An Agent-based Decentralised Process Management Framework for Web service Composition

Jun Yan, Phillip Pidgeon
School of Information Systems and Technology
University of Wollongong
Wollongong, NSW, Australia
61 2 4221 5411
{jyan,pep79}@uow.edu.au

Aneesh Krishna
School of Computer Science and Software Engineering
University of Wollongong
Wollongong, NSW, Australia
61 2 4221 4043
aneesh@uow.edu.au

Jianming Yong
Department of Information Systems
Faculty of Business
University of Southern Queensland
Toowoomba, QLD, Australia
61 7 46312448
yongj@usq.edu.au

ABSTRACT
Web service composition provision which requires efficient coordination of the execution of component services is a critical issue in service-oriented computing. Nowadays, BPEL4WS, the de facto industry standard for service compositions, is predominantly deployed in a way in which all interactions and intermediate data must go through one server. This centralised management results in problems such as poor performance, impaired reliability, limited scalability, and restricted flexibility. To address these problems, this research proposes an agent-based decentralised process management framework for Web service composition. This framework allows distributed BPEL engines, each of which is represented by a software agent, to manage the execution of relevant sub-processes, and to interact with one another directly to coordinate the execution of the whole process. Such a framework naturally reflects the distributed and dynamic features of the Web services environment and subsequently offers improved coordination support for service composition provision.

Categories and Subject Descriptors
D.2.11 [**Software Engineering**]: Software Architectures – *Domain-specific architecture, Patterns*

General Terms
Management, Design

Keywords
Agent, Process Management, Web Service Composition

1. INTRODUCTION
Over the past a few years, the paradigm of service-oriented computing (SOC) has gained intensive attention from both researchers and practitioners [9]. The most well-known integration platform for SOC is the Web Services technology which provides a framework to improve the cross-language and cross-platform interoperability for distributed computing and resource sharing over the Internet. Moreover, the Web services technology offers a cost-effective way for developing distributed applications such as business-to-business processing. By compositing distributed services into a network of services, i.e., a service composition, dynamic business processes and agile applications can be created with little effort.

A service composition is actually a process consisting of a set of independent component services which are executed in a partially-ordered manner in order to achieve the overall business objectives. In this sense, the execution of the component services needs to be well coordinated according to the pre-specified procedural rules. The de facto industry standard for Web service composition is Business Process Execution Language for Web Services (BPEL4WS, BPEL in short) [2]. Nowadays, BPEL is predominantly deployed in centralised servers, which implies that all interactions and intermediate data must go through one server. Therefore, the problems in relation to centralised management that have been encountered in the non-service environment are also observed here, including poor performance, impaired reliability, limited scalability, restricted flexibility, and so on [7] [11]. Moreover, due to the distributed and dynamic natures of the Web services environment, these problems are even aggravated, thus, becoming one of the major obstacles for wide deployment of the Web services technology, especially for applications where transfer of large amount of intermediate data is needed.

To address these problems, this research proposes an agent-based decentralised process management framework for Web service composition. This framework allows a set of distributed BPEL engines, each of which is represented by a software agent, to manage the execution of a Web service composition jointly. A BPEL process can be decomposed into a set of sub-processes which are then distributed to BPEL engines located on or close to the sites of component services. Distributed BPEL engines manage the execution of relevant sub-processes and interact with one another directly to coordinate the execution of the whole process.

The rest of the paper is organised as follows. The next section briefly introduces major related work. Section 3 presents the design of the agent-based decentralised framework. Based on this design, the system functionalities are described in Section 4.

INFOSCALE 2007, June 6-8, Suzhou, China

DOI 10.4108/infoscale.2007.904

Finally, Section 5 concludes this paper and outlines authors' future work.

2. RELATED WORK

In process management research, client-server basaed centralised process coordination contributes largely to problems like poor performance, impaired reliability, limited scalability, and restricted flexibility [11]. Therefore, peer-to-peer (p2p) based decentralised process management has been recognised as one of the most strategic future directions [7]. Fakas presents a conceptual p2p technology for dynamic workflow management, which is based on concepts such as a Web Workflow Peers Directory (WWPD) and Web Workflow Peer (WWP) [6]. Using this technology, peers are proposed to register with the system and offer their services and resources to other peers. PeCo [5] decentralises workflow management using collaborative technologies and concepts while providing a pluggable framework for integrating business process applications and human contributors. SwinDeW [11] contributes a decentralised framework and corresponding process coordination technologies for process management. However, these approaches may not be appropriate in the Web services environment, as the unique features of this environment, such as complete distribution and high autonomy, have raised new challenges to process management.

Software agents have been recognised as a promising technology for managing processes and orchestrating Web services. A well known project [1] develops and demonstrates the basic technology for agent-enabled process management to coordinate information delivery from multiple business systems. Buhler and Vidal develop a distributed multi-agent system enacting BPEL processes [3]. Guo et al. describe a multi-agent platform for enacting distributed business processes using LCC, a Light Coordination Calculus that can interpret BPEL specification and enable distributed enactment [8]. Purvis et al. propose a multi-agent based workflow management system with embedded Web services [10]. Nevertheless, research on agents in the context of process management is still immature. It has typically focused on multi-agent system architectures, agent interaction and coordination, and other agent technology centric issues that map to some basic requirements of distributed processes management. Most agent-based approaches offer in fact centralised process management where the agents are controlled by a central server or are deployed internally within the engine for specialised management tasks, which lead to inefficiency and vulnerability.

3. PROCESS MANAGEMENT APPROACH

3.1 Overall Framework

The key idea of the decentralised process management is to have a set of distributed nodes in the system, each of which has the processing capability. These nodes manage different resources and are able to interact with one another directly. The process coordination traditionally performed by a centralised server is thus collectively fulfilled by this set of nodes through collaboration. As the centralised coordination service is often associated with the performance bottleneck and the single point of failure, decentralised process management represents better performance, improved reliability, and enhanced scalability.

As shown in Figure 1, the proposed decentralised process management framework is based on a set of software agents which are distributed across multiple hosts. Characterised by the functions they perform, these agents can be categorised as initialising agents, monitoring agents, and peer agents. An initialising agent normally interacts with the end client and initialises the execution of the overall process. It obtains the original BPEL composition and the corresponding WSDL file, decomposes it into separate sub-process partitions, and distributes the sub-processes to relevant peer agents located on or close to the nodes that provide the component services for deployment and execution. An initialising agent is also responsible for delivering final results to the client. A monitoring agent monitors the execution of individual services and the whole composition by continuously scanning the state of service enactment and the actual values of non-functional parameters. A peer agent has the ability to manage the execution of an atomic service or even a sub-process in a Web service composition. At the same time, peer agents are able to exchange information in a meaningful way so that the execution of the composition can proceed. Please note that a physical agent can demonstrate one or more of these functions. For example, an initialising agent can perform the function of monitoring as well, thus becoming a monitoring agent. A peer agent can further decompose the sub-process when necessary, thus playing the role of the initialising agent.

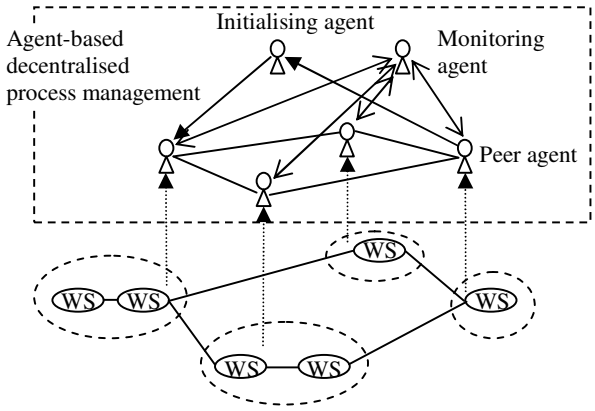

Figure 1. Overall process management framework

With support of this framework, the execution of a Web service composition works as follows. The initialising agent invokes the execution of the whole composition and receives the response when the execution of the whole composition concludes. Process coordination takes place in a distributed way through peer agent interaction to ensure the sub-processes are deployed, started, executed and managed. The monitoring agent subscribes the monitoring data from peer agents during the process of execution.

3.2 Agent Architecture

Peer agents are the most fundamental units in the system. As shown in Figure 2, a peer agent consists of a peer management module and a peer repository, a process management module and a process repository, and its own BPEL engine that hosts the sub-processes distributed to it.

The peer management module manages peer-related functions for the agents including peer registration and peer discovery, and maintains agent-related information in the peer repository. The

peer repository is a local file directory. A peer index is stored in the repository that containing a list of agents that share partnerlinks with the agent's processes, including agents that provide alternative component Web services used by the agent's processes. The process management module controls and monitors the BPEL engine of the agent. This involves managing the process definition files received from the initialising agent, saving the process definition files in the agent's process repository, configuring the BPEL engine with the process definition files, and starting, managing, and terminating the BPEL engine's operations. The process repository is a local file directory that houses process-related information used by the agent. The agent uses the process definition files stored in this repository, such as configuration, BPEL, and WSDL files, to deploy its processes autonomously. The BPEL engine located within the agent is capable of executing BPEL processes that are distributed to this agent. The agent has control over initialisation of the BPEL server, deployment of BPEL processes, establishment of the incoming and outgoing partnerlinks that connect the process to clients and/or other component Web services, and starting and stopping the engine's serving of the processes.

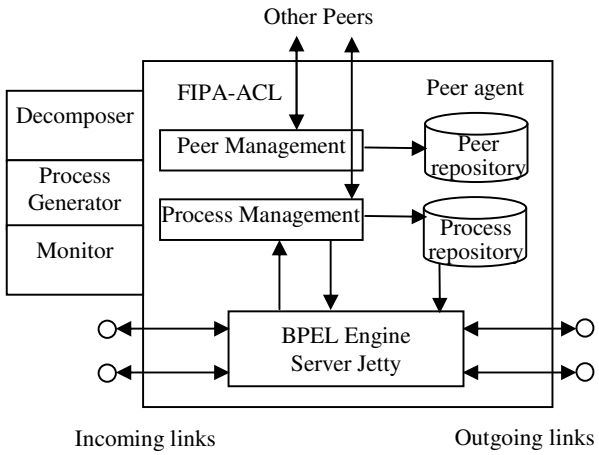

Figure 2. Agent architecture

As discussed in Section 3.1, a peer agent may exhibit additional functions such as initialising a Web service composition and monitoring the execution of the whole process. This is achieved by optional modules including a decomposer, a process generator, and a monitor. A decomposer module is able to create a set of partitioned processes by decomposing an original BPEL process. A process generator module is capable of re-constructing partitioned BPEL processes as a collection of files from the original process BPEL file, the process WSDL file, and the component Web services WSDL files. A monitor module is responsible for process execution monitoring.

4. SYSTEM FUNCTIONALITIES

4.1 Process Deployment

On each host that is involved in the decentralised process management, an agent environment is created, which joins with each other into a single multi-agent environment. An initialising agent is created dynamically on the host that is responsible for the execution of the whole BPEL process. In order to deploy the process, the initialising agent firstly obtains a configuration file from the service composition system. This configuration file defines the full detail of the BPEL process including the reference information for the process BPEL file, the process WSDL file, and the WSDL files for all the component Web services. The decomposer module of the initialising agent then uses the configuration information to decompose the process into sub-process partitions. Each partition is defined by its own process configuration file containing the sub-process BPEL file, a new sub-process WSDL file, and any component WSDL files used by the sub-process. Please note that the new WSDL file always starts from a *receive* element with the *createinstance* attribute being set to "yes". This is used at the later stage for process execution.

As shown in Figure 3, during decomposition, a package file is also created that collects all the sub-process configuration files together as a deployment package. This package file can then be deployed to setup the completely decentralised process execution. In addition, this file can be used in the future to re-deploy the whole process again if necessary without re-running the decomposition process. When the package file is deployed, agents are created on the host nodes. When an agent receives its process configuration file together with other relevant files (i.e., the sub-process BPEL file, the sub-process WSDL file, and component WSDL files used by this sub-process), it stores the files in its process repository. Then the configuration file can be deployed to the BPEL engine to load the BPEL process, set up the partnerlink endpoints, and start the engine ready for invocation. The configuration file, the BPEL file and the WSDL files are serialised into a string object, which is then sent as the content of an *INFORM* FIPA ACL message.

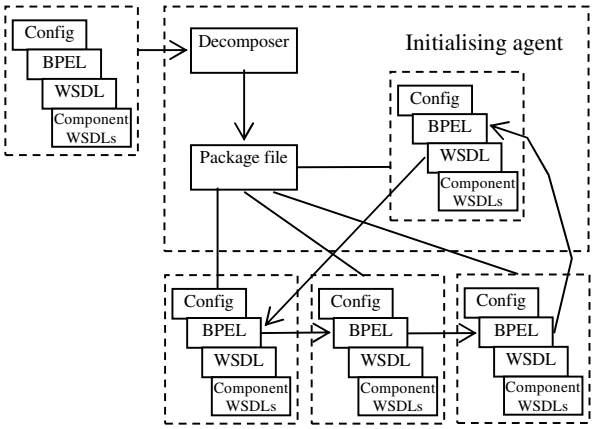

Figure 3. Deployment of BPEL process

Various strategies can be taken for process decomposition. The outcomes of decomposition of a BPEL process is logical partitions that consist of BPEL activities representing the component Web services invocation endpoints. This research uses Symphony [4], one of the well-known approaches developed at IBM' India research laboratory, to decompose a BPEL process. Using this approach, fixed activities of *receive* and *reply* must remain with the initialising agent, while *invoke* activities are fixed and co-located with their corresponding Web services. The remaining activities are considered portable and free to move to the partition which is most dependent on the activity for its control and data requirements.

4.2 Process Execution

Once an agent deploys a sub-process to its BPEL engine, the sub-process is started and ready for execution. On receiving a SOAP message on any of the sub-process's incoming partnerlinks with the *createinstance* being "yes", the execution of the sub-process is triggered. Execution occurs based on input variables supplied in the incoming SOAP message. The sub-process continues through the process activities sequentially or in parallel as pre-defined. Variables are progressively modified as each activity is completed. Component Web services can be invoked using the synchronous invoke activities, and asynchronous one-way invoke. The sub-process concludes at the end of it generally with a *reply* or an *invoke* activity passing an output variable response back to the caller, or onto the next sub-process.

As the Web services environment is highly dynamic, errors do occur during the process execution. For example, the scope of variables and process runtime conditions violate BPEL runtime constraints, or a service provider becomes unavailable due to lack of resources. Although some errors can be handled by specific BPEL fault handlers with the deployed BPEL code, most errors are too complex to be handled locally. In this system, errors that occur in a specific sub-process can be detected locally by the agent and more detailed error sequence information is provided back to the initialising agent or the monitoring agent for logging to assist in crafting a better fault-handler. Service provider unavailable is one of the most frequently encountered exceptions in service composition provision. This research use alternative fall-back service providers to deal with such exceptions by rerouting invocations to compatible replacement component Web service implementations. To do so, an agent stores the agents that provide alternative component Web services in its peer repository. Once a service provider unavailable exception is detected by the agent, it can be immediately reported back to the initialising agent. The shared partnerlink is supplied for re-deployment of the sub-process to a suitable alternative agent. The agents with the incoming links are then notified of the change of the partnerlink so that the invocation is re-attempted. Since alternative component Web services may not have the same interface, the agent may have an adapter BPEL process that converts the component Web service interface into a common interface understood by the sub-process. Figure 4 shows the agent interactions for alternative Web service invocation.

5. CONCLUSIONS AND FUTURE WORK

Service composition provision in the Web services environment refers to coordinated execution of a series of Web services to deliver new values. Adequate support is a key to achieving the full potential of the Web services technology. However, the current implementation of widely-accepted BPEL standard has demonstrated deficiencies in non-functional aspects. Due to the mismatch between the centralised management and the decentralised Web services environment, problems such as poor performance, impaired reliability, limited scalability, and restricted flexibility have been inherently encountered. To address these problems, this paper has innovatively proposed an agent-based decentralised process management framework for service composition provision. This approach utilises agent interaction to support distributed deployment of a service composition and decentralised coordination of the execution of component Web services. Based on this framework, the mechanisms of process deployment and process execution have been discussed.

In the future, more research work will be carried out based on the framework proposed in this paper. In particular, advanced process management operations will be investigated, including the mechanisms supporting dynamic SLA negotiation and service provider selection, process monitoring and profiling, and generic exception handling. A proof-of-concept prototype will be implemented, based on which a comparative performance study with variable amount of data being transferred will be conducted to determine efficiency improvement.

6. ACKNOWLEDGMENTS

This work is partly funded by the Australian Research Council Discovery Project Scheme under the grant DP0663841 and Faculty of Informatics Research Development Scheme, University of Wollongong.

7. REFERENCES

[1] ADEPT: Agent-Based Business Process Management project, http://www.ecs.soton.ac.uk/%7Enrj/adept/index.html

[2] Andrews, T., et al., Business process execution language for Web services version 1.1, 2003, ftp://www6.software.ibm.com/software/developer/library/ws-bpel.pdf

[3] Buhler, P. A., et al., Towards adaptive workflow enactment using multiagent systems, Information Technology and Management Journal, 6(1), 61-87, 2005

[4] Chafle, G. B., et al., Orchestrating composite Web services under data flow constraints, in Proc. of IEEE Int. Conf. on Web Services, 211-218, Orlando, Florida, USA, July 2005

[5] Coon, M. D., Peer-to-peer workflow management white paper, 2002, http://www.proteustechnologies.com/cmm/docs/p2p_workflow_whitepaper.doc

[6] Fakas, G., et al., A peer to peer (p2p) architecture for dynamic workflow management, Information and Software Technology, 46(6), 423-431, 2004

[7] Fischer, L. ed., Workflow Handbook 2002, Lighthouse Point, Fla.: Future Strategies, 2002

[8] Guo, L., et al., A generic multi-agent system platform for business workflows using Web services composition, in Proc. of 2005 IEEE/WIC/ACM Int. Conf. on Intelligent Agent Technology, 301-308, Los Alamitos, USA, 2005

[9] Papazoglou, M. P., et al., Service-oriented computing, Communications of the ACM, 46(10):25-28, Oct. 2003

[10] Purvis, M. A., et al., A multi-agent based workflow system embedded with Web services, in Proc. of the 2nd Int. Workshop on Collaboration Agent: Autonomous Agents for Collaborative Environments, Beijing, China, Sept. 2004

[11] Yan, J., et al., SwinDeW-a p2p-based decentralized workflow management system, IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans, 36(5), 922-935, 2006