Interaction with Web Services in the Adaptive Web

Liliana Ardissono, Anna Goy, Giovanna Petrone and Marino Segnan

Dipartimento di Informatica, Università di Torino
Corso Svizzera 185, Torino, Italy
{liliana, goy, giovanna, marino}@di.unito.it

Abstract. The provision of personalized services based on the orchestration of simpler Web Services is often viewed as an activity that can be performed in an automated way, without involving the end-user. This paper addresses the need to involve the user in the loop and discusses the communication challenges imposed by this viewpoint. The paper also presents a conversation model for the management of the communication between Web Service consumers and providers aimed at addressing those challenges.

1 Introduction

The composition of Web Services is the focus of the Service Oriented Computing paradigm (SOC), which "utilizes services as fundamental elements for developing applications" [17]. According to [17], several ingredients are needed to enable applications to operate in a SOC environment. For instance:

- A machine-readable service specification language for the description of the services.
- A service discovery protocol for directory services enabling consumers to locate Web Services and to discover their details.
- An open communication protocol enabling consumers to invoke Web Services and to capture the results of the requested operations.
- Quality of Service support, in terms of transactional integrity, security and authentication, and privacy protection.

Moreover, as discussed by Curbera et al. in [9], coordination protocols are needed to orchestrate the invocation of services in complex business logics and to handle failure and recovery during the service execution.

Up to now, the main factor pushing the research on Web Services has been the management of business interactions in electronic commerce, where standard flow languages, such as BPEL4WS [9], have been defined integrate legacy software in open,

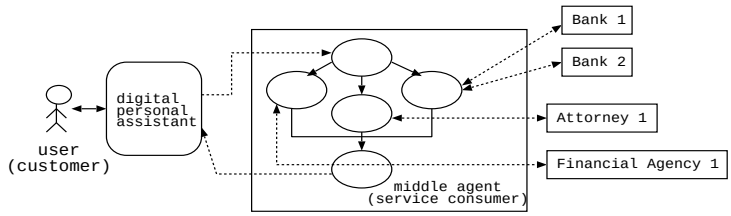

Fig. 1. Interaction between end-user, middle agent and Web Services.

distributed systems. As the main priority of this research is the composition of suppliers in a possibly complex workflow, major efforts have been devoted to orchestrate the invocations of the suppliers. However, the individual service provider has been modeled as a functional unit providing an atomic service that may be requested by means of synchronous, one-shot interactions.

The exploitation of Web Services in the Adaptive Web may be seen as an evolution of the SOC paradigm where the consumer application and/or the integrated Web Services customize the services they offer to the individual end-user. We foresee two main directions (see Figure 1):

- *Making a personalized service, such as a recommender system, available as a Web Service.* For instance, a movie recommender could be extended with a public interface that enables some digital personal assistants to invoke it on behalf of their users. Moreover, a bank might offer a loan customization service that can be exploited by distributed commercial applications to negotiate financial support for their customers. Furthermore, an information service might tailor the type of content it delivers to the end-user's device by choosing different electronic formats depending on bandwidth and device capability constraints.
- *Composing Web Services in a consumer application offering a personalized service to the end-user*. For instance, middle agents, such as real estate agents and car sellers, could develop Web-based applications supporting a completely automated interaction with the customer (or with her digital personal assistant), from the selection of the good to be purchased to the contract definition. Similar to the traditional scenario, populated by human operators, the middle agent will manage a complex workflow, invoking and orchestrating services such as attorneys, banks and/or financial agencies, in order to offer the customer a personalized solution. Moreover, the middle agent could select, out of the pool of alternative Web Services offering the same service, those best satisfying the customer's preferences [3, 4].

The exploitation of adaptive systems as Web Services, as well as the personalized composition and selection of Web Services in a complex consumer application, deeply challenge the Web Service management and composition technologies developed so far. To provide a few examples:

- Rich (but scalable) communication protocols and infrastructures should enable consumers and providers to interact with each other. For example, the composition of a set of Web Services in a workflow may require the management of long-lasting

interactions, to be suspended and resumed, depending on the availability of the invoked service providers. Moreover, highly interactive Web Services might need to manage several interaction turns with the consumers, including repairs to possible service failures, before the services are fulfilled.

– The provided communication protocols and infrastructure have to support a flexible type of interaction with the customer. Some personalized services, such as those supporting the booking of hotel rooms and flights, could in principle interact with a digital personal assistant operating on behalf of the customer. However, other interactive services, such as those supporting the configuration of complex products (e.g., a bicycle, or a loan) and services (e.g., an IP/VPN, or a video conference service) need to explicitly involve the end-user during the decision making, in order to guarantee that the configuration result fulfills her requirements. If this is not the case, they should start a new configuration process. In fact, configuration choices have advantages and disadvantages that, for trust and transparency reasons, should be handled by explicitly involving the user, and possibly repaired by means of a negotiation phase that can hardly be carried out by an automated assistant.

In a scenario similar to the one depicted in Figure 1, the middle agent needs rich communication capabilities to suitably compose and orchestrate the various Web Services into a personalized interactive service for the end-user (customer). Specifically, the middle agent has to manage flexible interactions with the invoked Web Services, as well as to bridge the communication between the end-user and the service providers. Unfortunately, the current standards for the Web Services composition neglect this communication aspect and reduce the invocation of service providers to simple *request-reply* interactions, where the consumer application requests the execution of an operation on the provider and collects the results (the current Web Service communication standards are however evolving to the management of asynchronous communication; e.g., see WSDL 2.0 [21]). Similarly, the recent proposals concerning user-centered Web Services (e.g., [4]), base the retrieval of personalized service results on the iteration of one-shot invocations on the Web Services; in these invocations, the consumer application employs personalization and relaxation strategies to retrieve a bunch of alternative proposals to be presented to the end-user. In contrast, few efforts have been devoted to the enhancement of the interaction between all the parties of the Business to Business relationship, including the customer. In order to address this issue, we have proposed a conversation model for Web Services, aimed at supporting complex interactions, where several messages have to be exchanged before the service is completed. Our framework, described in detail in [2, 1], takes inspiration from the traditional dialog-management approaches developed in the Computational Linguistics research [7]. However, we have simplified the conversation model to take the emerging Web Services standards into account and to make the development of an effective conversation framework feasible.

Before presenting our framework, it is worth noting that we assume that the service discovery phase has been performed and we focus on the service execution phase. The identification of the service provider is a separate activity, to be performed either directly, as suggested in the SOC research (e.g., see [14]), or by exploiting mediation agents, as investigated in the Semantic Web community [19, 12]. Moreover, after

a provider is identified, the consumer should carry an explicit binding activity out in order to match its own ontology with that exploited by the service provider.

In the rest of this paper, we sketch our conversation model and the communication infrastructure we are developing. In particular, we explain in which way the communication features provided by our conversation model support the management of highly interactive Web Services, which need to put the end-user in the loop during the service execution. Specifically, Section 2 describes the framework we are developing; Section 3 presents the related work and Section 4 concludes the paper.

2 A Conversation Model Supporting Web Service Interaction

Our model focuses on two main requirements: first, the model should be compatible with the current standards for the service publication and invocation. Second, the management should be easy for the consumer, in order to make the interaction with multiple providers as seamless and possible. We thus designed a model charging the service provider with the control of the conversation and explicitly guiding the consumer application in the service invocation. specifically:

- The service provider describes the offered service by listing the operations to be invoked.
- The service provider may also publish the interaction flow specification, although this is not necessary for the run time management of the interaction with the consumers.
- As the service provider is in charge of controlling the interaction with the consumers, it has to maintain a local interaction context, for each active conversation.
- At each step, the provider enriches the messages it sends with contextual and turn management information in order to make the consumer aware about the eligible turns it may perform.

2.1 Conversation Flow Language

We adopted a Finite State Automaton representation to specify the conversation flow at the conceptual level, as FSA are a simple formalism and they are well understood; see also [5] and [6]. As far as the conversation turns are concerned, we have clearly separated the representation of the turn-taking activity (for instance, which party can send messages at each conversation step) from the arguments of the messages that the peers exchange. Each conversation turn is aimed at invoking an operation on the receiver: the sender asks the recipient to perform the operation occurring as an argument. For instance, the sender may invoke the execution of a domain-level operation, it may notify the partner about the results of an operation it has just performed (success, failure), or it may suspend/resume the interaction.

The conversation turns are represented by means of send message (*SendM*) activities described as WSDL (Web Services Description Language, [20]) operations and having the following arguments:

- The message sender, which may be either the consumer *C*, or the service provider *S*.

– The recipient of the message (similar).
– The object-level operation to be executed by the message recipient.
– The list of the possible continuations of the conversation (*nextOps*). As the service provider is in control of the interaction, this argument is only present in the messages directed to the consumer. The argument includes the set of alternative operations offered by the provider which the consumer may invoke in the next conversation step.

Our *SendM* activities are requests that the message sender performs to make the receiver execute the object-level operations. This is different from the normal usage of WSDL statements, which directly refer to object-level operations, without wrapping them in conversation turns.

2.2 Conversation Flow of a Product Configuration Service

In order to guide the development of our conversation model we selected a use case concerning the customization of loans. This use case imposes interesting requirements on the interaction between the (human) customer, the middle agent and the Web Services providing basic services; moreover, the use case provides a realistic example where the customer has to be explicitly engaged in the negotiation of the overall service.

The scenario of the loan customization use case is depicted in Figure 1: the customer exploits a personal assistant to contact a middle agent providing personalized loans. The middle agent offers a complex service accessible via a standard Web Service interface. The middle agent exploits and orchestrates "simpler" Web Services (banks, financial agencies, attorneys, etc.) to build the overall service; moreover, it retrieves the customer's requirements from the personal agent. In turn, the personal agent can get the requested information (e.g., her age) from the customer's user model. However, when the needed information is not available, the personal agent asks the customer and forwards her response to the middle agent, which sends the information to the requesting Web Service.

The interaction between personal agent and middle agent may be rather complex and involve different phases; e.g., acquisition of some information about the customer, needed to select the most convenient service providers to be contacted,[1] the management of the interaction with the providers, the signature of the contract. However, in this description, we focus on the loan configuration process, assuming that the middle agent has selected a specific bank to negotiate the loan for the customer. In this case, the loan definition is guided by the bank, which runs the configuration engine for the generation of loan proposals and retrieves the customer information necessary to propose a solution from the middle agent. The key point is that simple customer data, such as her age, might be directly provided by the personal agent, but the agent might not be able to make critical decisions and is expected to ask the user about them. For instance, the customer should be in charge of choosing between different combinations of the loan duration and its maximum rate. The information items requested by the bank have to be

[1] For example, banks adopt stricter policies than financial agencies to decide whether a customer is eligible for funding.

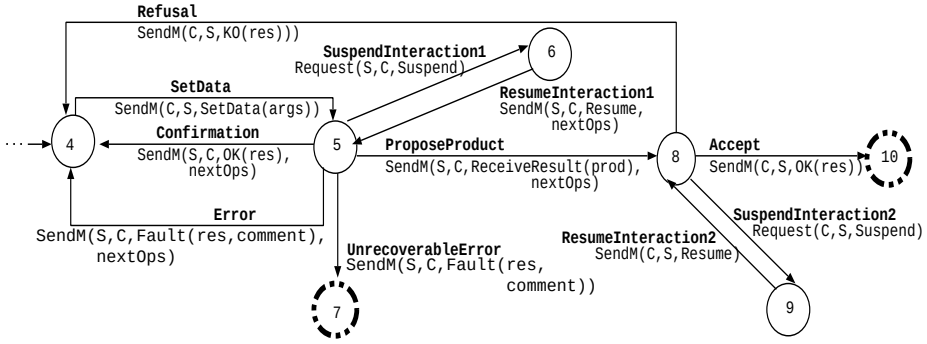

Fig. 2. Portion of the representation of the conversation flow of the loan customization service. Notice that we have labeled the arcs with a boldface identifier (e.g., "Refusal") to simplify the identification of the conversation turns.

provided *during* the configuration process to generate a suitable configuration solution. Moreover, the user can be involved again at the end of the process, in order to choose between alternative solutions; e.g., she may inspect the details of the proposed solution and accept it, or refuse it and start the configuration of a different one.

This use case is also interesting from the viewpoint of the interaction between the middle agent and the orchestrated Web Services (e.g., the bank) because the interaction flow has to be decided during the service execution, depending on the business logics of the invoked Web Service. Therefore, depending on the type of Web Service invoked by the middle agent, different contract features could be set. Furthermore, both the invoked Web Services and the middle agent may need to suspend the conversation and resume it later on, in order to carry out a nested interaction with other parties involved in the business relationship. For instance, the bank might need to invoke some remote services to complete the controls on the good to be funded. Moreover, the middle agent might need to suspend the interaction waiting for the customer to provide the requested data, or to decide whether accepting or rejecting the proposed solution.

Figure 2 shows a portion of the conversation flow specification for the loan customization service. The automaton, held by the bank Web Service, describes the data acquisition and the product proposal phases. The states of the automaton represent the dialog states: the plain circles denote the conversation states and the thick dotted ones (7, 10) are final dialog states. The labels of the arcs represent the conversation turns. The states having more than one output arc are alternative conversation turns.

In the data acquisition phase, the consumer sends the customer data and the requirements on the loan to the Web Service. When the consumer sets data, e.g., the preferred monthly rate (*SetData(args)* operation), the service provider may react in different ways. For instance, it may confirm the correct acquisition of the data (*Confirmation* arc) and enable another invocation of the *SetData* operation to let the consumer set other product features. Or, the Web Service may notify the consumer that there was a failure in the product customization process (*Failure*) and enable the selection of other values

for the conflicting features. The failure management supports the search for a compromise between customer requirements and domain-specific constraints.

The data acquisition phase can end in two ways. Either an unrecoverable error (e.g., the customer's requirements are incompatible with one another and thus they cannot be jointly satisfied), or the customization process succeeds and the service continues the interaction by proposing the product (*ProposeProduct*). The consumer application may accept the proposal (*Accept*) or reject it (*Refusal*), in which case a different loan can be configured; the acceptance/rejection depends on the customer's decisions. Notice that both parties may suspend the interaction and resume it later on, in order to handle delays due to the invocation of sub-suppliers, or the customer's responses.

2.3 Architecture of our Conversation Framework

The previously described conceptual conversation flow specification has to be translated to a standard, executable format, so that service consumers can easily understand it and it can be run by a flow engine within the service provider. Indeed, the *SendM* activity is a normal WSDL operation, with the only difference that some of its arguments are complex XML objects describing the object-level operation to be invoked on the other participant. The executable specification of the automaton can be derived from the conceptual one by translating states and transitions to a suitable process language, such as, for instance, BPEL4WS, which can be executed by existing flow engines.

Although the WSDL representation of the *SendM* operations makes our conversation model compatible with the emerging standards for the Web Service publication and management, it proposes a peculiar exploitation of the constructs offered by such language. Therefore, specific software is needed to manage the interpretation of the incoming WSDL messages (i.e., to extract the information about the next operations and the invoked object-level operation at the recipient side) and to generate the responses. In order to manage the conversation at both sides, the two peers should therefore run, respectively, a *Conversation Manager* and a *Conversation Client* modules. The former is employed by the provider to manage the interaction with the consumers, which would only rely on the light Conversation Client to parse the incoming messages and return the responses.
The Conversation Client has three main responsibilities:

- *Facilitating the reception and interpretation of messages at the consumer side,* especially as far as the interpretation of the eligible continuations of the interaction is concerned (*nextOps* argument of *SendM* messages).
- *Supporting the correct invocation of the operations on the provider,* by performing type and consistency checks to guarantee that the parameter values set by the consumer application satisfy the constraints on the arguments of the operations to be invoked.
- *Facilitating the management of the outbound messages to the provider,* by generating and sending the *SendM* messages that specify the invocation of operations on the provider.

Before binding the invocation of operations to its own business logic, the consumer should download from the Web Service site the Conversation Client to be run during

the service invocation. Moreover, the service provider should exploit the Conversation Manager and run it on its own flow specification (executable version of the conversation automaton). It should be noted that the XML representation of the *SendM* messages supports the interoperability between service consumers and providers, but the Conversation Client we implemented can only be run in a Java-based environment. The idea is therefore that other versions of the Client should be implemented to provide advanced conversation capabilities in different environments such as, for instance, .Net. Details can be found in [1].

3 Related Work

The main difference between our work and other recent approaches to the management of personalized Web Services is that we aim at providing a framework that can be exploited to enhance the interaction with current Web Services, while most of the related work is focused on Semantic Web Services. For instance, Balke and Wagner propose to base the personalization of services for the end-user on the availability of an intelligent system that reasons about Semantic Web Services descriptions and applies logical inference engines to support the run-time invocation of operations [3, 4]. In contrast, our proposal bases the management of personalized services on the possibility of managing long-lasting, flexible communication between the end-user, the middle agent taking care of the Web Service invocation, and the Web Services; moreover, our proposal relies on the emerging Web Service standards and has thus more changes to be employed in the immediate future.

As discussed in [8] and [3, 4], the personalization features offered by Adaptive Hypermedia techniques could also be applied to support the composition of Web Services. All these authors suggest that automated problem solvers could be employed to suggest suitable compositions of Web Services in order to build complex services, i.e., to satisfy the goals to be fulfilled by the complex service under definition. As a matter of fact, the automated generation of flow specifications for Web Service composition is a major goal to be achieved and is attracting a lot of attention in the Semantic Web Services research; e.g., see [16]. However, no techniques have been developed that can easily be applied to carry this activity out in large Business-to-Business application domains, without human intervention. In fact, Web Services are usually described in standard formalisms such as WSDL, that do not specify their semantics. Noticeably, other approaches have been proposed that support the customization of a complex service without requiring semantic information about the services to be composed. For instance, Han et al. [11] explain that a business level composition language should be defined, as an abstraction of the low-level flow specifications typically provided by flow languages such as BPEL4WS, in order to support the service administrator in the customization of the overall service; e.g., the administrator might activate certain subservices only at night, or during certain week days. In this perspective, the technical details concerning the Web Service composition, and the related bindings to services, are resolved by a software engineer, who prepares the environment to be customized by possibly non-expert service administrators.

A different example is provided in more restricted environments, such as the educational one, where the knowledge representation and reasoning frameworks typical of the Semantic Web are attracting a lot of attention. For instance, an interesting approach to bring adaptivity into the Semantic Web is proposed by Dolog et al., who focus on providing personalized access to distributed resources in an open environment; see [10]. Within the perspective of the Semantic Web, where heterogeneous resources (information and services) are described by means of standardized metadata, user requirements should also be represented in a formal and standard language, in order to apply reasoning mechanisms able to personalize the access to such resources.

4 Conclusions

The provision of personalized services based on the orchestration of simpler Web Services is often viewed as an activity to be performed in an automated way, without involving the end-user in the service negotiation. In this paper, we introduced a different viewpoint, which includes the user in the loop, and we discussed the communication challenges imposed by this viewpoint. We also presented a conversation model for the management of the communication between Web Service consumers and providers aimed at addressing those challenges. Our model enables the management of flexible interactions between:

- The end-user who interacts with services by exploiting her own digital assistant;
- The middle agents offering complex services by orchestrating other simpler Web Services;
- The composed Web Services themselves.

The communication features offered by our model support the provision of highly interactive Web Services, such as those carrying out problem solving activities (e.g., the configuration of complex products and services), but can be applied to simpler Web Services, as well.

We believe that our proposal addresses important open issues in the provision of personalized services based on the exploitation of Web Services. However, we recognize that our approach is rather conservative as far as the knowledge-sharing aspects underlying the Web Services cooperation are concerned. For instance, the management of a personalized Web Service might rely on the availability of a user model (either directly accessible, or provided by a User Modeling Server [13]) describing the individual user's preferences; see [18, 15, 10]. Unfortunately, until the Semantic Web becomes a reality, we believe that this scenario will be hardly applicable in wide Business to Business domains.

References

1. L. Ardissono, D. Cardinio, G. Petrone, and M. Segnan. A framework for the server-side management of conversations with Web Services. In *Proc. of the 13th Int. World Wide Web Conference*, pages 124–133, New York, 2004.

2. L. Ardissono, G. Petrone, and M. Segnan. A conversational approach to the interaction with Web Services. *Computational Intelligence*, 20(4), 2004.
3. W.T. Balke and M. Wagner. Towards personalized selection of Web Services. In *Proc. of 12th International World Wide Web Conference (WWW'2003)*, Budapest, 2003.
4. W.T. Balke and M. Wagner. Through different eyes - assessing multiple conceptual views for querying Web Services. In *Proc. of 13th International World Wide Web Conference (WWW'2004)*, New York, 2004.
5. B. Benatallah, F. Casati, F. Toumani, and R. Hamadi. Conceptual modeling of Web Service conversations. In *Proc. Advanced Information Systems Engineering, 15th International Conference, CAiSE 2003*, Klagenfurt, Austria, 2003.
6. D. Berardi, F. De Rosa, L. De Santis, and M. Mecella. Finite state automata as a conceptual model of *e*-services. In *Proc. Integrated Design and Process Technology (IDPT 2003)*, Austin, Texas, 2003.
7. P.R. Cohen and H.J. Levesque. Rational interaction as the basis for communication. In P.R. Cohen, J. Morgan, and M.E. Pollack, editors, *Intentions in communication*, pages 221–255. MIT Press, 1990.
8. O. Conlan, D. Lewis, S. Higel, D. O'Sullivan, and V. Wade. Applying Adaptive Hypermedia techniques to Semantic Web Service composition. In *Proc. International Workshop on Adaptive Hypermedia and Adaptive web-based Systems (AH 2003)*, pages 53–62, Johnstown, PA, 2003.
9. F. Curbera, R. Khalaf, N. Mukhi, S. Tai, and S. Weerawarana. The next step in Web Services. *Communications of the ACM, Special Issue on Service-Oriented Computing*, 46(10), 2003.
10. P. Dolog, N. Henze, W. Nejdl, and M. Sintek. Towards the Adaptive Semantic Web. In *Proc. 1st Workshop on Principles and Practice of Semantic Web Reasoning (PPSWR'03)*, Mumbai, India, 2003.
11. Y. Han, H. Geng, H. Li, J. Xiong, G. Li, B. Holtkamp, R. Gartmann, R. Wagner, and N. Weissenberg. VINCA - a visual and personalized business-level composition language for chaining Web-based Services. In *Proc. International Conference on Service-Oriented computing (ICSOC 2003)*, pages 165–177, Trento, Italy, 2003.
12. T. Kamamura, J. DeBlasio, T. Hasegawa, M. Paolucci, and K. Sycara. Preliminary report of public experiment of semantic service matchmaker with UDDI business registry. In *Proc. International Conference on Service-Oriented computing (ICSOC 2003)*, pages 208–224, Trento, Italy, 2003.
13. A. Kobsa. Generic user modeling systems. *User Modeling and User-Adapted Interaction, Ten Year Anniversary Issue*, 2000.
14. F. Leymann and K. Güntzel. The business grid: providing transactional business processes via grid services. In *Proc. International Conference on Service-Oriented computing (ICSOC 2003)*, pages 256–270, Trento, Italy, 2003.
15. M. Maybury and P. Brusilovsky, editors. *The adaptive Web*, volume 45. Communications of the ACM, 2002.
16. M. Paolucci, K. Sycara, and T. Kamamura. Delivering Semantic Web Services. In *Proc. of 12th International World Wide Web Conference (WWW'2003)*, Budapest, 2003.
17. M.P. Papazoglou and D. Georgakopoulos. Service-oriented computing. *Communications of the ACM*, 46(10), 2003.
18. D. Riecken, editor. *Special Issue on Personalization*, volume 43. Communications of the ACM, 2000.
19. DAML Services Coalition (A. Ankolekar, M. Burstein, J. Hobbs, O. Lassila, D. Martin, S. McIlraith, S. Narayanan, M. Paolucci, T. Payne, K. Sycara, and H. Zeng). DAML-S: Web Service description for the Semantic Web. In *Proc. International Semantic Web Conference*, pages 348–363, Chia Laguna, Italy, 2002.

20. W3C. Web Services Definition Language. http://www.w3.org/TR/wsdl, 2002.
21. W3C. Web Services Definition Language Version 2.0. http://www.w3.org/TR/wsdl20/, 2004.