SOARWare: A Service Oriented Software Production and Running Environment

Hailong Sun, Xudong Liu, Xiang Li, Jin Zeng, Zicheng Huang

School of Computer Science and Engineering, Beihang University
37 XueYuan Rd., Haidian District, Beijing, China
{sunhl,liuxd,lixiang,zengjin,huangzc}@act.buaa.edu.cn

***Abstract*—Service oriented computing provides a novel approach to building new software applications through the reuse of existing services. In this paper, we present SOARWare, a suite of middleware and tools, for software production and running based on Web services technologies. Basically, SOARWare consists of three major components including SOARBase, Service Oriented Software Production Line and Service Running Bus. Additionally, SOARWare provides a web-based platform for various users to access to the system functionality in a SaaS manner. We depict the design principle, system architecture and major functions of SOARWare.**

Keywords-SOA; service oriented computing; software development; Web service;

I. INTRODUCTION

Service oriented computing[1] is known as a new computing paradigm that utilizes services as fundamental elements for developing applications. The underlying technologies usually include SOA (Service Oriented Architecture) and Web services.

A traditional way of software development is based on the so-called "divide-and-conquer" manner, in which software is divided into several modules and modules are implemented separately. Instead, service oriented computing provides a novel approach to building new software applications through the reuse of existing services. A service is a self-describing entity that can be discovered and accessed through standard-based protocols. Thus software development with service oriented computing is more about integrating existing services to meet application requirements. Intuitively, software productivity can be much improved with service oriented computing technologies. One of the challenges faced by service oriented computing is how to make a system adaptable to rapid changing user requirements. One way to address this issue is late-binding, which means that only abstract services are specified in the process of modelling and development while concrete services are chosen in runtime. To be more specific, service oriented software is finally implemented and instantiated in running. In other words, the development and running of service oriented software can not be separated clearly like traditional software. Therefore, the development and running of service oriented software should be considered simultaneously in practise.

On the basis of the above understanding, we aim at designing an environment which can support the full lifecyle of software development through service oriented computing. To achieve this goal, we divide this task into four subtasks as follows.

- Service resource management. As more and more services are developed and published over both Internet and Intranet, there must be some mechanism to manage this resource so as to provide support for efficient service discovery and consuming. Service resource management mainly include service crawling, organization, mining and searching.
- Design and development. Business process is the core concept in service oriented software development. The business process centred development involved a series of activities including modelling, service orchestration, verification and testing, which requires various people to participate in. Thus how to provide support for the efficient corporation is an important issue.
- Runtime management. After the application is developed, it will be deployed to runtime environment. The service oriented application is constructed through composing various services. The service container and composite service engine are designed to support the running of atomic and composite services respectively. Additionally, other issues including the management of service containers and composite service engines, message routing and adaptive service selective are all the necessary functionality of runtime management.
- Unified access platform. Similar to clouding computing, it is very important to provide a unified access platform for service consumers, service providers and service developers.

There has been plenty of research work in service oriented computing, including the aforementioned issues. For example, service discovery is studied using semantic technology[2], peer-to-peer approach[3] and data mining technologies[4]. And the approaches to service composition primarily fall into three categories: semi-automatic[5], AI planning[6] and semantic composition[7]. However, few of these works incorporate all related service oriented computing technologies together to build a complete system solution. In this paper, we present SOARWare, a suite of middleware and tools for software production and running based on Web services technologies. Basically, SOARWare consists of three major components including SOARBase, Service Oriented Software Production Line and Service Running Bus. Additionally, SOARWare provides a web-based platform for

978-0-7695-4012-2/10 $26.00 © 2010 IEEE
DOI 10.1109/APWeb.2010.48

various users to access to the system functionality in a SaaS (Software as a Service) manner.

The rest of this paper is organized as follows. In Section II, we describe the design considerations of SOARWare and its system architecture; Finally, Section III concludes this work.

II. Design of SOARWare

As mentioned in Section I, there are four major issues that need to be addressed in the design of SOARWare. Basing on that, we hope that SOARWare should have the following characteristics.

Service manageability. Services are fundamental resources for building service oriented applications. On the one hand, although Web service technology is based on a stack of standard protocols, there are still no standard ways to describe the function of a Web service precisely. Even for the same function, its description provided by different service providers can be quite different. Moreover, the number of services available is continuously increasing, which makes it a challenging issue to locate a needed service efficiently. On the other hand, the non-functional properties like response time, availability, reliability and reputation are critical to the ultimate software applications. And the management of non-functional service properties are non-trivial for service oriented computing. Therefore, we define service manageability as the ability of the effective management of the functional and non-functional properties of service resources.

Service composability. In essence, a service oriented application is the appropriate composition of existing services. Service composability denotes the ability to effectively compose services so as to meet complicated application requirements.

Evolvability. This characteristic captures the system ability to adapt to dynamic application requirements and runtime environments. In other word, a service oriented system should have the ability to provide on-demand service based on the dynamic service environments. The evolvability can be achieved through dynamic QoS-aware service selection and evolution of the structures of a business process itself.

Usability. There are several kinds of entities including service providers, application developers and application consumers in service oriented computing. Usability is considered in our design so as to provide a unified platform for different entities to access to the tools and middleware support of SOARWare.

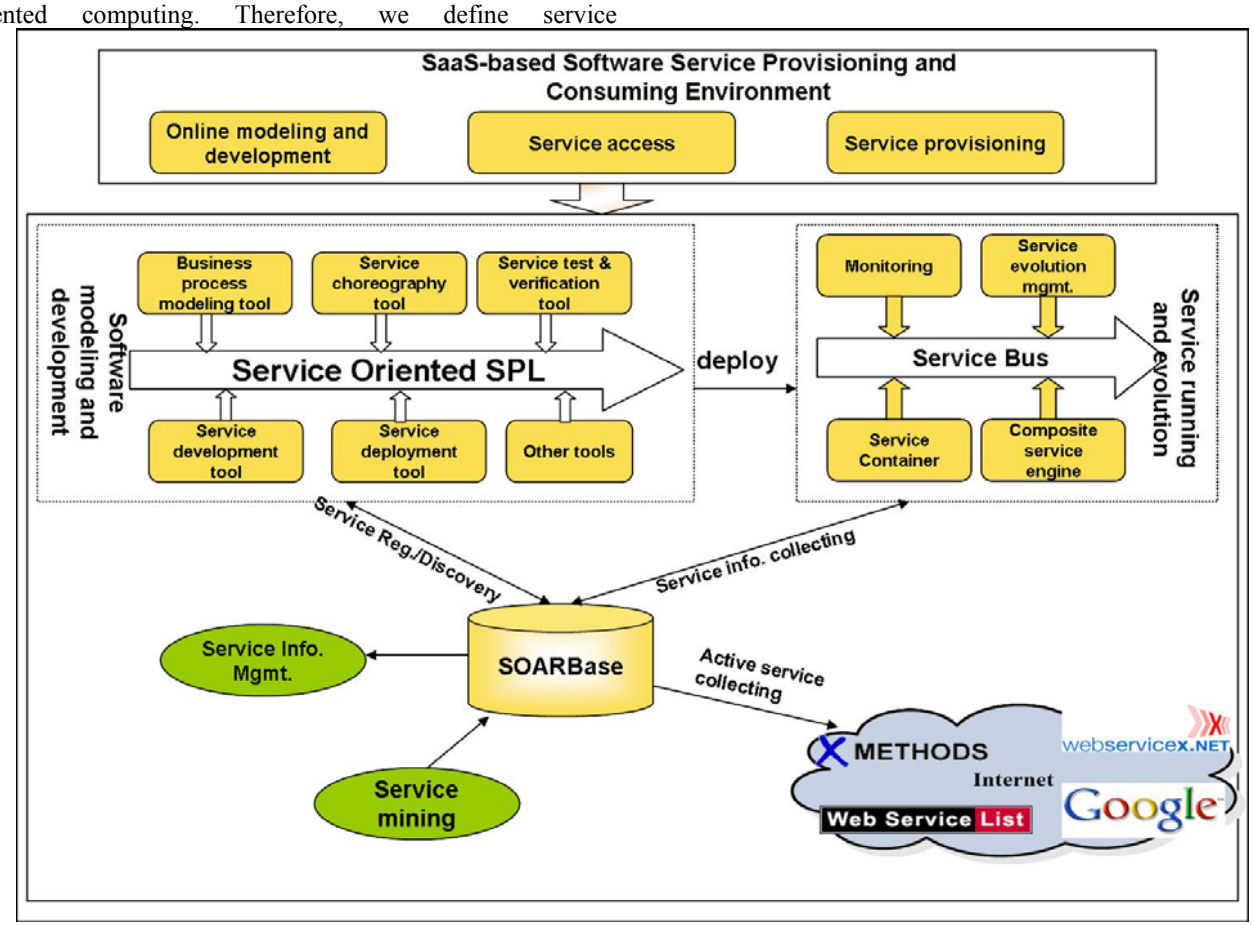

Fig. 1 System architecture of SOARWare- A Service Oriented Software Production and Running Environment

Fig. 1 depicts the system architecture of SOARWare, which is composed of SOARBase, Service Oriented SPL (Software Production Line), SOARBus and SaaS-based Software Service Provisioning and Consuming Environment. These four components address the four issues we mentioned respectively.

First, SOARBase is a fundamental component to address the service manageability issue. The role of SOARBase in SOARWare is similar to the role of DBMS in traditional software systems. Not only service providers can publish their services to SOARBase, but also SOARBase itself can search existing online service registries or make use of web search engine to actively collect Web services over the Internet. Furthermore, SOARBase is responsible for managing the collected service information using data mining technologies. In all, the objective of SOARBase is to provide support for discovering needed services as efficiently as possible.

Second, since SOARBase provides the source of services for producing software, there still needs an effective mechanism to manage the software production process. This mechanism is provided by SOSPL (Service Oriented Software Production Line). The business process is the kernel concept in software production with SOSPL. Software production is composed of a series activities centring on business process, including business process modelling, service orchestration, verification, testing and deployment. On the one hand, SOSPL provides tools to support various development activities; on the other hand, SOSPL must support the automatic control of the development process.

Third, after service oriented software is produced by SOSPL, it is deployed onto runtime environment that consists of SOARBus (our implementation of ESB), service containers, composite service engines and service evolution management modules.

Finally, SaaS-based service provisioning and consuming environment aims at providing a simplified UI for different users of SOARWare including service providers, application developers and service consumers. The advantages of SaaS paradigm are well known for good usability. It helps save user efforts for installing and maintaining software. With our web-based unified access platform, service providers can publish their services to SOARBase; service developers can use online business process modelling and service orchestration tools powered by Adobe Flex technology to develop application services; finally service consumers can submit application requests and monitor the processing of their requests.

III. CONCLUSIONS

As a new computing paradigm, service oriented computing provides a novel approach to software development. Not only it can help improved software productivity through reusing standard based services, but also it can improve the non-functional properties of software applications through dynamic runtime adaptation. In this paper, we describe SOARWare, a suite of middleware and tools to support service oriented software production and running. SOARWare focuses on service resource management, service oriented software production, runtime management and unified access mechanism. SOARWare can be adopted as a completed service oriented solution to software development, and users can also select some of the tools or middleware to provide their own solution.

We are planning to perform a thorough test and further improve the stability and usability of SOARWare. And we learn that productivity and software trustworthiness are two important objectives for software development. In future, we will continue to study trustworthy software production issues.

ACKNOWLEDGMENT

This work was partly supported by China 863 program under Grant No. 2007AA010301. And we thank Chao Zhao, Xu Wang, Yipeng Ji, Fanxin Deng, Xianyang Qu, Yunkun, Xiong, Ruofan Jin, Peng Gao, Xiao Xu and other colleagues for their valuable contribution to the design and development of SOARWare.

REFERENCES

[1] M. P. Papazoglou, P. Traverso, S. Dustdar, and F. Leymann, "Service-Oriented Computing: State of the Art and Research Challenges," in *IEEE Computer*. vol. 40 (11), 2007, pp. 64-71.

[2] M. Paolucci, T. Kawamura, T. R. Payne, and K. Sycara, "Importing the Semantic Web in UDDI," in International Workshop on web services, e-business, and the semantic web, 2002.

[3] C. Schmidt and M. Parashar, "A Peer-to-Peer Approach to Web Service Discovery," in International Conference on World Wide Web, 2004.

[4] X. Dong, A. Halevy, J. Madhavan, E. Nemes, and J. Zhang, "Similarity Search for Web Services," in the 30th International Conference on VLDB, 2004.

[5] I. Altintas, E. Jaeger, K. Lin, B. Ludaescher, and A. Menon, "A Web Service Composition and Deployment Framework," in IEEE International Conference on Web Services, 2004.

[6] S.-C. Oh, D. Lee, and S. R. T. Kumara, "Effective Web Service Composition in Diverse and Large-Scale Service Networks," IEEE Transactions on Services Computing, vol. 1(1), pp. 15-32, 2008.

[7] L. Zeng, B. Benatallah, M. Kalagnama, and Q. Z. Sheng, "Quality Driven Web Services Composition," in 12nd International Conference on World Wide Web, 2003.