Agent Services-Oriented Architectural Design of Open Complex Agent Systems

Longbing Cao, Chengqi Zhang, Jiarui Ni
Faculty of Information Technology, University of Technology Sydney, Australia
{lbcao, chengqi, jiarui}@it.uts.edu.au

Abstract

Architectural design is a critical phase in building agent-based systems. However, most of existing agent-oriented software engineering approaches deliver weak or incomplete supports for the architectural design of distributed and especially Internet-based agent systems. On the other hand, the emergence of service-oriented computing (SOC) brings in intrinsic mechanisms for complementing agent-based computing (ABC). In this paper, we investigate the dialogue between ABC and SOC and their integration in implementing architectural design. We synthesize them and develop the computational concept Agent Service, and build a new design approach called Agent Service-Oriented Architectural Design (ASOAD). The ASOAD expands the contents and ranges of agent and ABC, and synthesize the qualities of SOC such as interoperability and openness with the performances of ABC like flexibility and autonomy. It is suitable for designing distributed agent systems and agent service-based enterprise application integration.

1. Introduction

Agent-based architectural design (ABAD) plays significant role in designing agent-based systems especially enterprise agent systems. With the increase of system complexities of agent systems, ABAD faces new challenges [1]. In practice, ABAD approaches must adapt to the following increasingly emergent requirements for agent systems: (i) they are towards middle to large scale [4], (ii) they are increasingly Internet-based, and (iii) they are required to integrate enterprise applications. These kinds of agent systems are classified as *open complex agent systems* in terms of system complexity classification [1].

However, the architectural design supports available from most of existing agent-oriented software engineering approaches, for instance GAIA [8], are either at abstract level, or provide limited practical capabilities. They are not deployable for designing the above kinds of agent systems. Furthermore, from theoretical perspective, there are many critical issues emerging in utilizing existing ABAD to solve real problems. For instance, services are not explicitly separated and supported; supports for the architectural design of network-based agent systems, e.g. how to design system architecture for Internet-based applications, are weak; the requirements for enterprise applications integration, for instance, how to locate and transport distributed heterogeneous enterprise applications, brings new problems for "traditional" agent-based architecture.

The above issues propose new requests on agent-based computing (ABC) [6, 7]; ABC needs to fill in new supports. Fortunately, service-oriented computing (SOC) [2] has emerged as a powerful paradigm for designing distributed and especially Internet-based systems. The interaction between agent and service can enhance the function and range of agent; while the dialogue between ABC and SOC can integrate bilateral benefits from both of them, which are more suitable for building enterprise applications.

Motivated by the above ideas, this paper discusses the dialogue between agent and service, and between ABC and SOC; we define a new computational model called *agent service* which synthesizes the strengths of agent and service. A new architectural design approach called *agent service-oriented architectural design* by integrating ABC and SOC is proposed and outlined in this paper. Compared with other approaches, our approach is more effective in implementing the architectural design of distributed agent systems and especially enterprise application integration.

2. Dialogue between agents and services

In general, agent [7] and service [3] are two independent computational concepts. Agent is

proposed for modeling stakeholders with autonomous actions and cooperative abilities to archive their individual or global design objectives. ABC presents unprecedented computational intelligence of flexibility and autonomy in modeling complex software systems. However, the existing ABC techniques face many challenges from distributed and especially Internet computing, for instance, how to build a suitable architecture for enterprise application integration.

On the other hand, service is developed to wrap application logics, which exposes message-based interfaces suitable for being accessed across a network. SOC [2] is good at integrating and managing Internet-based enterprise interoperation, and wrapping legacy applications. However, SOC is weak in supporting the qualities of service such as autonomy and flexibility.

By nature, the capabilities of ABC and SOC are quite complementary. For the internal qualities of applications, it is agent and ABC that can do well; while service and SOC are more suitable for implementing the software-as-service initiatives and application-to-application architecture. The integration of agent and service, and of ABC and SOC can benefit each other in building a more powerful computational system [1]. This is the motivation of developing the integrated concept *agent service*, and *agent service-oriented architectural design* approach.

In practice, the integration of agent and service, and of ABC and SOC are feasible. On one side, we'll develop new concepts such as *agent service*, *service of agent*, and *service of service* [1] to build the internal linkages between agent and service at the low level. On the other hand, we'll build mechanisms for the integration of agent and service, and of ABC and SOC in architectural designs.

3. Agent service abstract model

Agent service abstract model [1] specifies fundamental architectures and properties of an agent service, intra-agent activities, and inter-agent communications. We define agent model and service model respectively.

3.1 Agent model

Agent model specifies agent architecture and agent classes. An *agent architecture* defines some specific commitments about the internal structure and intra-agent activities of an agent or a set of agents. There are mainly four categories of agent architectures; they are deductive agent architecture [6], reactive agent architecture [5], belief-desire-intention agent architecture [7], and hybrid architecture. For these agent architectures, we can build high-level agent architecture diagram for them.

Agent ::= *f*(*Name*, *Type*, *Locators*, *Owner*, *Roles*, *Address*, *Message*, *InputVariables*, *Preconditions*, *OutputVariables*, *Postconditions*, *Exception*)

3.2 Service model

Service model describes an agent's (or service's) activities required to realize the intra-agent or intra-service's roles and their properties, and inter-agent or inter-service communications in legacy and enterprise integration.

The agent or service activities embody basic attributes of agent services; these attributes represent intra-agent or intra-service's roles and properties. On the other hand, the definition of services managing inter-agent and inter-service communications is to design interaction and communication supports and patterns via attributes for cross-agent or cross-service interoperability. By nature, attributes describing intra- and inter-agent behaviors can be combined or to some degree shared with each other.

Similar to agent, the following shows a multiple-attribute tuple of service model.

Service ::= *f*(*Activity*, *Service Type*, *Locators*, *Owner*, *Roles*, *Type*, *Address*, *Message*, *InputVariables*, *Preconditions*, *OutputVariables*, *Postconditions*, *Exception*)

4. Agent service-based architecture for enterprise application integration

With agent service, service of agent, and service of service, and the integration of ABC and SOC, we build *agent service-oriented architectural design* (ASOAD) [1]. The benefit of the ASOAD is bilateral. For instance, the strengths of service and SOC make the agent service-based architecture more suitable for a distributed and especially Internet-based agent system; while an agent service-based distributed system can synthesize the qualities of SOC such as interoperability and openness, and the performances of ABC like flexibility and autonomy.

Some basic activities performed in ASOAD include: (i) Integrate enterprise applications, (ii) Design application architecture and agent service pattern, (iii) Design the user interfaces, (iv) Design system interfaces, (v) Design and integrate data

sources, (vi) Design and integrate the system controls. Specifically, it builds the following mechanisms for architectural design: (i) agent service abstract model, (ii) agent service design pattern, (iii) agent service management, (iv) agent service communication, (v) interaction pattern, (vi) information integration pattern, (vii) system architectural frameworks, and (viii) integration level and strategy.

It is potential to build agent service-based architecture for integrating enterprise applications compared with middleware-based integration and Web service-based integration. In *agent service-oriented integration*, applications are packed (or wrapped) as agents or services via a mediation layer or an integration layer that supports automated integration of data and business logics.

Agent service-oriented integration synthesizes the intrinsic strengths from ABC and SOC. It can deal with some limitations in Web service-based integration. It provides capabilities such as autonomy, flexibility and efficiency. In the remaining of this chapter, we shall focus on agent service-oriented integration.

In the implementation of agent service-oriented legacy integration, the following issues may need to be considered: integration level, integration type, interface design, message passing, event or transaction management, exception handling.

- Integration level: this involves at what level the two applications are integrated.
- Integration type: this is to determine that the inter-application communication is based on either wrapping or middlewaring integration.
- Interface design: agent services integrating legacy applications or components may generate agent service interfaces automatically, mirror the legacy applications, expose application APIs, or implement some predefined specification programmatically.
- Message passing: this defines the communication protocol and transport way between individual applications.
- Event and transaction management: this involves administration and management of agent services including event aspects such as dispatching, directory, transport, location, mobility, security, and transaction supports such as ACID (atomic, consistent, isolated, and durable) capabilities.
- Exception handling: this is to capture various exceptions emerged in the integration and execution of the integrated system.

5. Implementing agent service-based integration

Agent service-oriented integration can be implemented in terms of varying strategies and techniques. Two options are "multiagent + Web service" and "multiagent + service oriented computing".

5.1 Multiagent + Web services

This approach builds a dialogue between ABC and XML-driven Web service (from hereon referred to as *Web service*) [9]. The basic work in building a Web service-based infrastructure includes:

- Providing a service description that, at minimum, consists of a WSDL document;
- Being capable of transporting XML documents using SOAP over HTTP.

On the other hand, ABC technology plays significant roles in aspects beyond the above basic infrastructure. For instance, the following demonstrates some main functionality that ABC can serve.

- Agentized components: applications and application components in enterprise can be agentized on demand, for instance, agents for human-computer interaction, resource access dispatching, management of Web services, and enterprise business logic, etc. The problem here may include how to link and administer these agents in the Web services-centric environment.
- Management, dispatching, conversation, negotiation, mediation and discovery of Web services: for these issues, multiagents can play great roles with flexible, intelligent, automated and (pro-) active abilities. For instance, a gateway agent located at a remote data source listens to requests and extracts data from the source after receiving data request messages, it generates and dispatches an agent to deliver the extracted data to the requestor after completion. This can reduce the network payload and enhance the flexibility of remote data access.
- Agent-based services: services for middlewaring and business logics such as adapter, connector, matchmaker, mediator, broker, gateway and negotiator can be customized as agents. For instance, an agent-based coordinator fulfilling partial global planning has the capability to generate short-term plans to satisfy themselves, it may further alter local plans in order to better

coordinate its own activities. Agent-based services can enhance the automated and flexible decision-making of these services.

5.2 Multiagent + service oriented computing

This strategy is based on the combination of multiagent and service-oriented computing. Service-oriented computing is not necessary to be Web service-driven or -centric. In this approach, multiagents act as the main building blocks of a system. The system following this approach is called as "multiagent-driven service system". The fundamental rules for developing multiagent-driven service systems are as follows.

- The system analysis is performed in terms of agent-oriented early and late requirements analyses, in particular the fore-detailed organization oriented analysis;
- The system architectural design is undertaken in light of agent service-oriented design, especially the fore-mentioned agent service-oriented architecture and application integration;
- The system detailed design is achieved via agents and services, and agent service-oriented interactions;
- Agents consist of the main building blocks in the system; while services play significant roles such as services of agents, services of services, inter-application communications and integration, etc.;

By contrast, the content of services here is a little different from the services in Web services. This can be embodied in terms of two aspects: (i) the conceptual scope is broader than that of Web services, for instance, services could be of agents and services; (ii) agent service is a unified concept and a computational entity, agents encompass explicit service attributes and features, services are agentized through supporting autonomy, flexibility and proactive capability.

6. Conclusions and future work

Agent-based computing and service-oriented computing are generally viewed as two independent computational paradigms. In practice, ABC and SOC are highly complementary; the dialogue between them can benefit existing ABC in designing distributed especially Internet-based agent systems. This paper discusses this dialogue and the integration of agent and service, and proposes a new agent-based architectural design approach, referred to as *agent service-oriented architectural design*.

The ASOAD expands the concept and its function of agent and ABC. This makes it more powerful for designing distributed especially Internet-based automated systems and enterprise application integration. We deploy it in developing a real agent service-based system – F-Trade [10]. It shows that an agent service-based distributed system can combine the qualities of SOC such as interoperability and openness with the performances of ABC like flexibility and autonomy.

More work is on building practical mechanisms for agent service management and communications in Internet environment.

Acknowledgments

This work gets funding support from Capital Market CRC Australia for the Data Mining Program, UTS Research Fellowship Funding, and data service supports from AC3 Australia.

References

[1] L.B. Cao. Organization and service oriented analysis and design – engineering open complex agent systems. *PhD dissertation*, University of Technology Sydney, Australia, 2004.

[2] T. Erl: Service-Oriented Architecture: A Field Guide to Integrating XML and Web Services. Pearson Education, 2004

[3] FIPA SC00001L. FIPA Abstract Architecture specification, 2002

[4] A. Garcia et al. (eds) Software engineering for large-scale multi-agent systems. Springer, 2003.

[5] P. Maes (Eds): Designing autonomous agents. The MIT Press, 1990

[6] G. Weiss (Eds): Multiagent systems: a modern approach to distributed artificial intelligence, The MIT Press, 1999

[7] M. Wooldridge: An Introduction to Multiagent Systems. John Wiley & Sons, Ltd, 2002

[8] F. Zambonelli, N.R. Jennings and M. Wooldridge. "Developing multiagent systems: the GAIA Methodology", *ACM Trans on Software Engineering and Methodology*, 12(3):317-370, 2003

[9] www.xml.com

[10] L.B. Cao, J.Q. Wang, L. Lin, and C.Q. zhang. Agent Services-Based Infrastructure for Online Assessment of Trading Strategies. *Proceedings of the 2004 IEEE/WIC/ACM International Conference on Intelligent Agent Technology*, IEEE Computer Society Press, 345-349.