in terms of *rules* that apply when certain detectable situations apply.

3.2.7 Conflict Resolution

Inevitably, in any system where there are different elements belonging to different legal entities, there will be differences and conflicts arising from business interactions. While resolving conflicts directly is probably beyond the scope of the WSA, the nine layer model in Figure 1 does give some guidance in dealing with conflict.

Essentially, according to Figure 1, we can expect business-level conflicts at three levels:

- Miscommunication can result from a message being missed or unexpectedly repeated; from a formatting error or from a semantic misunderstanding. This latter form of failure may be of the form of a term being misused (such as a 'Jaguar' referring to a cat instead of a car) or a lack of mutual understanding (such as 'what is that XYZWidget referred to?').
- Failures in execution can be of any number of forms; not delivering a payment advice, not paying the due amount and so on. Essentially, in the context of a contract or service-level agreement, such a failure is modelled by one or more clauses in the contract being *violated*.
- A failure at the relationship level often takes the form of a valid transaction being enacted by a system without the correct authority to perform the action.

The merit of this analysis is that it helps to disentangle the various kinds of conflict that might occur and to establish a framework for appropriate resolution. For example, it makes no sense to go to court when the failure has been a miscommunication; on the other hand, no reordering of messages is going to correct an authority violation.

4 Conclusion

Our vision is that Web Services can be an effective vehicle for realizing dynamic, evolving business relationships: everything from accessing a simple thermometer reading service to a full inter-corporate alliance. The special factors that are new today are the ubiquity of the Internet and the willingness to adopt global standards to facilitate integration; together these promise to reduce friction and enhance innovation.

Our thesis is that by adopting the three key layers associated with relationships, shared tasks and shared understanding, the WSA can offer the world a top-to-bottom account of the interactions between principals using Web Services as a vehicle. It also offers a clear account and framework for the various security and trust technologies that will be needed to permit people to deploy and use Web Services with confidence.

Finally, by separating the different levels of concern into distinct layers, it leads to a sound and evolvable architecture for deploying Web Services that can support all aspects of the business relationship.

5 Acknowledgements

The authors would like express their thanks to all of the people who have contributed to this work, in particular, Stefan Brantschen, Bernard Burg, Monique Calisti, Patricia Charlton, Dominic Greenwood, Luc Jarry, Norbert Mikula, Marc Miller, Ned Smith, Donald Steiner, Mike Stockman, Kate Stout, Katia Sycara, Ivana Trickovic, Laszlo Zsolt Varga, Steven Willmott and Sinisa Zimek.

The opinions expressed in this paper are not necessarily the opinions of organisations that each author represents.

6 Appendix: Enhancements to the Web Services Architecture Requirements Document

6.1 Business Friendly Goal

The WSA must provide a framework which reflects the evolving needs of businesses as they conduct business using Web Services.

Critical success factors for measuring the success of this goal are:

AC023 Peer-to-peer interoperability;
AC024 Multi-party interactions;
AC025 Service re-use;
AC026 Semantic descriptions; and,
AC027 Relationships.

6.1.1 Peer-to-Peer Interoperability

The WSA must support interoperability between peers as well as client-server interactions.

AR023.1 The WSA must permit a rich range of message interaction patterns, including patterns such as request-response, publish-subscribe, forwarding, proxy-ing and event notification.
AR023.2 It must be possible for peers to have persistent identities that are distinguished from any other attribute, such as their location or type.
AR023.3 It must be possible for peers to interact without the required presence of any third-party intermediary.
AR023.4 It must be possible for peers to discover each other.

6.1.2 Multi-Party Interactions

Web Services must be able to support N-party interactions, such as auctions, escrow services, proxy services and broker services.

AR024.1 It must be possible to quote, verbatim and modified, messages within top-level messages, to an arbitrary depth.
AR024.2 It must be possible for Web Services to support interactions where one of more parties of the interaction are anonymous.
AR024.3 It must be possible to express multiple receivers and to express 'wait' points in service orchestration.

6.1.3 Service Re-Use

The WSA must provide a framework for effective re-use of existing services.

AR025.1 It must be possible to compose services dynamically, on the fly, as well as statically.
AR025.2 The service composition model must permit the expression of and the evolution of composed relationships.
AR025.3 It must be possible to express the sequencing of services and the nesting of services, as well as the flow of information between services.
AR025.4 It must be possible for third-parties to verify the performance of services (where performance includes results as well as timeliness).

6.1.4 Semantic Descriptions

It must be possible to characterize a Web Service so that its semantics are clear to an automatic system.

AR026.1 The WSA should be aligned, where appropriate and possible with the Semantic Web. This may require some modification of current technology choices. (This is a version of D-AC009.)
AR026.2 It must be possible to publish references to an ontology in a Web Service description.
AR026.3 It must be possible to characterize a Web Service using purely publicly observable semantics. The semantic description of a Web Service should rely on public explicit agreements and these descriptions should be based on the purely observable characteristics of services and principals.

6.1.5 Relationships

It must be possible to model the identities, roles and relationships of principals involved in a Web Service.

AC027.1 There must be proper separation of roles and identity in transactions:
- AR027.1.1 Choreography and orchestration must be role-oriented.
- AR027.1.2 It should be possible to identify and authenticate a principal acting in a given role.

AC027.2 There must be an account for the many different time-scales over which relationships can occur:
- AR027.2.1 It must be possible for relationships to persist across changes in the environment.
- AR027.2.2 Temporal characteristics of relationships should be explicitly documented in Web Service descriptions.