Developing B2B Virtual Enterprises

Ghazi Alkhatib
Applied Science University, Jordan

Zakaria Maamar
Zayed University - Dubai, United Arab Emirates

INTRODUCTION

Nowadays, Web services are emerging as a major technology for achieving automated interactions between distributed and heterogeneous applications (Benatallah, Sheng, & Dumas, 2003). Various technologies are behind this achievement including WSDL, UDDI, and SOAP[1]. (Curbera, Duftler, Khalaf et. al. 2002) These technologies aim at supporting the definition of services[2], their advertisement, and their binding for triggering purposes. The advantages of Web services have already been demonstrated and highlight their capacity to be composed into high-level business process (Benatallah et al., 2003). Usually, *composite services* (CS) denote business processes and are meant to be offered to users who have needs to satisfy.

The increasing demand of users for high quality and timely information is putting business under pressure of continuously adjusting their known-how and seeking for more support from other businesses. One of the strategies that implement such support has consisted of merging business process. This has resulted into the deployment of *virtual enterprises* (VE) (Venkatraman, & Henderson, 1998). A VE is a temporarily network of independent businesses that decide to join their efforts until certain objectives are reached. Outsourcing operations between businesses is a good illustration of the operations of a VE. Reasons for outsourcing are multiple, including cost-effectiveness and expertise-availability.

In our work, we aim at establishing VEs through combinations of CSs and *software agents* (SA) (Jennings, Sycara, & Wooldridge, 1998). This combination occurs at two levels. The first level is reserved to Web services and takes care of the following aspects: identify which business 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 met the outsourcing requirements, tracking the execution of the Web services and implementing corrective actions according to the back-up strategies.

The remainder of this paper is as follows: Section 2 overviews the core concepts of our research mainly software agents, Web services, and composite services. 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 and identifies issues for further research.

BACKGROUND

Software Agents (SA)

An SA is a piece of software that acts autonomously to undertake tasks on user's behalf (Jennings, Sycara, & Wooldridge, 1998). Many SA's design is based on the approach that the user needs only to specify a high-level goal instead of issuing explicit instructions, leaving the 'how" and "when" decisions to the agent. An 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.

Web Services (WS)

A Web service is an accessible application that can be automatically discovered and involved by other applications (and humans). We adopt a definition which considers an application as a Web service if it is: independent as much as possible from specific platforms and computing paradigm; developed mainly

for inter-organizational situations rather than for intra-organizational situations; and easily composed, that is its composition with other Web services does not require the development of complex adapter) (Benatallah et al. 2003, Tsalagatidou, & Pilioura, 2002)

For the purpose of this research, a Web service is specified with a *service chart diagram* (SCD). (Maamar, Benatallah, & Mansoorm, 2003) An SCD leverages the UML state chart diagrams (Harel & Naamad, 1996), 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 having a set of attributes. 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 states of a service constitute a state chart diagram.

Composite Services

A composite service consists of component services that are either primitive (i.e., Web services) or composite. In the following statements, we summarize the different ways a composite service can be developed (Chakraborty & Joshi, 2001).

Proactive composition vs. Reactive compositions: proactive composition is an off-line process that gathers in advance available component services to form CSs. This means that CSs are pre-compiled and ready to be launched upon user's requests. In a proactive composition, the component services are usually stable and constantly running on resource-rich platforms. Reservation of air tickets is a good illustration of this type of composition. With regard to reactive composition, it is the process of creating CSs on the fly. A CS is devised on a request-basis from customers. Because of the fly-property, a component manager that ensures the smooth identification of and the efficient collaboration between the component services is necessary. Despite the complexity that could feature a reactive composition process, it presets several advantages, including the consideration of the component services' current states and the runtime optimization of certain arguments such as bandwidth and execution charges.

Mandatory-composite services vs. Optional-composite services: a mandatory CS illustrates the compulsory participation of all component services to the execution process. Because it is expected that the component services will be spread over the net, the reliability of the execution process of these component services affect the reliability of the whole composite service to which they belong. On the other hand, an optional CS does not necessarily require the participation of all component services. Certain component services can be skipped during execution for various reasons such as possibility of replacement and non-availability.

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

Figure 1. Representation of a service chart diagram