Putting Web Services in Context

David Martin¹

Artificial Intelligence Center
SRI International
Menlo Park, CA, USA

Abstract

When considering the full range of Web service-related activities, it becomes clear that dealing with context is a major challenge, requiring greater expressiveness, reasoning capabilities, and architectural components than are provided by the current widely accepted building blocks of the Web services stack. This paper presents an informal overview of concepts, requirements and challenges for handling contextual knowledge in connection with Web services, and briefly discusses several interesting projects in this area of research.

Keywords: Context, Web services, Semantic Web, Semantic Web services

1 Introduction

At first glance, it appears that Web services ought to be described in a “context-free” manner. After all, a Web service is normally conceived as a neatly encapsulated module of functionality that can be easily reused, so long as the inputs, outputs, and messaging protocol are conformant with its description. However, when we begin to look beyond toy examples, we see that the picture is not nearly so simple. To support automated discovery and selection of world-changing services, for example, service descriptions must be unambiguous about what situations will guarantee successful service uses, and what new situations will result from those uses. For some categories of services, service behavior may vary with time, location, user history, pre-existing

¹ Email: martin@ai.sri.com
contractual commitments, and so forth; descriptions of such distinctions can quickly become complex.

Moreover, many aspects of service use and management may require knowledge that isn’t normally captured in service descriptions. Matchmakers may want to consider provider track records, reputation, and recommendations from third parties. Service compositions, to be effective, may need to consider a variety of resource constraints and interrelationships between service providers. Recovery from failure may involve a complex set of factors including user preferences, account status, policies that vary for different kinds of transactions, availability of appropriate substitutes (items or actions), etc.

Indeed, when considering the full range of service-related activities, it becomes clear that dealing with context is a major challenge, requiring greater expressiveness and reasoning capabilities than are supported by the current widely accepted building blocks of the Web services stack.

This paper is concerned with context representation requirements on Web services, and with their relationship to current work on Web services technology. The goal is to set the stage (not to say “create a context”) for discussion and further work on context for Web services, and to this end I present an informal overview of concepts, issues, requirements and challenges in this area. The following section discusses the nature of context and its definition. In Section 3 I consider what kinds of contextual knowledge Web service-based systems may need to handle. Section 4 gives a sampling of several projects that use context in interesting ways, or propose new ways of handling it. Section 5 discusses technology challenges that are raised by context handling requirements. Section 6 discusses the state of the art in Web services and Semantic Web services technologies, in relation to these challenges, and Section 7 gives a brief summary.

2 Context and Web Services

What is “context”? As with many broad, abstract concepts there is no single crisp definition that is universally accepted. The word has many different nuances of meaning in connection with logic, linguistics, cognitive science, and information sciences. Even within information sciences there is no widely accepted definition. Some definitions focus on the interactions between software systems and their environment. For example, Brezillon, in [1], defines context as “the information that characterizes the interaction between humans, applications, and the surrounding environment”. Other definitions focus on the characterization of a situation. An example of this focus, given by Pomerol and Brezillon [2] is the following definition: “the collection of relevant conditions
and surrounding influences that make a situation unique and comprehensible”.

For purposes of this paper, we will adopt a simple, informal definition that is more in tune with the latter category, but which is also associated with getting something done: Context is “background knowledge useful in accomplishing some task”. What kind of task? In general, it could be anything: solving a problem, reaching a conclusion, making a decision, answering a question, taking an action. For Web services in particular, things like selecting a service and creating a service composition are typical tasks. What do we mean by background knowledge? It is knowledge that is not the focus of attention; that is, not central to the formulation of the task. Background knowledge includes what we normally call the “givens” of a problem or task, knowledge that is known or considered as a matter of course. (“Given that I belong to United’s frequent flyer club, my travel agent selected that airline for the trip to Paris.”) By-and-large such knowledge is more general and of longer duration than the explicit task that is in focus, but this is not always the case.

User preferences provide a good illustration of these ideas. We generally expect a system to make use of user preferences in a transparent manner, by and large. We expect to make a request such as “Reserve a flight from San Francisco to Boston on March 19” (a task request made explicit), and expect a system to automatically consider our preferences that are known to it (for example, time-of-day and seating preferences for flights). Hence, the preferences are left implicit, and may be regarded as background knowledge relative to that request.

Much work on context for services is concerned with the location of a mobile service user and her communication devices. Here, a typical task might be to find a nearby store at which a particular kind of product (say, pharmaceuticals) can be purchased. This task would typically be made explicit as, say, “Find a store that sells pharmaceuticals that I can get to quickly”. Here, the focus of the request is on the store and the time constraints, and the relevant background knowledge includes my current location.

This notion of foreground vs. background information is somewhat subtle, and admittedly does not admit of a “crisp” definition. For example, an alternative formulation of the previous task might be “Find a store selling pharmaceuticals that’s within one mile of my current location”, in which case the concept of “location” is mentioned explicitly. In this case, however, it is still appropriate to regard my location as background knowledge, because the system is expected to know (or be able to determine) my precise location as a matter of course; it is considered to be “a given”. The same can not be said of the primary focus of this request, finding a store. In this case, the system
is not expected to know the identity of such a store; indeed, if it were expected to already know that, the request wouldn’t make sense. Furthermore, the identity of the store will be made explicit to me, the requester, as the request is satisfied, which is not necessarily true of my current location. For these reasons the store’s identity may reasonably be regarded as foreground knowledge, and my location as background knowledge.

3 Categories of Contextual Knowledge

Bearing in mind our informal definition of context, we may identify some rough categories of contextual knowledge, by thinking about Web service tasks and the kinds of background knowledge that may be helpful in accomplishing those tasks.

In the following list, the categories are ordered, very roughly, in terms of frequency of change. That is, in the categories near the beginning of the list, the contextual body knowledge changes slowly, whereas in the categories near the end, it changes more rapidly. For example, user location (in the User Situation category below) changes fairly often, at least several times a day, so that a given proposition about a user’s location will not remain true for very long. User preferences, on the other hand, do not normally change so quickly, so they belong to the User Characteristics category, which comes earlier in the list. Similarly, Organizational Arrangements, such as policies and structure of organizations, are presumed to change even more slowly than User Situation (although of course exceptions do occur). It is worth emphasizing, again, that these is a rough classification scheme, and also not intended to be an exhaustive classification.

- **Organizational Arrangements** include such things as organizational structure; relationships between organizations; relationships between people and organizations (e.g., membership); relationships between people within organizations; ongoing policies; contractual commitments; and ongoing partnerships.

  For example, if a Web service is provided by an organization having an ongoing contractual relationship with my organization, that is relevant to the task of service selection for corporate procurement purposes. Also, the manager / subordinate relationship between my boss and me is relevant background knowledge with respect to the task of prioritizing my incoming email messages.

- **The Service Characteristics category** includes all available Web services (and their descriptions), service registries, brokers, etc. — roughly speaking, the static elements that make up the world of Web services. This can
also include various kinds of information about service provider reputation, recommendations from third parties, etc.

For example, if a Web service guarantees delivery of a product within 3 days, that may be relevant to the task of service selection for purchasing a camcorder before my son’s birthday. If a weather service only provides information for locations within Europe, that is relevant to the task of getting a weather report for tomorrow’s travel destination on my itinerary.

It is debatable whether service descriptions, such as WSDL or OWL-S descriptions, should ever be regarded as part of context, since they are central to so many Web service tasks. For present purposes, they are included here, if only to emphasize the point that what is considered to be context is relative to the formulation of a task.

- **User Characteristics** includes relatively stable characteristics of (human and/or software agent) service users, such as preferences and constraints, level of expertise, and possibly the user’s connection with projects and documents in some settings.

  The travel examples above, involving a user’s membership status and seating preferences, illustrate this category.

- **User Situation** includes such things as location, time of day at that location, physical characteristics and surroundings of a (human and/or agent) service user. It also includes resources and devices available to the user, mobility vs. stationarity, network connectivity, and resource requirements. The user’s plans and schedule may also be relevant, as well as the status of her ongoing activities, and such things as the status of her accounts with different organizations.

  The examples above having to do with finding a drugstore illustrate this category.

- **Transaction History** includes records of past transactions involving Web services. Clearly this is related to the Service Characteristics category, but the emphasis here is on (relatively complete, up-to-the-minute) records of individual uses of services.

  For example, if records show that a particular service provider has recently been responding more slowly to service requests, that may be considered in subsequent provider choices.

  If records show that a composition of services A and B gets handled more quickly when they are both provided by the same provider, that contextual knowledge may influence me to continue in the practice of choosing the same provider for both.

- **Resource State** includes information about resource usage in connection
with Web services.

For example, if the machines or network connections of a particular service provider are currently overloaded, that information may influence an agent to choose a different provider that offers the same or a similar service.

Whereas Transaction State (just below) reflects the status of a particular transaction, Resource State reflects the status of a set of resources associated with many transactions.

- **Transaction State** includes aspects of a transaction that a user is involved in currently, such as the provider identities and network locations of specific services within a larger composed service. It can also include such things as the step reached in a composed service, and the time elapsed since the transaction was initiated, or since the last message was sent.

As mentioned above, this categorization reflects in a rough manner the temporal aspect of how rapidly the knowledge is changing. Although space does not permit it here, it might also be interesting to categorize contextual knowledge along the dimensions of location and provenance. That is, one might consider the following questions: Where in cyberspace are the different kinds of contextual knowledge created, stored, and needed? By whom in organizational space are they created and by whom are they owned?

### 3.1 Context Categories in Relation to Service Management Tasks

There is a distinction between tasks that are accomplished by a Web service — such as making a flight reservation — and tasks that are related to the use or provision of Web services themselves, such as service development, selection, contracting, composition, monitoring, etc. For lack of any common terminology, let us call this latter category of tasks the *service management* tasks. Context is relevant to both types of tasks.

It may be useful to consider briefly what kinds of context are most relevant with respect to the different service management tasks. These are not to be taken as rules or constraints, but are merely rough conclusions based on my perspectives on these tasks and context categories. With respect to the service management tasks of development and publishing, Service Characteristics and Organizational Arrangements are the most important kinds of context. Discovery and selection, notably, can depend on *all* of the categories, but with the possible exception of Transaction State. Contracting and negotiation may rely on Transaction History, Organizational Arrangements, Service Characteristics, User Characteristics, and sometimes User Situation. Composition (considered apart from its reliance on discovery) relies primarily on Service Characteristics and Resource State. Monitoring and recovery are
likely to draw on all categories except Transaction History.

4 Case Studies

Here we briefly discuss several research systems that deal with services with reference to context, and consider the kinds of problems they address, the kinds of knowledge that are treated as contextual, the categories to which this knowledge belongs, and the technologies employed.

Task Computing [10], developed at Fujitsu Laboratories of America, is a framework that transparently provides access to (relatively low-level) services that are needed to accomplish a (relatively high-level) user task. That is, the framework fills the gap between user’s tasks and the available means to carry them out. There is a particular focus on automating access to computational resources in a given physical environment, which may be an unfamiliar environment to the user. For example, in a meeting room, a user may request to “download the presentation from my desktop and show it on the projector here”, and the system will map that request onto the available local services (such as file management, file translation if needed, and projection). Similarly, in a car, the request might be to “get a map with directions from here to the local office and show it on my PDA”, which might employ services such as online mapping, file transfer, and screen display.

The contextual knowledge includes primarily user location and availability of user devices (User Situation); user relationship to documents (User Characteristics), and elements of Resource State and Transaction State. Relevant technologies employed include Semantic Web Services descriptions, matchmakers, composers, and other tools and components, based on OWL-S (in particular, atomic processes that may be grounded either to WSDL or to UPnP).

MyCampus [7], developed at Carnegie Mellon University, is a system designed to discover, compose, and execute services to fulfill a variety of tasks in a complex setting – a University campus. The system is designed to support access from mobile devices, and to assist the user in accomplishing a variety of day-to-day activities, such as scheduling meetings, finding destinations, sharing documents, organizing events, filtering and routing messages, and so forth. For example, if the system is asked to find a restaurant for a quick lunch, it might consider the walking time based on the user’s current location, the classification of a restaurant as “fast-food” or not, and also the weather. (If it’s raining it would try to locate a restaurant that can be reached without walking outside.)

As can be seen from the above description, the contextual knowledge is
varied and primarily falls in the categories of User Characteristics and User Situation. Attention is given to user preferences and security and privacy issues. Technical approaches include rule-based access to information sources, and the modeling of sources of contextual information themselves as Web services that can be discovered, composed, etc. OWL-S grounded atomic processes are used here also, along with sensors, triggers, and an eWallet component that encapsulates the security and privacy mechanisms for a particular user.

**OWL-SF** [6] is a system that provides proactive call-forwarding for a set of users. The system makes decisions about whether and where to contact a user (or redirect the call) based on its awareness of such things as the user’s location, calendar entries, time of day, and people in proximity to the user. For example, if a call comes for a user while he is in a high-priority meeting (as indicated by his calendar), the call would be redirected to voice mail so as not to interrupt the meeting. On the other hand, if the user is in the cafeteria, and sitting alone, the call would be routed to his cell phone.

This application is concerned primarily with User Situation. The technical approaches include the use of OWL-based subsumption reasoning, which takes place in deduction servers.

**Semantic Discovery Service**, developed at Stanford’s Knowledge Systems Lab, is a system that demonstrates how the functionality of BPEL4WS [4] can be augmented using Semantic Web Services (OWL-S) descriptions. The additional service characterization captured in OWL-S allows for the selection of more appropriate services, and also for the automatic selection and use of data mediation services. In a typical scenario, a user who is seeking an online mortgage has relocated from England to the United States. His software agent has the ability to execute a BPEL workflow that carries out the steps involved in acquiring a mortgage, with dynamic service binding at runtime. In the scenario, the service semantics added by OWL-S are used to select a credit reporting service in UK, based on the system’s knowledge that his credit history is located there. It is also able to select a mediation service that translates from a UK credit history report to one formatted according to US conventions.

The relevant contextual information in this scenario is the (actual or intended) location of the user’s residence, at two different times. This information is categorized under User Characteristics. The system may also consider user preferences, which are in the same category. The central technologies are BPEL4WS and OWL-S, including dynamic binding and OWL-S-based matchmaking.

**ConWeS**, developed by Sattanathan, Narendra, and Maamar, is a framework for managing context in connection with the execution of service comp-
positions. It addresses the problems of provisioning, coordination, resource management within composition, and ontology mediation (both for application data and for context management). It deals with contextual information regarding the execution status of service instances, the number of instances allowed and currently deployed, time constraints (deadlines) on instance completion, and interdependencies between composed services.

This contextual information falls primarily into the category of Transaction State, but also includes Resource State. This work proposes both a new ontology, OWL-C, and a new architecture for storing and managing information about Transaction State.

5 Challenges in Representing and Reasoning about Context for Services

Dealing with context for Web services clearly raises a number of challenges, which have not been widely recognized or addressed by the Web services community. The overarching questions here are how to represent contextual knowledge and how to make it available where it’s needed, in a way that enables reasoning, decision making, and taking action. In this section, I summarize some of the major issues that have arisen in work in this area.

Representation. We may begin by considering the language requirements for representing contextual knowledge. Because definitions of context are so broad, it is difficult to arrive at any firm conclusions about the expressiveness or language features that might be needed. (After all, in principle “knowledge” includes everything that can be represented, including the most complex and abstract definitions and axiomatizations.) Nevertheless, most systems dealing with context to date have created only very modest, even minimal representational requirements, which can be met by basic relational databases, basic uses of description logic, or simple uses of rules languages.

This can be perhaps be understood by revisiting the categories of contextual knowledge presented in Section 3. Four of these categories — User Situation, Transaction History, Resource State, and Transaction State — are concerned with simple, easily captured facts. (That is, given an appropriate ontology or schema, representing the facts about user situation, transaction history, and transaction state should be straightforward.) Furthermore, User Situation is one of the two categories most frequently featured in work on context for services. The other most widely used category — User Characteristics — raises the possibility of using rules, at least when it comes to preferences and constraints. For example, in preferences, one can easily imagine rules such as “When flying on an international flight, I prefer an aisle seat; other-
wise window”. But again, most such examples are readily handled using basic rules language features.

The Organizational Arrangements and Service Characteristics categories each offer special challenges for context representation, at least in principle. In Organizational Arrangements, one may want to include contractual agreements between organizations, which are potentially complex. However, I am not aware of any work on service context to date that has attempted to do so. Similarly, if service descriptions, in all their potential generality, are treated as elements of contextual knowledge, here again there can be a great deal of complexity to deal with.

**Security and privacy.** Dealing with context raises significant issues related to security and privacy. Obviously, information about user characteristics (such as preferences), user situation (such as location and status of financial accounts), and organizations (such as contractual arrangements) needs to be carefully guarded. It is equally clear that mobile access to such information from a variety of devices creates situations in which special measures are necessary. In addition, whenever a service management component wants to consider the context of multiple organizations or users, arrangements are needed to allow for the sharing of that information. Transaction history cannot be shared, at least not in its full generality, except by special arrangements. For all of these cases, architectural structures and mechanisms need to be designed and standardized to allow for access to the contextual information with appropriate levels of access control.

**Distributedness, heterogeneity, and mediation.** Working with context, especially in mobile and multi-organizational settings, can impose significant requirements for gathering and mediating information from widely distributed, heterogeneous knowledge sources, devices, and services. In [8], Sattanathan, Narendra, and Maamar propose the OWL-C (Ontology Web Language-based Context) ontology, whose primary purpose is to facilitate consolidation and mediation of contextual information related to the execution of composed services. (This information falls into the Transaction State category.)

**Scalability.** Two of our context categories — Service Characteristics and Transaction History — raise issues of scalability, simply by virtue of the potentially huge size of knowledge bases capturing those kinds of information. Other related issues include the speed with which contextual information can be retrieved with respect to a given task, and the time required to reason about it, especially on resource-limited devices.

**Sensors and triggers.** Many kinds of scenarios involve significant use of sensors and triggers in gathering and handling contextual information, espe-
cially in the categories of User Situation and Transaction State. For example, a system’s knowledge of the user’s location and available devices depends upon sensors that report that information. Similarly, a system’s knowledge of transaction state may depend upon cyberspace “sensors” that detect critical events in the course of transaction execution. Triggers are mechanisms that react to certain conditions, as they become true, by initiating some appropriate action. Such conditions are frequently regarded as part of contextual knowledge. For example, the task request “When I am in an XYZ project meeting in room 55, invoke the audio recording service” triggers off a condition about location, and “When I am working on Powerpoint slides, turn on the rapid archiving service” triggers off a condition about a type of activity. Architectural provisions and tool features will likely be needed to make it easy for service (provider and client) developers to make use of both these kinds of mechanisms.

**Other architectural issues.** A number of other issues have to do mainly with architecture; that is, with the need for mechanisms for making contextual information available, at the right time and place, to the agents that need it.

In some settings, matchmaking and service composition components are likely to outgrow the essentially “context-free” approaches that are currently in the forefront of research. By “context-free” is meant approaches in which basic service requests (given to matchmakers) or service goals (given to composers) are presumed to contain all the information that these components need to do their job. As these components start to make better use of contextual information, they will need to have effective, standard means of finding, accessing, and reasoning about it.

For example, context repositories that reflect the activities of multiple service providers, and may be accessed by multiple service providers, may be needed in some applications, as illustrated by the C-contexts of [8]. Similarly, transaction history registries may need to be established as a means of sharing information about the track records of services and service providers. In some settings, the need to share contextual knowledge could be met by information propagation mechanisms, such as blackboard, publish / subscribe approaches, or the triple store approach proposed in [3].

Other general questions related to architecture include: How is context structured, how are changes detected and assessed for context update purposes, and what is the load on a Web service from taking context into account?

We see here that the enablement of contextual reasoning for Web services, in its full generality, generates a very broad set of challenges. As Web service-based systems become more ambitious in their scope and capabilities, it will become increasingly important for Web service languages and architectures to be designed in light of these challenges.
6 Discussion

We may well ask where we stand with respect to technical infrastructure and approaches for handling contextual information. Although the basic Web services technology stack provides the interoperability and messaging mechanisms by which information can be transferred between actors on the Web, it has essentially nothing to say about how the different categories of contextual information will be represented, where they will be stored, how and by whom they will be maintained, when they will be shared, what kinds of reasoning will be used with them, how to ensure their scalability and security, and so forth.

One may also look to the Semantic Web Services (SWS) research field. SWS work has mainly focused on developing languages and ontologies that are suitable for use in characterizing services. The most visible work in this area includes OWL for Services (OWL-S) [5], the Semantic Web Services Framework (SWSF) [9], and the Web Services Modeling Ontology [11]. Because the SWS vision is broad and its goals are ambitious, SWS researchers have developed very expressive, general-purpose languages for representing a broad spectrum of service characteristics in a single framework. Expressiveness in these frameworks, then, is already adequate for purposes of context handling. Moreover, the work on domain-independent ontologies of service elements is promising, in that it builds up from primitive elements to define a single comprehensive set of concepts that can be used to capture many different kinds of contextual knowledge. In this regard, SWSF, although somewhat immature, is a good exemplar. But to cover even the low-hanging fruit of capabilities for context-aware systems, ontology development is still needed in many areas, as illustrated by the ontological proposals in [8]. Another area of work in SWS, and in Semantic Web work in general, that can be applied to the handling of contextual knowledge is mediation, which is most particularly a focus in the WSMO effort, and is also discussed in [8].

Architectural requirements by and large have not been addressed from the perspective of context handling. Indeed, there are still many unanswered questions about how foreground knowledge, such as preconditions and effects of services, will be expressed and handled in real-world environments — such as how, when and where precondition and effects expressions will be evaluated. SWSF and similar efforts assume the existence of local knowledge bases for service clients and providers, but have not yet developed a complete picture about knowledge exchange and consistency between service and provider (or peer) knowledge bases. There is also not a clear delineation of the types and scope of knowledge that can be expected to be present at any given provider, client, or peer site. These problems are yet more complex in the case of
distributed composed services involving multiple providers, or peers. The handling of background knowledge (context) has received much less attention from SWS researchers. However, the problems just mentioned are closely related to the issues around context, and progress on these problems can be expected to be applicable to context issues.

In addition to the SWS field, it would also be valuable to consider Semantic Web technology more generally, as well as Grid computing and Semantic Grid, but those areas are beyond the scope of this paper. It should at least be noted, however, that the work on the Web Services Resource Framework (WSRF) [12], also at OASIS, includes some techniques that can potentially contribute to the management of our Transaction History, Resource State, and Transaction State categories of contextual information.

In principle, the challenges associated with contextual knowledge are potentially as broad as the field of knowledge representation and reasoning itself. However, there are various assumptions and limitations that can lead to special-purpose approaches, as illustrated by the work cited in Section 4. Thus, a great deal of valuable functionality can be achieved without solving all these challenges in their full generality.

Moreover, some high payoff areas can easily be identified, such as user location and physical context, user preferences, constraints, and memberships, track records of service providers, and status and history of execution instances. In these areas a great deal of utility can be derived from relatively simple representational means. Semantic Web technology already provides basic infrastructure by which to capture and exploit knowledge in these areas already. The remaining effort needed is in building and establishing consensus around shared ontologies, and creating the architectural components by which the knowledge can be managed. By and large, these are the primary focal points of current work in this area.

7 Summary

Although “contextual knowledge” is not formally defined, nevertheless it is a useful notion that brings to light a number of representational and architectural requirements related to Web services. This paper has discussed the different kinds of knowledge that can be considered contextual, with respect to Web services tasks, and the many challenges around meeting the needs of developers who are attempting to build systems that are context-aware. In addition, it has presented several projects that have made contributions to our understanding of the role of context, and has considered the directions that are needed in Web services research to enable more effective handling of the
wide range of contextual knowledge that may be incorporated in future Web services-based systems.

8 Acknowledgments

This paper grew out of my presentation as keynote speaker at the CWS-05 International Workshop on Context for Web Services. I would like to thank the workshop organizers — Djamal Benslimane, Chirine Ghedira and Zakaria Maamar, for inviting me to participate in this event, and for their excellent work in organizing the workshop.

References


