Mobile Agent System for Web Services Integration in Pervasive Networks

Fuyuki Ishikawa[1], Nobukazu Yoshioka[2], Yasuyuki Tahara[2], Shinichi Honiden[2,1]

[1] Graduate School of Information Science and Technology
The University of Tokyo, 7-3-1 Hongo, Bunkyo-ku, Tokyo 113-8654, Japan
[2] National Institute of Informatics,
2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo 101-8430, Japan

Abstract. Web Services integration using languages such as BPEL is to be applied not only on the Internet but also in pervasive networks using wireless mobile devices. However, in such a network it is necessary to deal with constraints in resources, typically the relative narrowness and instability of network connections. Our study adopts the mobile agent technology in response to this problem and presents a mobile agent system for Web Services integration. In our system, the physical behaviors of mobile agents, including migration and cloning, are separated from the integration logic and are described as simple rules added to the BPEL integration process description. This makes it possible to add or change physical behaviors according to the environmental conditions without modification of the BPEL description.

1 Introduction

Research on XML Web Service technologies for the dynamic integration of services has recently commenced[1]. The goal of the Web Services effort is to achieve universal interoperability between applications by using Web standards. Web Services use a loosely coupled integration model to allow flexible integration of heterogeneous systems. Starting from the basic specifications for messaging and service description, the Web Service technologies now include languages for specifying how to integrate existing services and compose a just-in-time service. BPEL (Business Process Execution Language for Web Services)[2] is common one of such languages and defines a notation of an integration process based on interactions between the process and its partners (existing services or clients).

On the other hand, the rapid development of mobile devices and wireless technologies is leading to new applications in pervasive networks. In pervasive networks, both users and service providers utilize mobile computers/devices communicating with each other using wireless connections. For example, the user accesses services in the wireless LAN to obtain information about the shops he is about to visit, as he accesses services on the Internet using wired connections.

As the Web Services technologies provide the basis for interoperability and dynamic discovery and integration of services, their benefits exist not only on

Ishikawa F., Yoshioka N., Tahara Y. and Honiden S. (2004).
Mobile Agent System for Web Services Integration in Pervasive Networks.
In *Proceedings of the 1st International Workshop on Ubiquitous Computing*, pages 38-47
DOI: 10.5220/0002681900380047

the Internet but also in the pervasive networks. Now Web Services can be implemented on mobile devices[3]. However, since the Web Services effort has mainly targeted services on the Internet and the interaction models are heavily based on remote messaging, they cannot be applied directly in pervasive networks, due to the relative narrowness and instability of wireless connections. As a typical example, it becomes inadequate in pervasive networks to exchange multimedia data itself between distributed services in order to compose a multimedia information collection service.

Our study adopts the mobile agent technology in response to this problem and presents a mobile agent system for Web Services integration. A mobile agent is a software component that has the ability to move from one host to another while preserving its state. It has been utilized for adaptation to various environments, typically mobile computing environments[4, 5]. On realizing integration by mobile agents, it is important to facilitate change of physical behaviors without modification of the integration logic, in order to adapt to changes in the environments. Considering this requirement, in our system the physical behaviors of mobile agents, including migration and cloning, are separated from the integration logic and described as simple rules added to the BPEL description. This makes it possible to add or change physical behaviors according to the environmental conditions without modification of the BPEL description.

In Section 2 of this paper, we discuss the problems in Web Services integration in pervasive networks and show our approach utilizing mobile agents. We present our system model in Section 3 and discuss implementation issues in Section 4. We present several considerations and the conclusion in Section 5 and 6.

2 Web Services Integration in Pervasive Networks

In this section, we present a brief introduction of the Web Services integration using BPEL and discuss a problem in realizing it in pervasive networks. We then describe how we apply the mobile agent technology to the problem.

2.1 Web Services Integration using BPEL

BPEL (Business Process Execution Language for Web Services)[2] is an XML language used to describe an executable process that provides a new service by integrating existing services.[1] Figure 1 illustrates an example of the structure of such a process. This process first receives a request from a client, concurrently invokes two data services and then an analysis/search service for each obtained data, and finally assigns the results to a variable and replies to the service consumer. In BPEL each involved task is called an *activity*. The example shown in Fig. 1 includes the basic activities of *receive*, *invoke*, *assign*, and *reply*. These activities are nested in structured activities that represent activity control information (*sequence* and *flow* in the figure). A *sequence* activity indicates sequential

[1] Interfaces of all these services are described using WSDL (Web Services Description Language)[6].

execution of nested activities and a *flow* activity indicates that control flow of nested activities is specified in a graph-oriented manner with *link* elements(the arrows in the figure). In this way, BPEL defines a notation of an integration process based on interactions between the process and its partners (existing services or clients). BPEL includes many other activities for branch, iteration, fault handling, event handling, and compensation.

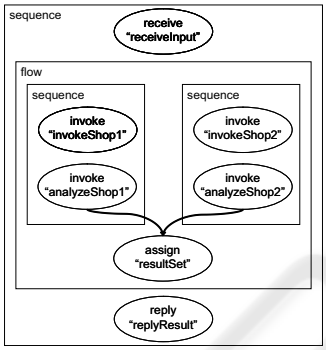

Fig. 1. Example of BPEL Process Structure

The BPEL process described here integrates two types of services: (1) services providing multimedia information with a standardized general interface for queries based on metadata, and (2) services providing methods for specialized, sophisticated, or original analysis/search of the multimedia data itself. Like this, BPEL and the underlying Web Service technologies promote the dynamic and arbitrary combination of services, facilitating composition of an original just-in-time service by selecting favorite service providers as partners. [2]

2.2 Problems in Pervasive Networks

As the Web Services technologies provide the basis for interoperability and dynamic discovery and integration of services, their benefits exist not only on the Internet but also in pervasive networks. However, since the Web Services effort has mainly targeted services on the Internet and the interaction models are heavily based on remote messaging, they cannot be applied directly to some applications in pervasive networks, due to the relative narrowness and instability of network connections. For example, the composition of multimedia information service shown in Section 2.1 becomes inadequate in pervasive networks with limited communication resources, since this requires exchange of the multimedia data itself between distributed services. This problem is significant especially when temporary environments without sufficient resources are considered.

[2] This selection is done by associating service bindings to a process instance, or specifying a process that finds adequate service providers with directroy services by itself.

In the remainder of this paper, we consider adoption of the mobile agent technology in response to this problem. Note that we limit our discussion in this paper to this problem. We are not concerned with another problem of dynamic discovery and selection of (possibly temporary) available services, assuming some mechanisms are available such as directory services and multicast protocols.

2.3 Adoption of Mobile Agents

A mobile agent is a software component that has the ability to move from one host to another while preserving its state[4, 5]. It has been utilized especially for information retrieval applications and mobile computing, with its migration ability facilitating the following typical behaviors.

Local Access A mobile agent can migrate to the host where the service stays so that it enables fast local interaction with the service, at the cost of migration overhead. In addition, it can reduce network traffic in information retrieval applications, by carrying on the analysis/search method to the service host and bringing (or sending) back only the necessary result.

Work Away from the User's Device A mobile agent can select hosts on which it achieves tasks so that it can interact with the user on his device and achieve other tasks on other more stable hosts. This decreases the dependence on an unstable connection of the moving user's device while maintaining fast responses during interactions with the user. Moreover, the user only have to connect when needed, e.g. when launching and receiving agents.

Existing mobile agent systems also support cloning as well as migration[7, 8]. Cloning complements migration behaviors with additional abilities, for example, concurrent local access to multiple services and reduction in migration cost by dispatching a clone with only necessary data.

To realize Web Services integration by mobile agents, it is necessary to add descriptions of these physical behaviors to the integration logic. It is important to facilitate change of physical behaviors without modification of the integration logic, in order to adapt to changes in the environments. Taking this requirement into consideration, we propose a mobile agent system for Web Services integration, where the descriptions of integration logic and physical behaviors are separated. Our system aims to:

- Utilize the standard language, BPEL, to describe integration logic.
- Readily introduce typical physical behaviors to an integration process using simple rule descriptions.

Our discussion in this paper thus targets how to describe the behaviors of a mobile agent that integrates services.

3 Mobile Agent System for Web Services Integration

In this section, we present our mobile agent system for Web Services integration, taking particular note of the descriptions of the agents' behaviors.

3.1 System Model

Figure 2 shows our model for integration by mobile agents. In this system, a mobile agent that achieves Web Services integration is defined with three parts of descriptions: (1) **a BPEL process description** expresses the integration logic and is interpreted by the agent, (2) **behavior rule descriptions** decide the physical behaviors of the agent, and (3) **packed services** are carried by the agent and invoked locally in the BPEL process. The provider of an composed service registers these to a platform that supports execution of agents.

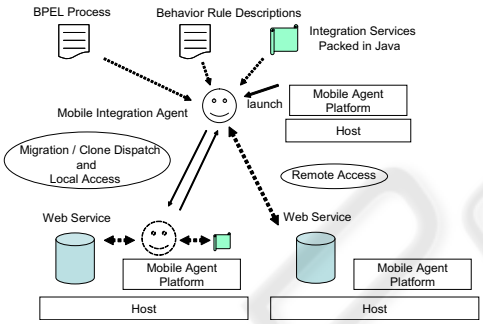

Fig. 2. Web Services Integration by Mobile Agents

The combination of a BPEL process and behavior rule descriptions decides the behaviors of the agent. Behavior rule descriptions are explained in more detail later in this section. Packed services are invoked in the same way as external services, but the agent dynamically deploys them on the host where it stays. We currently limit this to only Java classes referred by WSDL Java Extension. This functionality to carry services is necessary, for example, in order to achieve local access and analysis/search as mentioned in Section 2.3.

Note that this paper does not cover how to select partners from available service providers and assumes that ordered lists for available service providers are supplied when the agent is instantiated.

3.2 Overview of Behavior Rule Descriptions

In our behavior rule descriptions, each rule associates a physical behavior, i.e. migration or cloning, to an *execution block*, that is, a set of activities to be executed in succession. An execution block consists of any form of: (1) **an activity**, (2) activities that are nested directly in a sequence activity and are to be executed successively (**subsequence**), or (3) activities that are nested directly in a flow activity and make a connected graph starting from one activity (**subflow**).[3]

[3] This definition comes from the fact that only sequence and flow activities can nest multiple activities directly.

In all cases there is one activity that is to be executed first in the block and we call this the *start activity* of the block. We allow a block to nest in another block recursively but not to cross the boundary of another block.

Here we explain two types of primitive rules with the example rules in Fig. 3. These rules are associated with the example BPEL process in Fig. 1.

Migration A *migrate* rule indicates that the agent migrates to a certain host and executes the block on the host if the migration is successful, excluding other possible occurrences of migration associated with concurrent blocks. The purpose of associating an execution block to a host, instead of indicating a point where migration occurs, is to assure realization of successive local interactions, e.g. local access and analysis/search mentioned in Section 2.3. Conflicting migration of concurrent blocks are detected before execution. The migrate rule in the figure indicates that the agent migrates to the host where the information service stays (the keyword *local*) and on the host it invokes the service ("invokeShop1") and the carried service for analysis ("analyzeShop1"). If it fails to migrate, it continues the trial until timeout (the *retry* element). These options are described again later in this section.

Cloning A *clone* rule indicates that the agent creates a new clone agent and allocates the execution block to it. Since BPEL is capable of describing concurrency control with flow activities such as join conditions, our clone rule only describes which activities are to be executed by the created clone. The clone rule in the figure indicates that the agent creates its clone and makes it access another information service locally as indicated by the nested migrate rule.

```
<migrate block="invokeShop1 analyzeShop1">
  <target> <local /> </target>
  <retry time="30" />
</migrate>

<clone block="invokeShop2 analyzeShop2">
  <migrate block="invokeShop2 analyzeShop2">
    <target> <local /> </target>
  </migrate>
</clone>
```

Fig. 3. Example of Behavior Rule Description

Interpretation of the combined BPEL process and behavior rule descriptions is done in almost the same way as the interpretation of a standard BPEL process. The difference is that the controls of physical behaviors, such as migration actions and data exchange between agents, are inserted according to the behavior rule descriptions, before and after execution of each activity. We are presently investigating the formal definition of these behaviors. In our system, an agent is an entity that is responsible for execution of a certain set of activities in the BPEL process. Intuitively, migration changes the host which the agent belongs

to and cloning changes the agent which the activities in the execution block belong to. These behaviors preserves the original semantic of BPEL, in terms of control and data flow between activities and actions associated to each activity.

3.3 Options for Migration Controls

Here, we briefly describe some options for migration controls to facilitate realizing typical behaviors.

First of all, it is necessary to specify how to decide the target host of migration. Apart from specifying a host address directly, our system supports various options for this. We show some characteristic ones below.

Ordered list of hosts The agent tries to migrate to hosts in the list one by one until it succeeds or it fails all.

***local* keyword** The agent migrates to the host where the service invoked in the start activity stays. This allows programmers to realize local interactions easily without being concerned about which service providers are actually chosen possibly in runtime. This requires addressing support of platforms, as we will discuss in the next section.

Reference to an allocation service The agent utilizes a provided allocation service that returns a list of available hosts according to the environmental conditions.

Secondly, it is necessary to handle migration failure as we are taking unstable wireless connections into consideration. As shown in the example in Section 3.2, behavior rule descriptions enable agents to continue the trial until timeout. In addition, specifying a list of target hosts, as mentioned above, indicates migration retry changing the target host, and specifying a *change* keyword indicates migration retry for local access by changing the service provider to interact with according to the given list of available providers. When the agent still fails to migrate after such retries it accesses services remotely as a stationary BPEL engine by default. However, in cases where migration is very important, for example when exchanging enormous amounts of data, the agent may give up execution of the block, throwing an associated fault. Behavior rule descriptions also facilitate such association of migration failure to fault throwing in the BPEL process.

4 Implementation

In this section, we show the current implementation status and discuss some implementation issues particular to our system.

4.1 Current Status and Issues

We implemented a prototype system using the Bee-gent framework[7] and examined the effectiveness of mobility in Web Services integration through several experiments. In this paper we don't present these experiments as space is limited

and the experiments are basic and traditional ones for evaluation of the mobile agent paradigm.

Here, we briefly discuss some implementation issues particular to our system.

Realization of Local Access As mentioned in Section 3.3, to allow programmers to realize local access to a service selected in runtime, the system should support addressing of a platform that is on the same host as the service. Such information should be embedded as part of the metadata of the service. In the current implementation, directory services manage this information.

Data Exchange between Clones In our system, a clone is an agent that is created to execute a set of activities according to a clone rule. To realize such task assignment, it is necessary to send required data from one agent to another. Although data dependency between activities can be easily obtained from the BPEL process, each agent must know which agent is to execute the activity that requires the agent's data. The current implementation analyzes the BPEL process and behavior rules, and prepares in advance a table that associates an activity to the agent responsible for execution of the activity.

Minimization of Clones In Section 2.3 we mentioned reduction in migration cost by cloning. This can be achieved by analyzing the BPEL process and dispatching a clone with only the necessary data.

We are presently investigating more sophisticated implementation addressing these issues. This implementation is based on the Web Services technologies, intended to achieve the maximum platform-independence and interoperability.

5 Consideration

In this section we present several considerations about our system.

5.1 Effectiveness of Mobile Agents in Web Services Integration

Generally, the mobile agent paradigm enables various types of access to resources by combining (1) migration between hosts, (2) parallel execution by cloning, and (3) remote communication, as mentioned in Section 2.3. Thus, it is considered to be a general framework for modeling and implementation in distributed environments[4, 5]. It has been utilized for adaptation to various environments with resource constraints[9, 10], though we only mentioned the problem of wireless connections in this paper. The mobile agent paradigm will surely provide Web Services with a powerful way to adapt to various environmental conditions.

However, the effectiveness of a certain physical behavior, e.g. local access to an information service mentioned in Section 2.3, depends on the various parameters in the environment: exchanged data amount, network bandwidth and stability, computing resources, number of clients accessing the service simultaneously, and so on[11, 12]. Our future work is to conduct further experiments and evaluation in order to explore effectiveness of complex behaviors in various situations. This will help us provide both description guidelines and further high-level rule descriptions of complex behaviors.

5.2 Behavior Descriptions of Mobile Agents

In typical mobile agent systems, programmers describe both application logic and physical behaviors together in procedural languages such as Java[8]. This leads to very complex codes and makes it difficult to change only physical behaviors according to changes in the environments.

In our system, the BPEL process description as integration logic and the behavior rule descriptions for physical behaviors are separated, which is our original objective. It is easy to see what the actual tasks for integration are and to change only the physical behaviors in order to test which behaviors are the most suitable for the environment or to adapt to changes of the environment. The separation also enables step-by-step development styles: first develop and test only a BPEL process (or use an existing one), then add and test physical behaviors according to the environment. Moreover, in our behavior rule descriptions, it is not necessary to specify additional controls explicitly, such as remote communication between agents, as they are performed according to the control and data flow described in BPEL.

5.3 Related Work

Much research has been conducted on services integration (not limited to XML Web Services), some of which utilize mobile agents. However, none of them combine the mobile agent technology with standards for Web Services integration such as BPEL. From the point of view of mobility utilization, some approaches adopt simple migration models such as "always local access"[13], which may be inefficient in some situations. Many other approaches adopt migration models based on estimation and optimization of costs with dynamic resource managements[9, 10, 14] (including so-called "grid" approaches). Our work is intended to facilitate selection and combination of such various models of physical behaviors, for example, specifying target hosts of migration statically in some domains and utilizing provided allocation services with their own allocation policies in others. It is notable that in [10, 14], mobile agents provide "mobile services", rather than utilize other services as we have considered in this paper. It seems very interesting to utilize our system to implement such mobile services, as a BPEL process is both a service consumer and a service provider.

On the other hand, the need for separation of application logic and mobility has been investigated in several studies. In [15], mobility is described as declarative, event-driven policies. Although our current rules are very limited and don't handle event-based migration, the features of our system are that (1) it targets Web Services, composing a mobile agent with exchangeable services to carry and providing dedicated descriptions such as *local*, and (2) it also enables simple control of cloning.

6 Conclusion

Our study proposes a mobile agent system for Web Services integration, where physical behaviors of mobile agents, i.e. migration and cloning, are separated

from a BPEL integration process and described as simple rules. We are presently investigating the formal definition of behavior descriptions. This research is to ensure that our framework preserves the original semantic of BPEL and to facilitate tool support and validation. Starting with our work in this paper, our objective is to facilitate flexible behaviors of mobile agents in Web Service applications, to adapt to various environments such as pervasive networks.

References

1. Web services. http://www.w3.org/2002/ws/.
2. Satish Thatte et al. Business process execution language for web services, version 1.1. http://www-106.ibm.com/developerworks/webservices/library/ws-bpel/, 2003.
3. alphaworks : Web services tool kit for mobile devices. http://www.alphaworks.ibm.com/tech/wstkmd, 2002.
4. David Kotz and Robert S. Gray. Mobile agents and the future of the internet. *Operating Systems Review*, 33(3):7–13, 1999.
5. Deja Milojicic. Mobile agent applications. *IEEE Concurrency*, 7(3):7–13, 1999.
6. Web service description language (WSDL) 1.1. http://www.w3.org/TR/wsdl, 2001.
7. Bee-gent multi agent framework. http://www2.toshiba.co.jp/beegent/.
8. Mitsuru Oshima, Guenter Karjoth, and Kouichi Ono. Aglets specification. http://www.trl.ibm.com/aglets/spec11.htm, 1998.
9. Munehiro Fukuda, Yuichiro Tanaka, Naoya Suzuki, Lubomir F. Bic, and Shinya Kobayashi. A mobile-agent-based pc grid. In *Autonomic Computing Workshop 5th Annual International Workshop on Active Middleware Services (AMS'03)*, 2003.
10. C. Ragusa, A. Liotta, and G. Pavlou. Dynamic resource management for mobile services. In *The 5th International Workshop on Mobile Agents for Telecommunication Applications (MATA'03)*, 2003.
11. L. Ismail and D. Hagimont. A performance evaluation of the mobile agent paradigm. In *the Conference on Object-Oriented Programming, Systems, Languages, and Applications*, pages 306–313, 1999.
12. Robert S. Gray et al. Mobile-agent versus client/server performance: Scalability in an information-retrieval task. In *The 5th International Conference on Mobile Agents(MA2001)*, pages 229–243, 2001.
13. Amir Padovitz, Shonali Krishnaswamy, and Seng Wai Loke. Toward efficient and smart selection of web service. In *AAMAS'2003 Workshop on Web Services and Agent-based Engineering*, 2003.
14. Zakaria Maamar, Quan Z. Sheng, and Boualem Benatallah. On composite web services provisioning in an environment of fixed and mobile computing resources. *Information Technology and Management, , Kulwar Academic Publishers,*, 5(3), 2004.
15. R. Montanari and G. Tonti. A policy-based infrastructure for the dynamic control of agent mobility. In *3rd International Workshop on Policies for Distributed Systems and Networks(POLICY'02)*, pages 206–209, 2002.