Towards a Web Services Composition Approach for Establishing Virtual Enterprises: a Software Agent-based Perspective

Zakaria Maamar[1] and Ghazi Alkhatib[2]

[1]Zayed University, Dubai, U.A.E, zakaria.maamar@zu.ac.ae
[2]Qatar College of technology, Doha, Qatar, Alkhatib@qu.edu.qa

Abstract

Web services technologies aim at supporting the definition of Web services, their advertisement, and their binding for triggering purposes. A Web service has the following features: independent as much as possible from specific platforms and computing paradigms; developed mainly for inter-organizational situations rather than for intra-organizational situations; and easily composable (its composition with other Web services does not require the development of complex adapters). A potential strategy that implements such a support for composite services consists of merging business processes. This has resulted into the deployment of Virtual Enterprises (VE}. In our work, we aim at establishing VEs through the combination of composite services and Software Agents (SAs). This combination occurs at two levels. The first level is reserved to Web services and takes care of the following aspects: identify which businesses of the VE will provision Web services, when and where the provisioning of Web services will happen, how the Web services from separate businesses will coordinate their activities and exchange information, what back-up strategies will be used in case the execution of Web services fails. The second level is reserved to agents and consists of identifying what types of agents will be needed for searching for the businesses that have the capacity to meet the outsourcing requirements, tracking the execution of the Web services, and implementing corrective actions according to the back-up strategies.

Key Words: Web services, composite web service, software agents, virtual enterprises.

1 Introduction

Web services are emerging as a major technology for achieving automated interactions between distributed and heterogeneous applications [2]. Various standards back this achievement such as WSDL, UDDI and SOAP [5]. These standards aim at supporting the definition of Web services (also called services in the rest of this paper), their advertisement, and their binding for triggering purposes. The advantages of Web services highlight their capacity to be composed into high-level business processes [2]. Usually, Composite Services (CSs) denote such business processes.

The increasing demand of users for high quality and timely information is putting business under the pressure of continuously adjusting their known-how and seeking for more support from other businesses. One of the strategies that implement a similar support consists of merging business processes, which results in Virtual Enterprises (VEs) [13]. A VE is a temporarily network of independent businesses that join their efforts until some objectives are reached. Outsourcing operations between businesses is a good illustration of the operations of a VA. Reasons for outsourcing are multiple including cost-effectiveness and expertise-availability.

In our work, we aim to establishing VEs by combining CSs and Software Agents (SAs) [8]. This combination occurs at two levels. The first level is about Web services and consists of the following aspects: identify which business of the VE provision Web services, when and where the provisioning of Web services happen, how the Web services form separate businesses coordinate their activities and exchange information, and what back-up strategies are used in case of exceptions. The second level is about software agents and consists of identifying what types of agents are needed for searching for the businesses that have the capacity to met the outsourcing requirements, tracking the execution of the Web services, and implementing corrective actions according to the back-up strategies. The rest of this paper is organized as follows. Section 2 overviews the concepts of software agent, Web service, and composite service. Section 3 introduces the process of agentifying composite services. The different types of agents and their roles as well are also discussed in this section. Finally, Section 4 summarizes the paper.

2 Background

A software agent is a piece if software that acts autonomously to undertake tasks on user's behalf. A SA exhibits a number of features that make it different from other traditional components: autonomous, goal-oriented, collaborative, flexible, self-starting, temporal continuity, character, communicative, adaptive and mobile.

A Web service is an accessible application that can be automatically discovered and involved by other applications (and humans). We adopt the definition which considers an application as a Web service if it is [2]: independent as much as possible from specific platforms and computing paradigm; developed mainly for inter-organizational situations rather than or intra-organizational situations; and easily composable (i.e., its composition with other Web services does not require the development of complex adapters).

For the purpose of this research, a Web service is specified with a Service Chart Diagram (SCD) [10]. A SCD leverages an UML state chart diagrams [6], putting the emphasis on the context surrounding the execution of a service rather than only on the states that a service takes (Figure 1). A SCD wraps the states that a service takes into four perspectives, each perspective has a set of attribute. The flow perspective corresponds to the execution chronology of a composition of services (previous services/next services attributes). The business perspective identifies the organizations that are ready to provide a service (business attributes). The information perspective identifies the data that are exchanged between services (adapt from pervious services/data for next services attributes). Finally, the performance perspective illustrates the way a service is involved for execution whether remotely or locally (performance type attribute). It should be noted that the stats of a service constitute a state chart diagram.

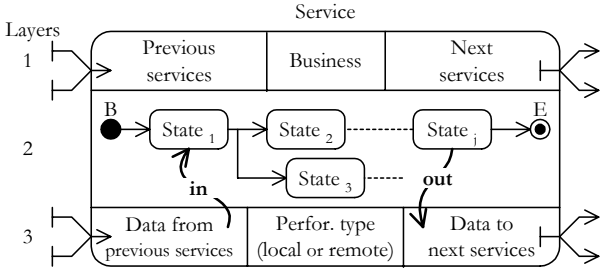

Figure 1 Service chart diagram of a Web service

A composite server consists of component services that are either primitive (i.e. Web services) or composite. In what follows, we summarize how a composite service can be developed [4].

- Proactive composition vs. Reactive compositions: a proactive composition is an off-line process that gathers in-advance available component services to constitute a composite service. A reactive composition is the process of creating composite services on-the-fly. A composite service is devised on a request-basis from users.
- Mandatory-composite services vs. Optional-composite services: a mandatory composite service corresponds to the compulsory participation of all the component services in the execution process. An optional composite service does not necessarily involve all the component services. Some component services can be skipped during execution due to various reasons such as possibility of substitution or non-availability.

Because a CS consists of several component services, the process model underlying that CS is specified as a service chart diagram whose states are associated with the service chart diagrams of the component services, and transition are labeled with events, condition, and variable assignments, operation (Figure 2).

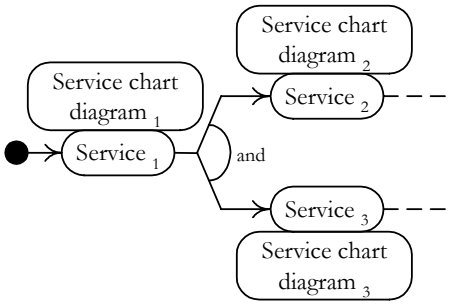

Figure 2 State chart diagram of a composite service

3 Agentification of Web services integration

3.1 Agent-based deployment

The agentification process aims at identifying the relevant types and roles of agent that handle the integration of CSs. This integration may require the outsourcing of certain parts of a CS to third parties. Therefore, the identification of these parties requires brokering mechanisms that ease the search of who has the capacity to meet the requirements of the outsourced parts. In addition, when merging the different parts of the CS, it is necessary not only to check that there are violations of the temporal and resource constraints of these parts, but also to ensure that the necessary relations are maintained in the merged composite services.

For the needs of the agentification process, we considered two types of agent: service-agent and manager-agent (Figure 3). A service-agent is associated with a Web service and thus, is responsible of managing its associated SCD (Figure 1). In addition a service-agent is seen as a wrapper that runs on top of a Web service [7]. All the requests that are submitted to a Web service are handled but its respective service-agent. A service-agent makes all the decisions on behalf of a Web service with regard to taking part in a composite service. A service-agent may refuse to participate to a composite service because of time of availability, overloaded status, etc.

A manager-agent is associated with a composite service and supervises the work of service-agents. The manager-agent ensures that the collaboration between the service-agents is efficient and no conflicts between them are happening for instance regarding the resources (i.e., because of concurrent services or overlapping periods of use) on which their service will be carried out. Because of the outsourcing that may occur during Web services integration, a manager-agent can be specialized into two types: outsourcing-manager-agent and outsourced-manager-agent. The links between outsourcing-manager-agent and outsourced-manager-agent illustrate establishing a virtual enterprise. In Figure 3 composite service$_1$ has 4 component services: 3 primitives namely Web services$_{1,2,3}$ and 1 composite service namely CS$_2$. Composite service$_2$ has two primitive component services Web services$_{4,5}$. Web services$_{4,5}$ are outsourced to another manager-agent.

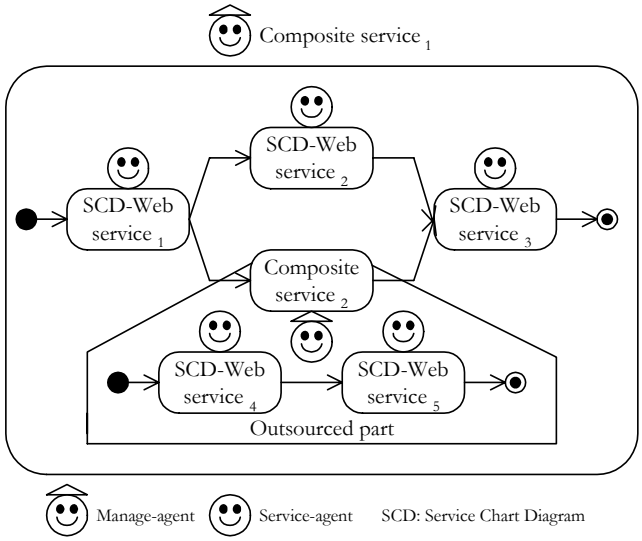

Figure 3 Agents for Web services integration

3.2 Tracking the status of a composite service

In this paper, we associated a composite service with a structure referred to as context. The rationale of this context is to track the status of a composite service by knowing for example the status of its component services, which component services are being outsourced, etc. In the following, we list some of the arguments that structure a composite service's context.

- Label: corresponds to the identifier of the composite service.
- Manager-agent label: corresponds to the identifier of the manager-agent in charge of the composite service.
- Previous Web services (outsourced (yes/no)): indicates which Web services of the composite service have been executed with regard to the current Web services. It also indicates whether these services have been outsourced.
- Current Web services (outsourced (yes/no)): indicates which Web services of the composite service are currently under execution. It also indicates whether these services are currently outsourced.
- Next Web services (outsourced (yes/no)): indicates which Web services of the composite service will be called for execution with regard to the current Web services. It also indicates whether these services will be outsourced.
- Begin time: informs when the execution of the composite service has started.

Our primary objective is to enhance composite services with context awareness mechanisms. This can be achieved by satisfying the requirements of adaptability and dynamic composition. The agentification approach of Figure 3 satisfies both requirements. A composite service might have to adapt its list of component Web services because of the availability of certain of these components. In addition since Web services participate in a composition on a request-basis, this means that a dynamic composition is supported. We organize the context of a composite service along three interconnected perspectives (Figure 4):

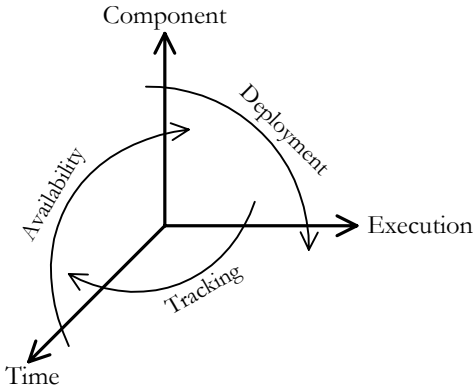

Figure 4 Perspectives of a service-centric context

- Component perspective: is about deploying Web services, assigning them to composite services, and getting the next component Web services ready for execution.
- Execution perspective: is about satisfying the computing requirements of the component Web services, tracking the execution status of these components, and avoiding conflicts on these computing resources.
- Time perspective: is about time-related parameters of the component Web service instances and constraints on these components (e.g., period of request, period of availability, end-time expected).

3.3 Web services engaging in conversations

In a reactive composition, the selection of Web services of a composite service is done during run-time. This selection is entrusted to services-agents that engage in conversations with the respective service-agents decide whether their respective Web services could join a composition, what states these Web services should take based on the outcome of conversations, and what activities these Web services should perform within these states.

When a Web service is being executed, its service-agent initiates conversations with the service-agents of the next Web services that are due for execution. The purpose of these conversations is twofold: (i) invite service-agents and thus their Web services to join the composition process; and (ii) ensure that these Web services are ready for execution in case the invitation is approved. Furthermore, conversations between service-agents enable dealing with the composability issue of Web services [11]. For instance mapping operations of exchanged parameters between Web services can be required. Ensuring the composability of Web services can be achieved through conversations. Service-agents engage in conversations to agree on what to exchange, how to exchange, when to exchange, and what to expect from exchange.

Figure 5 represents a conversation diagram between two service-agents whose Web services are components of a composite of a composite service CS. CS consists of n component

services (Web service$_{1\ ..,i,j,...n}$). For the sake of simplicity, the Web services are sequentially ordered. In this Figure, rounded rectangles are states, italic sentences are conversation, and the numbers are the chronology of conversation. Initially, Web service takes an execution state. While its service-agent takes a conversation state where activities interact with the service-agent of the next Web services (i.e. Web service) are carried out. In Figure 5 the first established conversation consists of sending a request from the service-agent of Web service$_i$ to the service-agent of Web service$_j$ to join the composite service (1). This composite service is decomposed into three segments. The first segment corresponds to the Web services that have completed their execution (Web service$_{1,...,i-1}$). The second segment corresponds to the Web service that is currently under execution (Web service $_i$). Finally, the third segment corresponds to the rest of the composite service that is under preparation (Web service$_{j,..,n}$). Initially, the service-agent of Web service$_j$ is in standby mode waiting to receive invitations of joining composition. When it receives an invitation, the service-agent of Web service$_j$ enters the assessment state. In that state, the service-agent of Web service$_j$ considers its constraints and makes a decision about to decline the invitation, delay its making decision, or accept the invitation. Examples on constraints could be the maximum number of active requests that invoke simultaneously a Web service or the period of Web service unavailability for some maintenance work.

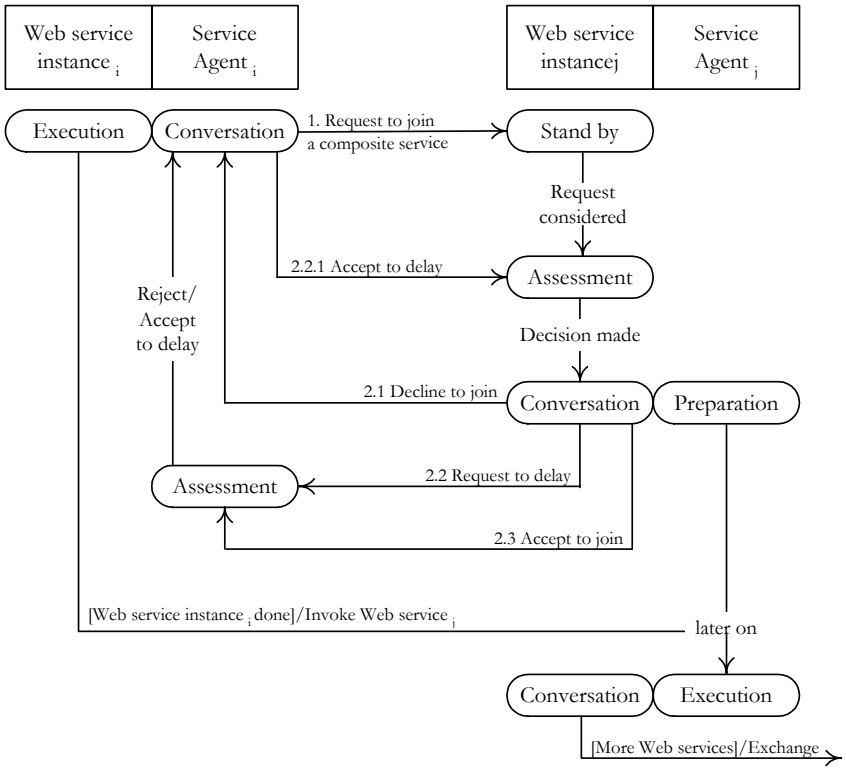

Figure 5 Conversation diagram between service-agents

- Case A. service-agent of Web service$_j$ declines the invitation of the service-agent of Web service$_i$. Thus, a conversation message is sent back from the service-agent of Web service$_j$ to the service-agent Web service $_i$ for notification (2.1). The service-agent enters again the conversation state, asking the service-agent of another Web service $_k$ ($k \neq j$) to join the composite service (1).
- Case B. service-agent of Web services$_j$ cannot make a decision before the response deadline that the service-agent of Web services$_i$ has set. Thus, the service-agent of Web service$_j$ requests from the service-agent of Web service$_i$ to extend this deadline

(2.2). The service-agent of Web service$_i$ has two alternatives: a) refuse the request of the service-agent of Web service$_j$; this means that the service-agent of Web service$_i$ has to look again for another service-agent of Web service (Case A.) or b) accept the request of the service-agent of Web service$_j$; this means that the service-agent of Web service$_j$ will get notified about the acceptance of the service-agent of Web service$_i$ (2.2.1). In alternative b) the service agent of Web service$_j$ enters the assessments state again in order to make a decision. The service-agent of Web service$_j$ may request an extension of the deadline for several reasons such as the service-agent cannot commit additional instances of Web service$_j$ while other instances have not yet completed their execution.

- Case C. service-agent of Web service$_j$ accepts to join the composite service. Thus, it notifies its acceptance to the service-agent of Web service$_i$ (2.3) and a Web Service Level Agreement (WSLA) is established [9] between both agents. At the same time the service-agent prepares Web service$_j$ for execution.

When Web service $_i$ finishes is execution, it invokes Web service$_j$ according to the agreement that was established in Case C. Therefore, Web service$_j$ enters the execution state. At the same time, its respective service-agent initiates conversation with the service-agents of the next Web services. The service-agent of Web service$_j$ adopts the aforementioned approach.

4 Summary

In this paper, we briefly presented our work that consists of integrating Web services using software agents. This results in deploying virtual enterprises. Since it is expected that a virtual enterprise will be joined by an unspecified number of businesses providing Web services, their composition is deemed appropriate to the creation of sub-domains of Web services capable of satisfying customers' requests. Two types of software agents are employed to handle the links and conversations between individual Web services and composite services: service-agent and manager-agents, respectively. This paper considered the simple version of serial execution of Web services. If parallel execution is incorporated into the model, grid computing could be employed to monitor and control concurrent access of different Web services and outsourcing decisions. Another area of further research is workflow management, transaction management and message queuing of all activities involved in the VE as presented in this research.

5 References

[1] B. Benatallah and F. Casati (Guest Editors). Special Issue on Web Services. Distributed and Parallel Databases, Kluwer Academic Publishers, 12(2-3) September 2003.

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

[3] A. Berfeild, P.K. Chrysanthid, I. Tsamardinos, M. E. Pollach, and S. Banerjee. A Scheme for Integration E-Services in Establishing Virtual Enterprises. In Proceedings of the Twelfth International Workshop on Research Issues in Data Engineering: Engineering e-Commerce/e-Business Systems (RIDE'2002), San Jose, USA, 2002.

[4] D. Chakraborty and A. Joshi. Dynamic Service Composition: State-of-the-Art and

Research Directions. Technical report, TR-CS-01-19, Department of Computer Science and Electrical Engineering, UMBC, USA, 2001.

[5] F. Curbera, M. Duftler, R. Khalaf, W. Nagy, N. Mukhi, and S. Weerawaraana. Unraveling the Web Service Web: An Introduction to SOAP, WSDL, and UDDI. IEEE Internet Computing, 6(2), March/April 2002.

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

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

[8] N. Jennings, K. Sycara, and M. Wooldridge. A Roadmap of Agent Research and Development. Autonomous Agents and Multi-Agent Systems, Kluwer Academics Publishers, 1(1), 1998.

[9] H. Ludwig, A. Keller, A. Dah, and R.King. A Service Level Agreement Language for Dynamic Electronic Services. In Proceedings of the 4th IEEE International Workshop on Advanced Issues if E-Commerce and Web-Based Information system (WECWIS'2002), Newport Beach, California, USA, 2002.

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

[11] B. Medjahed, A. Rezgui, A. Bouguettaya, and M. Ozzunai. Infrastructure for E-Government Web Services. IEEE Internet Computing, 7(1), January / February 2003.

[12] A. Tsalagatidou and T. Pilioura. An overview of Standards and Related Technology in Web Services. Distributed and Parallel Databases, Kluwer Academic Publishers, 12(3) September 2002.

[13] N. Venkatraman and Henderson J.C Real Strategies for Virtual Organizing. Sloan Management Review, 40(1), Fall 1998.

Top