An Agent-Based Autonomous Management Approach to Dynamic Services

Fu Hou, Xinjun Mao, Junwen Yin, Wei Wu

An Agent-Based Autonomous Management Approach to Dynamic Services

Fu Hou, Xinjun Mao, Junwen Yin, and Wei Wu

College of Computer, National University of Defense Technology, NUDT
Changsha, Hunan, China
{fu.houmail,mao.xinjun,wuei08}@gmail.com
jwyin_cn@163.com

Abstract. The services over Internet are often dynamic, evolving and growing due to the changes of the requirements and operating contexts. Therefore, it is necessary to effectively manage the dynamics of services in order to support flexible, efficient, and transparent services access for applications at run-time. This paper proposes an agent-based technical framework of managing dynamics of services, in which agent as the bridge between the applications and the target services is responsible for encapsulating the services as its behaviours, monitoring the dynamic of services, and providing proper services for applications. The situation model of services and implementation architecture of service manager are designed to specify the expected dynamic aspects of services and support the monitoring and access on services. Based on Jade and Tomcat, we have developed a prototype platform called *ServiceAutoManager* to implement the above technologies and studied a case to show the effectiveness of our proposed approach.

Keywords: *dynamic services;software agent;autonomous management; service situation*

1 Introduction

Service-Oriented Computing (SOC) paradigm has gained popularity today for supporting the development of rapid, low-cost, interoperable, evolvable, and massively distributed applications. It is promised to easily assemble application components into a loosely coupled network of services that span organizations and computing platforms etc. [1]. Web services are currently the most promising SOC based technology. Several technologies and standards have been proposed to manage the web services, e.g. WSDL to describe the services, SOAP to access service etc., for service-based applications development and invocation. Based on the technologies and standards, service provider registers web service in registry with some basic information e.g. operations, binding port, etc., if application intends to invoke some service, it will look up the registry to get the required service information, and then access the service in term of SOAP.

However, the services over Internet are dynamic, evolving and growing due to the changes of the requirements and operating contexts. One typical case is the

evolving changes of web service's QoS. Doubtlessly, to determine the service to be accessed at design-time or only by considering the functionality information of service seems unfeasible, because the expected service may be unavailable or more suitable services for applications may arise due to the dynamic of services at runtime. This leads to the requirements of managing the dynamics of services in an autonomous and transparent way. Obviously, the autonomous management of services requires to perceive, organize, acquire the dynamic of services, and to decide which service should be accessed and invoked to satisfy the application requirements according to the dynamic information. There are several ways to tackle the issues: one is to re-construct web services and provide them with the capabilities of managing dynamics and providing autonomous decision. The other is to enrich the applications with the capabilities of monitoring the dynamics of services and of autonomy on the access of the expected services. However, both of these methods are complex because they need to modify the existing services or applications.

Software agent technology gives us inspirations about how to manage and organize the dynamic services. First, agent is context-aware, which means it can sense the situated environment. If we provide the monitoring capabilities for agents, they can perceive the dynamic information of services. Second, agent is autonomous on behaviours, which means agent can decide which behaviours should be performed according to its internal state and external requests. Therefore, we can access services by agent in an autonomous and flexible way. Third, agent is expected to be social, which means multiple agent can interact and cooperate with each other in order to achieve global objectives. Therefore, the complex management or organization can be achieved in term of the interaction of agents. Moreover, software agent technology enable legacy systems to be incorporated with new functions and capabilities.

The remainder of this paper is structured as follows. Section 2 discusses the related works about service management methods and the utilization of agent technology into service-oriented computing. Section 3 overviews the technical framework to manage dynamics of services. Section 4 details the management model and situation model of dynamic services, and presents the implementation architecture and the autonomous management process of dynamic services. Section 5 introduces a prototype platform - *ServiceAutoManager* that implements the above technologies and supports the autonomous management of dynamic services, and a web service application of travel searching is discussed to show the effectiveness of our proposed approach. Conclusions and future works are made in section 6.

2 Related Work

As more and more evolving and dynamic services are deployed on Internet, how to manage the dynamic of services has obtained great attentions from both industry and academic fields. Several attempts have been made in the past years, most of them focus on the service operation management. A typical work is to

establish the registry management pattern based on SOA [2], e.g. ESB, and there are also some researchers propose web service distributed management which essentially defines a protocol for interoperability of management information and capabilities in a distributed environment via Web services [3]. Recently seeding autonomic capabilities for service level management is an evolutionary service level management approach where autonomic computing capabilities anticipate IT system requirements and resolve problems with minimal human intervention [1]. E.g. Yu Cheng et al. propose a prototype architecture and discuss related implementation issues for an autonomic service management framework [4]. Bhakti et al. present the idea of adapting the autonomic computing paradigm into SOA to dynamically (re)organizing its topologies of interactions between the services with little human intervention [5].

Software agent techniques have been extensively exploited to solve issues occurring in several areas. They use software agent to implement the adaptive process, service composition, service selection and even a broker. E.g. Yu Fei Ma et al. present a lightweight autonomous agent fabric for Web service [6]. Lopes and Botelho present a framework to enable the execution of semantic Web services using a context-aware broker agent [7]. Sreenath and Singh proposes a new agentbased approach in which agents cooperate to evaluate service providers [8]. Chainbi et al present an agent-based framework for autonomic web services which is based on two agent-based systems collaborating to enrich web services and registries with self-* capabilities [9].

3 Agent-based Technical Framework of Managing Dynamics of Services

In order to manage the dynamic of services and support applications to flexibly access services, we propose an agent-base technical framework (see Fig. 1). In this framework, services are to be managed by a number of managers that are actually software agents. Each service manager can manage one or more services, and each service corresponds to at most one manager. Developer should explicitly describe the management relationship between the service and service manager at deploy-time. Such relationship can be dynamically adjusted at runtime in term of self-organizing of multiple service managers in order to achieve the re-organization and optimization of the management relationships. When the relationship is established, the infrastructure of services is responsible for monitoring the dynamics of the managed services and sending the dynamic information to the service managers. The managed services together with their dynamic information consist of the internal resources and states of service manager. The service managers should register at the service manager center. The registered information includes the agent ID and address of manager, the managed services, etc. Therefore, service manager center knows what the functions services provide, and who are responsible for managing these services. Applications that intend to access some service should first look up service manager center to obtain the information about which agent is responsible for managing

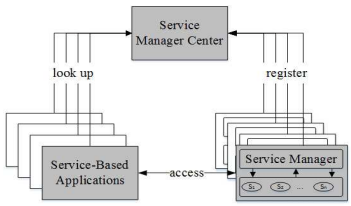

Fig. 1. Agent-based Technical Framework of Managing Dynamics of Services

the service that satisfies the expected functions. Then applications send request that describes the expected functions, constraints and corresponding parameters to the service manager. According to the requests and the dynamics of the managed services, service manager decides which service is more suitable for providing the function, and then access the service and return the service results.

4 Approaches to Autonomous Management of Services

4.1 *Model of Services Management*

Fig. 2 depicts the abstract model of service management respecting the above technical framework, in which service managers are designed to manage the dynamics of services, decide and invoke the services to satisfy the applications requests. The whole management consists of three aspects: service enrolment, service monitoring and service invocation.

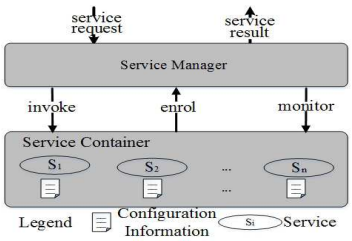

Fig. 2. Model of Services Management

a) *Service Enrolment.* The purpose of service enrolment is to establish the management relationship between service manager and the managed services. Before service manager begins to manage services, developers should deploy the service into service container and explicitly define service's configuration file. There are two types information in configuration file: service access information and the monitored dynamic aspects of the service. When the

services are deployed in the container, container is responsible for enrolling the configuration information into the service manager.

b) *Service Monitoring.* According to the monitored aspects of dynamics, service manager is responsible for monitoring the dynamics of services and maintaining the dynamic information. In this paper, we provide a situation model of service that covers several aspects of dynamics, including satiated context, quality status, and business capability. When service manager gets the configuration information of service, it will monitor the services and obtain the dynamic information of services based on the infrastructure.

c) *Services Invocation.* Once the enrolment of management relationship and service monitoring are established, service manager can autonomously manage and invoke the services. When receiving service requests from applications, service manager will find and access the proper services that satisfy the request.

4.2 *Implementation Architecture of Service Manager*

In the above abstract model, service manager is the core to achieve the objective of service's autonomous management. Based on the reactive agent architecture we propose the implementation architecture of service manager (see Fig. 3). In Fig. 3 we can see that service manager is comprised of six basic modules:

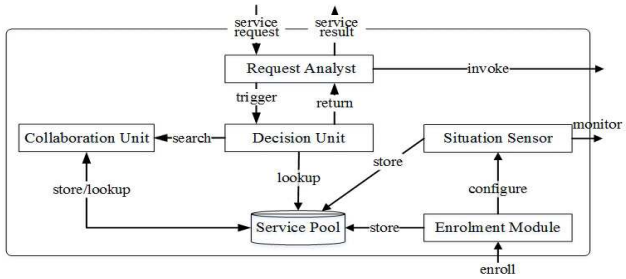

Fig. 3. Implementation Architecture of Service manager

a) *Enrolment Module.* Once the enrolment of management relationship and service monitoring are established, service manager can autonomously manage and invoke the services. When receiving service requests from applications, service manager will find and access the proper services that satisfy the request.

b) *Service Pool.* Service pool is used to store the managed services and their enrolment and dynamic information. The monitored information of services are sent and processed by the service pool. Therefore it saves up the latest information about the situation of the managed services. The decision center and collaboration unit can look up the service pool to get the dynamic information of the services.

c) *Collaboration Unit.* Collaboration unit is to perform the interactions among service managers in order to achieve the collaborations. The purpose of collaboration may be querying services managed by other managers or self-organizing among managers to reorganize the managed services in different managers.
d) *Decision Unit.* Decision unit is the component to decide which service managed by the manager is more suitable for the service request. The decision process is based on the composite considerations on the service request and the service dynamics and triggered by the service request.
e) *Request Analyst.* Request analyst is responsible for analysing the service request from application, interacting with the decision unit to obtain the expected service, and returning the service results to the applications.
f) *Situation Sensor.* Situation sensor is in charge of monitoring the managed services and obtain their dynamic information. It is also responsible for interacting with the service pool to store the perceived information.

4.3 *Autonomous Management Process of Services*

The autonomous management process for services consists of several phases in which different components and agents are involved and cooperated (see Fig. 4).

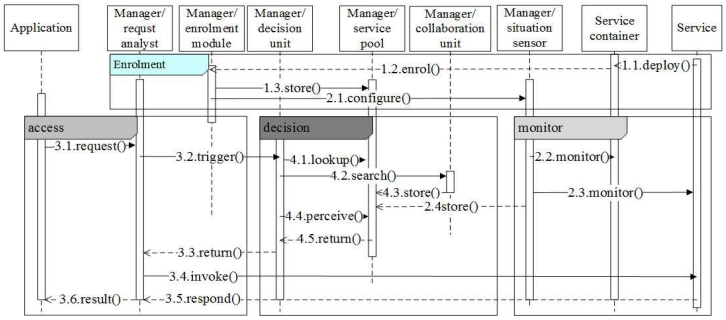

Fig. 4. Autonomous Management Process Sequence Diagram of Services

a) *Service Enrolment Phase.* First services are to be deployed into service container that supports the running of services and maintains the meta information of services. Then service container sends enrolment request to manager in order to establish the management relationship between the service and the manager.
b) *Service Monitoring Phase.* First services are to be deployed into service container that supports the running of services and maintains the meta information of services. Then service container sends enrolment request to manager in order to establish the management relationship between the service and the manager.

c) *Service Access Phase.* Request from application will trigger the process of accessing service. Typically, such request will be pre-processed by the request analyst, and request analyst will interact with the decision unit to find the proper service. Then service analyst will invoke the service and return result to application.
d) *Autonomous Decision Phase.* Once an application request trigger the decision unit, the autonomous decision process will begin. First decision unit will look up the service pool and perceive the service situation if there exist the suitable service that satisfies the request, decision unit will return the service. If there is no proper service, the decision unit will collaborate with other managers to obtain the proper service.

4.4 *Service Situation Model*

With the proliferation of services as a business solution to enterprise application integration, the QoS is becoming increasingly important to service providers [10]. However, various applications with different QoS requirements will compete for network and system resources e.g. bandwidth and processing time, so QoS is not enough to describe the dynamics of service. In this paper we propose a situation model of service to describe and define the service dynamic aspects covering situated context, quality status and business capabilities (see Fig. 5).

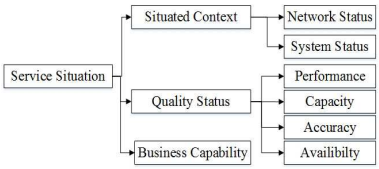

Fig. 5. Situation Model of Dynamic Service

Situated context describes the utilization status of network and system resource (e.g. bandwidth etc.). Quality status describes the service qualities aspects status information, including performance (e.g. throughput, response time etc.), capacity (e.g. the limit of the number of simultaneous requests), accuracy (e.g. the number of errors), and availability. Business capability describes business capability status information of service. For example, the current positioning precision of GPS's position service.

5 ServiceAutoManager Platform and Case Study

In order to support the autonomous management of services and corresponding development tasks, we have developed a prototype platform called *ServiceAutoManager*. *ServiceAutoManager* integrates Jade and Tomcat together, and provides the autonomous management functions for services. The implementation framework of *ServiceAutoManager* consists of four layers (see Fig. 6).

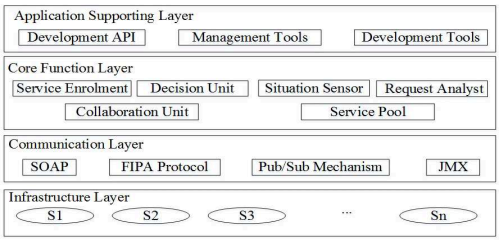

Fig. 6. Implementation Framework of SAutManager Platform

a) *Infrastructure Layer.* Infrastructure layer implements service container. It is in charge of service deployment and running. We adopt Apache Tomcat as the service running environment and JMX technique as the service monitoring engine.
b) *Communication Layer.* This layer is the basis of *ServiceAutoManager* and provides the FIPA protocol supporting service managers' collaboration. We implement this layer by means of expending Jade platform [11]. This layer also provides the basic mechanisms like JMX, SOAP and Sub/Pub to implement service enrolment and monitoring.
c) *Core Function Layer.* This layer is the core function layer of *ServiceAutoManager*. It is the concrete function implementation of service manager, including components like service enrolment, decision unit, request analyst, collaboration unit, service pool and service sensor.
d) *Application Supporting Layer.* In this layer we provide several supports to aid the development and management of applications, such as development API, management tools and development tools. Using the API and tools either the developers or managers can develop and manage the services quickly and efficiently, and can implement the transparent access.

In the following we give a case study to illustrate the above technologies and platform. The case is about travel searching problem, namely from **A** position to **B** position to find a path which satisfies user requirement with cellphone map searching application. In this paper we assume that the searching process needs three types of services: position service, path information service and taxi service. To reach **B** with the shortest time we need a position service to locate the position of **B**, and based on the path information service we will get some paths, and depending on the path status we need select one path and take taxi to **B**. This process is an artificial process, we hope this process can be implemented on the cellphone application, we assume that at the time there are three position services, four path information services and six taxi services, where the positioning precision P_1 of position service PS_1 satisfies $P_1 \geq 30\mathrm{m}$, position service PS_2 satisfies $P_2 \geq 10\mathrm{m}$ and position service PS_3 satisfies $P_3 \geq 25\mathrm{m}$; the current ratio of traffic jam respectively is: $R_{PIS1} = 6\%$, $R_{PIS2} = 3\%$, $R_{PIS3} = 8\%$, $R_{PIS4} = 7\%$, and the ratio of traffic jam is dynamic ; the response time of taxi respectively is: $T_1 = 2\mathrm{min}$, $T_2 = 4\mathrm{min}$, $T_3 = 6\mathrm{min}$, $T_4 = 5\mathrm{min}$,

T_5 = 3min, T_6 = 10min, the services and their managers relationship diagram sees Fig. 7.

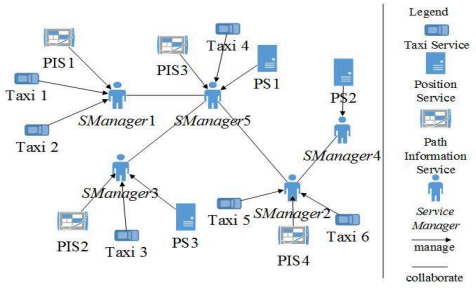

Fig. 7. Diagram of Travel Searching Case Services and Service Managers Relationship

We design the travel searching application under above hypothesis, the application accesses services process is: **Firstly**, find a Service Manager namely *SManager5* from service manager centre. **Secondly**, define the service requirements of application, the requirements just are the service situation constraints: 1).The requirement of position service: the positioning precision P of position service satisfies $P \leq$ 20m; 2).The requirement of path information service: the current ratio R of traffic jam satisfies $R \leq 5\%$; 3).The requirement of taxi service: the response time T satisfies $T <$ 3min. **Thirdly**, access the services based on requirements, in this process service manager will autonomously and transparently return the service results, in this case the results are **PositionService2**, **PathInformationService4** and **TaxiService1**.

6 Conclusions and Future Work

This paper researches the autonomous management technologies of services due to the dynamic of services and the requirements of flexible and transparent service access. Different from related works, we adopt agent as technical solution and design service managers to perceive, organize and maintain the dynamics of services, and to further decide which service should be invoked according to the service request and situation. Service managers are actually agents that encapsulate services, manage their dynamic and provide autonomous service access. Developer only needs to explicitly claim the management relationship between service managers and the managed services. Therefore, our approach enables us to manage services promote the reuse and re-organization of legacy service systems. In order to support service's autonomous management based on the agent-based technical framework, we have presented situation model of service and implementation architecture of service manager that answer what the dynamic information should be managed and how to manage the dynamic service in an autonomous way respectively. We have developed a prototype platform

called *ServiceAutoManager* based on Jade and Tomcat, and studied a travel searching case to illustrate the proposed approach.

The future works include: (1) the self-organization and collaboration among service managers to re-organize the managed services; (2) self-adaptation technologies for accessing web services in term of the changes of both requests, situated contexts and the dynamic of service itself; (3) more experiments to analyse the richness and weakness of our proposed approach.

Acknowledgments. The research leading to these results has received funding from National Nature and Science Foundation of China under Granted Nos.61379051 and 61133001, Program for New Century Excellent Talents in University under Granted No. NCET-10-0898 and Open Fund State Key Laboratory of Software Development Environment under Granted No. SKLSDE-2012KF-0X.

References

[1] M. P. Papazoglou, P. Traverso, S. Dustdar, F. Leymann.: Service-oriented computing: a research roadmap, International Journal of Cooperative Information Systems, vol. 17, pp. 223–255, (2008)

[2] M.P. Papazoglou , W.J. van den Heuvel.: Web Services Management: A Survey, IEEE Internet Computing, vol. 9, pp.58–64, (2005)

[3] H. Kreger, et. al.: Management Using Web Services: A Proposed Architecture and Roadmap, IBM, HP and Computer Associates, available at:`www-128.ibm.com/developerworks/library/specification/ws-mroadmap`, (2005)

[4] Y. Cheng, A. Leon-Garcia, I. Foster.: Toward an autonomic service management framework: A holistic vision of SOA, AON, and autonomic computing, Communications Magazine, IEEE, vol. 46, pp. 138–146, (2008)

[5] M. A. C. Bhakti, A. B. Abdullah, L. T. Jung.: Autonomic, self-organizing service-Oriented Architecture in service ecosystem, in Digital Ecosystems and Technologies (DEST), the 4th IEEE International Conference on, pp. 153–158, (2010)

[6] Y. F. Ma, H.X. Li, P. Sun.: A lightweight agent fabric for service autonomy, in: Proc. of the AAMAS 2007 Workshop Service-Oriented Computing: Agents, Semantics and Engineering, pp.63-77, (2007)

[7] A.L. Lopes, L.M. Botelho.: Executing semantic Web services with a context-aware service execution agent, in: Proc. of the AAMAS 2007 Workshop Service-Oriented Computing: Agents, Semantics and Engineering, pp.1-15, (2007)

[8] W.H. Oyenan, S.A. DeLoach.: Design and evaluation of a multi-agent autonomic information system, International Conference on Intelligent Agent Technology, pp.182–188, (2007)

[9] Walid Chainbi, Haithem Mezni and Khaled Ghedira.: AFAWS: An Agent based Framework for Autonomic Web Services, Multi agent and Grid Systems, vol. 8, pp. 45–68, (2012)

[10] KangChan Lee, JongHong Jeon, WonSeon Lee, Seong-Ho Jeoong, Sang-Won Park, QoS for Web Services: Requirements and Possible Approaches, `http://www.w3c.or.kr/kr-office/TR/2003/ws-qos/`, (2004)

[11] F. L. Bellifemine, G. Caire, D. Greenwood.: Developing multi-agent systems with JADE, vol. 7: Wiley. com, (2007)