On Integrating Conversations into Web Services Composition

Zakaria Maamar
Zayed University, Dubai, U.A.E
zakaria.maamar@zu.ac.ae

Soraya Kouadri Mostéfaoui
Oxford Brookes University, Oxford, UK
E-mail: kouadris@acm.org

Keywords: Web service, conversation, composition, context.

Received: November 28, 2004

We present an approach for integrating conversations into the process of composing Web services. A Web service is an accessible application that other applications and humans can discover and trigger to satisfy various needs such as weather forecasts. While much of the work on Web services to date has focussed on low-level standards, it is becoming urgent to allow Web services engage in conversations, make decisions, and adjust their behavior according to the context of the situations in which they participate.

Povzetek: Predstavljena je integracija koverzacije v spletne storitve.

1 Introduction

In this paper, we highlight the role of *conversations* in the field of *Web services* and assess the value-added of integrating conversations into the composition of Web services. Composition primarily addresses the situation of a user's request that cannot be satisfied by any available Web service, whereas a *composite service* obtained by combining available component services (i.e., Web services or composite services) might be used [3]. While much of the work on Web services to date has focussed on low-level standards for publishing, discovering, and triggering Web services [2], the use of conversations promotes Web services to a higher level by giving them the opportunity to act as active components. A conversation is a consistent exchange of messages between participants who are involved in joint operations and thus have common interests.

In order to engage in conversations, Web services (also called services in the rest of the paper) have to be leveraged from passive components, which only respond to triggers, to active components, which make decisions and adjust their behavior according to the *context* in which they evolve. Context is the information that characterizes the interactions between humans, applications, and the surrounding environment [4]. Huhns backs the importance of leveraging Web services by using software agent architectures [9]. Indeed Web services, unlike software agents, are not designed to use and reconcile ontologies. Moreover, software agents are inherently communicative, whereas Web services are passive until invoked.

While some authors agree on the importance of leveraging Web services to the level of active components, others have already identified some similarities between Web services and software agents [6]. Services (i) advertise their capabilities after specification using for example WSDL, (ii) search for other services using for example UDDI, and (iii) invoke services without prior notice using for example SOAP. This kind of behavior bears many likenesses to software agents. For instance, a service accepts requests (sense) and returns responses (action) [6]. In addition, once a service is invoked, it performs tasks with or without further inputs (autonomy). However, it is the authors' belief that the aforementioned behavior of a service is mainly hard-coded and consequently, limits the service in its action selection. Enhancing Web services with conversation capabilities will first, enable an emergent behavior during composition and second, permit to services to be more flexible in managing the situations in which they participate.

Web services composition is a very active area of research and development. However, *very little* has been achieved to date regarding the seamless integration of conversations into composition approaches of Web services. In particular, several obstacles still hinder this integration such as: (i) Web services are dealt with as passive components rather than active components that can be embedded with context-awareness mechanisms, (ii) existing approaches for service composition (e.g., BPEL4WS) typically facilitate orchestration only, while neglecting contextual information on services, and (iii) lack of support techniques for modeling and specifying conversations between Web services. In this paper, the focus is *on the conversations happening among a group of Web services, which are called to constitute composite services*.

2 Conversations and Web services

Several authors note that current standards of Web services are used in systems featured by simple interactions [1, 7]. Simple because the interactions adopt a *request-response pattern* (e.g., announce/confirm). However there are multiple situations that need more than two turns of interaction (e.g., propose/counter-propose/accept $\oplus$ reject $\oplus$ counter-propose/...). The participants in these situations have to engage in conversations before they reach an agreement. Another initiative in the field of Web services is the Web Services Conversation Language (WSCL). This language describes the structure of documents that a Web service expects receiving and producing, as well as the order in which the exchange of documents is expected to take place. While conversations in [1] occur between end-users and providers of Web services, and WSCL focusses on specifying the operations that Web services support, our focus is on the conversations happening among Web services. These services might originate from different sources and have to engage in conversations in order to agree on what to exchange, how to exchange, when to exchange, and what to expect from an exchange during composition.

2.1 *Composition stages*

To assess the benefits of conversations to Web services composition, we decompose the composition into three stages: pre-composition, composition, and post-composition. In the following, only the pre-composition stage is discussed. Because similar services exist on the Internet, it is important to search for and identify the services that satisfy specific user-defined selection criteria (e.g., execution cost, reliability, availability [14]). Conversations in the pre-composition stage concern the following aspects:

1. Identification aspect: use search mechanisms (e.g., UDDI registries) to identify Web services. We assume that Web services are already specified and advertised.

2. Invitation aspect: invite Web services to participate in a composition. An invitation is either accepted or refused. The rationale of inviting services instead of directly triggering them is given in [12]. In addition, conversations occurring during service invitation are described towards the end of this paper.

3. Compatibility aspect: check if the Web services can exchange meaningful information because of data heterogeneity issues that may raise. Details on the semantic compatibility of Web services are found in [13].

2.2 *Conversation's components*

Pre-composition, composition, and post-composition stages are each concerned with some of the conversational aspects that take place during Web services composition. Because of the variety of these aspects, we decided on (i) associating each aspect with a *conversation session* and (ii) implementing a session with a *course of conversational actions*.

To handle the multiple conversation sessions, we use *Conversation Schemas* (CSs) as a technique for describing these sessions and their respective course of conversational actions. We define a conversation schema as a specification of the exchange of messages that is expected to happen between participants. This exchange depends on several factors including application domain, active context, and current chronology. For example, a conversation schema describes both the side-effects of a conversation that is misunderstood and the corrective actions that permit fixing these side-effects.

First of all, the initiator of a conversation downloads a conversation schema from the library of conversation schemas (Fig. 2) according to the following elements: current composition stage, progress in this stage with regard to the current aspect, and intention behind establishing the conversation. These elements are needed since conversations are context sensitive [5, 15]. Next, the initiator instantiates the conversation object (i.e., gives a value) and checks the activation condition before it transfers a message to a receiver. Upon reception of the conversation object that the message conveys, the receiver takes one of the following actions:

1. Accepts the conversation object without any change and starts acting based on the conversation object's content.

2. Changes the conversation object and submits the modified conversation object to the initiator for either further change, approval, or rejection.

3. Rejects the conversation object and either submits a new conversation object to the initiator or ignores the initiator (this one has a time-out constraint).

Modeling conversations is a complex process as several requirements need to be satisfied. Some of these requirements are identified in [10]: (i) conversation models should be task-oriented, (ii) conversation models should be associated with a semantics, (iii) conversation models must provide communication abstractions, and (iv) conversation models should be reusable and extendable. In this paper, we specify conversation schemas with state charts [8]. In addition to satisfying some of the aforementioned requirements such as (i) and (iv), encoding the flow of conversations using state charts has several benefits. First, state charts have a formal semantics, which is essential for reasoning on the content of conversations. Next, state charts are becoming a standard process-modeling language as they are being integrated into UML. This process modeling helps in managing admissible turns and decision makings during conversations. Finally, state charts offer most of the control-flow constructs that can be found in real conversations such as branching and looping. Fig. 1 depicts the

mapping of the concepts of a conversation schema onto the concepts of a state chart.

- Name of states is labelled with **S/R** (sender/receiver).
- Name of transitions is labelled with activation condition and conversation object.
- Actions of states implement the information that conversation objects convey.
- A complete state chart illustrates the conversation schema of a conversation session.

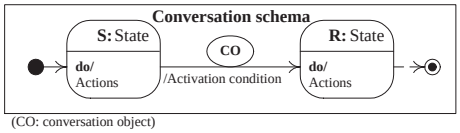

Figure 1: Conversation schema as a state chart

3 Context-aware conversations for Web services composition

To assess the way a composition of Web services progresses so relevant conversation sessions are entered and relevant conversation schemas are downloaded from the library of conversation schemas (Fig. 2), the use of awareness mechanisms is required. These mechanisms ensure that the status of each Web service is known and instantaneously reflected in a structure, which we denote by $\mathcal{W}$-context (context of $\mathcal{W}$eb service). using the information that $\mathcal{W}$-context caters, it is possible to know if a Web service is part of a composition, under execution, or invited to participate in a composition.

In one of our previous works [12], we strengthened the advantages of the combination (software agent, Web service, context). For example, agents track Web services in order to update their respective $\mathcal{W}$-contexts. The tracking is about the composition in which a service takes part, the current state that the service takes in the composition, and the type of conversations that the service has initiated during the composition. Fig. 2 presents the context-aware conversation approach for Web services composition that we developed. The features of this approach are below.

1. Three types of software agents are set: conversation-manager-agent, composite-service-agent, and service-agent.

   A conversation-manager-agent runs on top of the library of conversation schemas. It updates the library if a new specification of a conversation schema is developed (e.g, by a designer). Plus, the conversation-manager-agent responds to the requests of downloading conversation schemas that composite-service-agents and service-agents submit.

   A composite-service-agent triggers and monitors the deployment of the specification of a composite service (to keep Fig. 2 clear, the specification store is not represented[1]). This monitoring is reflected in a context, which we refer to as $\mathcal{C}$-context (context of $\mathcal{C}$omposite-service). In addition, the composite-service-agent interacts with service-agents when it comes to inviting services for composition or informing services of the changes in the specification of the composite services.

   A service-agent is responsible for getting a Web service ready for composition, monitoring its execution through its state chart, and updating its $\mathcal{W}$-context.

2. Besides the state charts of the conversation schemas (Fig. 1), additional state charts are associated with Web services (Fig. 2). The states that a Web service takes are immediately reflected in its $\mathcal{W}$-context. Interesting to note that the transitions in the state charts of services are *context-based* and *conversation-oriented* (Fig. 2-2). However, these conversations are less complex than the conversations that involve Web services (Fig. 2-1). Therefore, each state of a Web service is associated with a $\mathcal{T}$-context (context of Web-service s$\mathcal{T}$ate).

3. A library of conversation schemas that stores the specifications of conversation schemas (Fig. 1). The library is a resource from which composite-service-agents and service-agents download conversation schemas after interactions with the conversation-manager-agent.

In Fig. 2, the conversations occur in two separate locations. In the first location, the conversations concern the component Web services that participate in a composite service (Fig. 2-1). These conversations are specified using the conversation schemas of Fig. 1. In the second location, the conversations concern the states of the Web services (Fig. 2-2). It should be noted that the state chart of a conversation schema supports the interactions between the state charts of services. The distinction between states of services and states of conversations gives a much better flexibility in managing the aspects that each type of state chart is concerned with. State charts of services focus on the changes that apply to services such as availability, commitment, and execution, whereas state charts of conversations focus on the changes that apply to conversations such as formulation, reception, and response.

Because of both types of state (those associated with services and those associated with conversations), we annotate each conversation state with a context, which we refer to as $\mathcal{S}$-context (context of conversation $\mathcal{S}$tate). The rationale of $\mathcal{S}$-contexts of the states of conversations is similar to the rationale of $\mathcal{T}$-contexts of the states of services. Moreover,

[1]The specification of a composite service is based on state chart diagrams, where a state is labelled with a service chart diagram and a transition is labelled with events, conditions, and variable assignment operations [11].

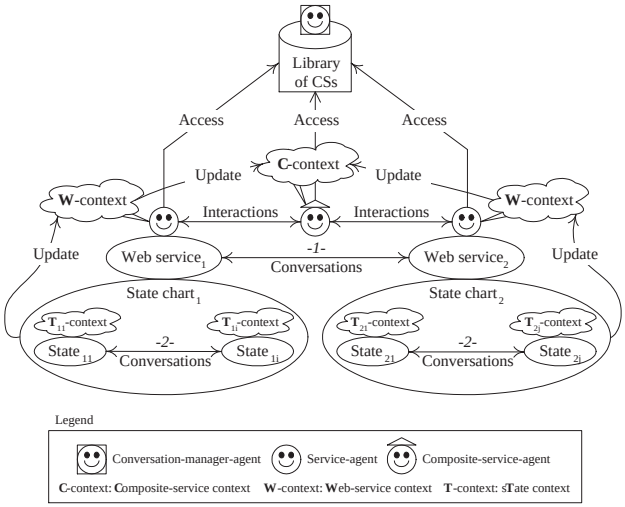

Figure 2: Context-aware conversation approach for Web services composition

to track the progress of a conversation a context, which we denote by $\mathcal{V}$-context (context of a con$\mathcal{V}$ersation), is used. This is similar to the $\mathcal{W}$-context of a Web service.

4 Development example of a conversation schema

We decomposed the composition of Web services into three stages: pre-composition, composition, and post-composition. Each stage consists of different aspects, which characterize the conversations that occur. In this section, we illustrate how a conversation schema is developed. This development consists of devising two state charts, one for the conversations and one for the services that participate in these conversations. Before we explain the development process, the following comments are made on both types of state charts:

- State labels are annotated with S/R (sender/receiver). If there is no annotation, the state identifies the communication network.
- Transitions implement the links between states. Two types of transition exist. The transitions that connect states of the same chart are represented with regular lines. The transitions that connect states of separate charts are represented with dashed lines.

To keep the paper self-contained, only the conversation schema featuring the invitation aspect is illustrated (Fig. 3). While a composite service monitors the component Web services that are currently under execution, the composite service submits an invitation to the next component Web service to join the composition.

States of services. There are four states, which are seen from two perspectives.

- Sender perspective: two states are associated with the sender service namely monitoring and assessment. Here, the sender corresponds to the composite service. One of the actions in the monitoring state consists of downloading the conversation schema for Web services invitation. We recall that this schema is stored in the library of conversation schemas (Fig. 2).
- Receiver perspective: two states are associated with the receiver service namely assessment and deployment. Here, the receiver corresponds to the next component Web service. One of the actions in the assessment state consists of checking the current commitments that the Web service has towards other composite services. Based on these commitments, the component service either accepts or rejects the invitation to participate in a composite service.

States of conversations. There are six states, which are seen from three perspectives. Ellipses glued to transitions correspond to conversation objects.

- Sender perspective: two states are associated with the sender service namely preparation and reception.
- Receiver perspective: two states are associated with the receiver service namely preparation and reception.
- Network perspective: two states are associated with the communication network namely transmission from the sender to the receiver, and transmission from the receiver to the sender.

5 Implementation

As a first step of validating the proposed approach of Fig. 2, we have developed a *conversation and Web services composition-manager*, using Borland JBuilder Enterprise

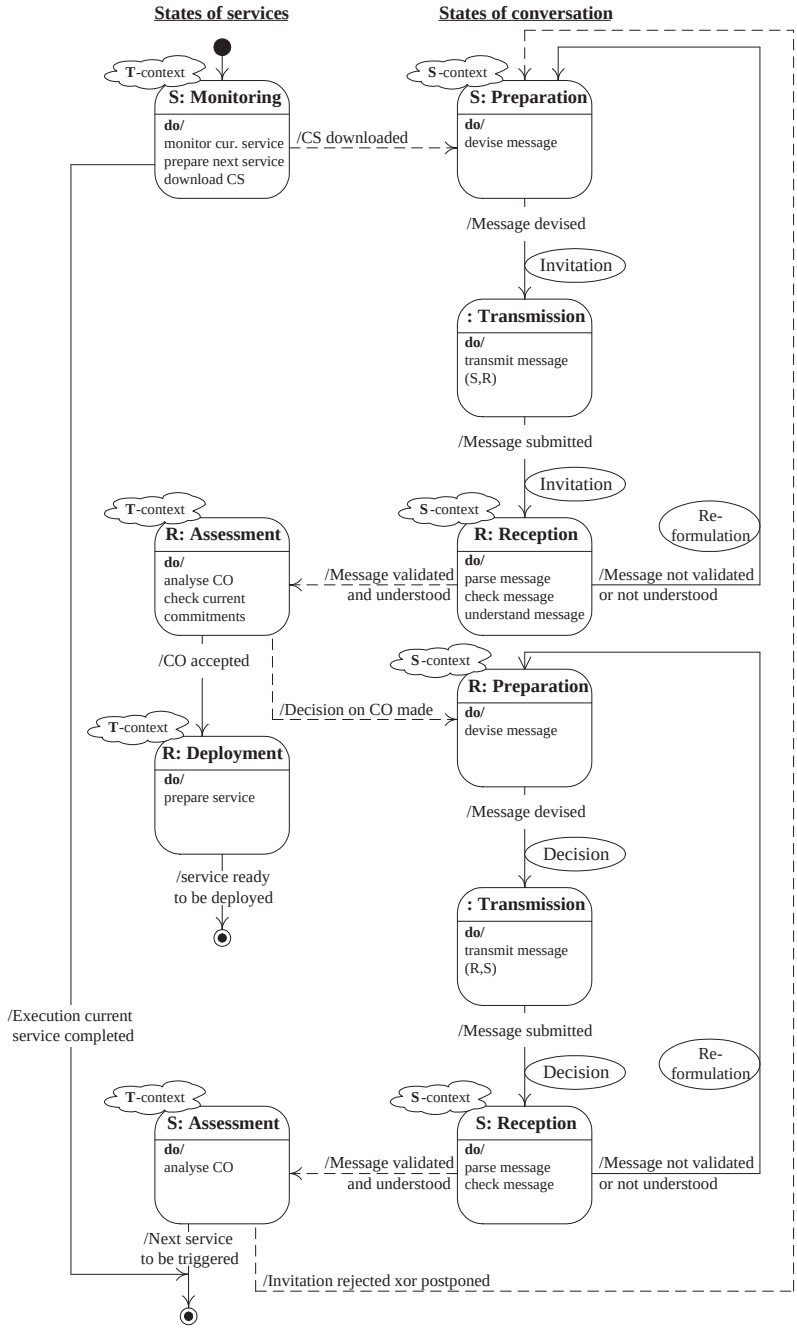

Figure 3: State chart of a conversation schema - invitation aspect

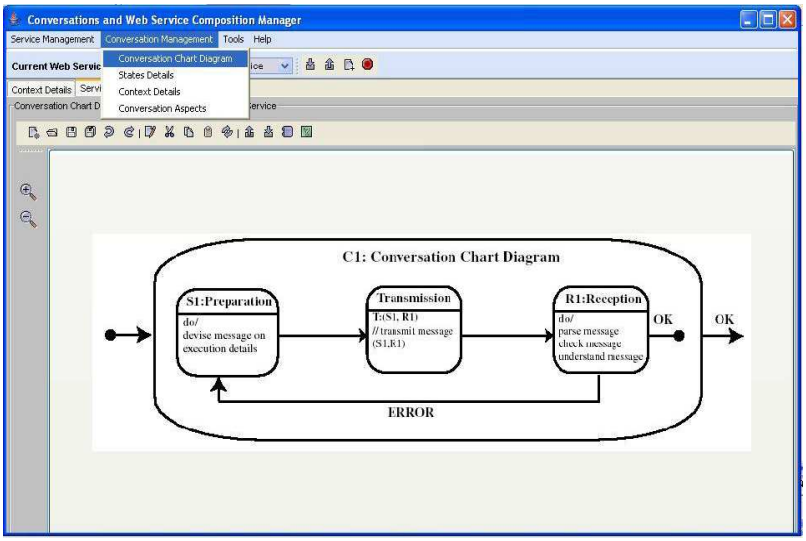

Figure 4: Graphical editor for conversation schemas and service chart diagrams

Edition version 9[2]. The prototype integrates a set of tools, which allow for instance Web services' providers and users to create, compose, and execute services based on the different contexts. WSDL is used for Web services specification, and UDDI is used for Web services announcement and discovery. Details of contexts and conversation sessions are structured as XML files. A dedicated XML editor was developed in order to create, validate, test, and monitor the different XML files. The validation of these files is based on two XML schemas (conversations.xsd and context.xds). In addition, the conversation manager offers an editor for describing the state charts of conversation sessions and composite services (Fig. 4). The graphical editor provides means for directly manipulating conversations and service chart diagrams, states, and transitions (add, remove, modify, etc.) graphically using drag and drop operations.

6 Conclusion

In this paper, we overviewed our approach for composing Web services using software agents, conversations, and context. Several types of conversation schemas are discussed such as those for inviting Web services to participate in composition. Conversation schemas have been associated with software agents and contextual information. Because needs and interests of users always change, it is important to ensure that the composition of Web services efficiently handles these changes. What is needed is to allow Web services to decide whether to join a composition, what states to take with regard to the outcomes of conversations, and what actions to perform within these states.

[2]http://www.borland.com/jbuilder/enterprise/index.html.

References

[1] L. Ardissono, A. Goy, and G. Petrone. Enabling Conversations with Web Services. In *Proceedings of the Second International Joint Conference on Autonomous Agents & Multi-Agent Systems (AAMAS'2003)*, Melbourne, Australia, 2003.

[2] B. Benatallah, Q. Z. Sheng, and M. Dumas. The Self-Serv Environment for Web Services Composition. *IEEE Internet Computing*, 7(1), January-February 2003.

[3] D. Berardi, D. Calvanese, G. De Giacomo, M. Lenzerini, and M. Mecella. A Foundational Vision for e-Services. In *Proceedings of The Workshop on Web Services, e-Business, and the Semantic Web (WES'2003) held in conjunction with The 15th Conference On Advanced Information Systems Engineering (CAiSE'2003)*, Klagenfurt/Velden, Austria, 2003.

[4] P. Brézillon. Focusing on Context in Human-Centered Computing. *IEEE Intelligent Systems*, 18(3), May/June 2003.

[5] T. Bultan, X. Fu, R. Hull, and J. Su. Conversation Specification: A New Approach to Design and Analysis of E-Service Composition. In *Proceedings of The Twelfth International World Wide Web Conference (WWW'2003)*, Budapest, Hungary, 2003.

[6] B. Burg. Agents in the World of Active Web Services. In *Proceedings of Second Kyoto Meeting on Digital Cities*, Kyoto, Japan, 2001.

[7] J. Dale, D. Levine, F. G. McCabe, G. Arnold, M. Lyel, and H. Kuno. Advanced Web Services. Technical

Report FLA-NARTM02-08, Fujitsu Laboratories of America, Inc., Sunnyvale CA, USA, 2002.

[8] D. Harel and A. Naamad. The STATEMATE Semantics of Statecharts. *ACM Transactions on Software Engineering and Methodology*, 5(4), October 1996.

[9] M. Huhns. Agents as Web Services. *IEEE Internet Computing*, 6(4), July-August 2002.

[10] F. Lin and D. H. Norrie. Schema-based Conversation Modeling for Agent-oriented Manufacturing Systems. *Computers in Industry*, 46(3), October 2001.

[11] Z. Maamar, B. Benatallah, and W. Mansoor. Service Chart Diagrams - Description & Application. In *Proceedings of The Twelfth International World Wide Web Conference (WWW'2003)*, Budapest, Hungary, 2003.

[12] Z. Maamar, S. Kouadri Mostéfaoui, and H. Yahyaoui. A Web Services Composition Approach based on Software Agents and Context. In *Proceedings of The 19th Annual ACM Symposium on Applied Computing (SAC'2004)*, Nicosia, Cyprus, 2004.

[13] B. Medjahed, A. Bouguettaya, and A. Elmagarmid. Composing Web Services on the Semantic Web. *The VLDB Journal, Special Issue on the Semantic Web, Springer Verlag*, 12(4), 2003.

[14] D. A. Menascé. QoS Issues in Web Services. *IEEE Internet Computing*, 6(6), November/December 2002.

[15] J. P. Morgenthal. Web Service Conversations. *Business Integration Journal*, August 2003.