Available online at www.sciencedirect.com
SCIENCE DIRECT®
Engineering Applications of Artificial Intelligence 17 (2004) 771–782
Engineering Applications of
ARTIFICIAL
INTELLIGENCE
www.elsevier.com/locate/engappai

A dynamically reconfigurable system based on workflow and service agents

Jian Cao[a,*], Jie Wang[b], Shensheng Zhang[a], Minglu Li[a]

[a]*Department of Computer Science and Technology, Shanghai Jiaotong University, Shanghai, 200030, PR China*
[b]*Department of Civil and Environment Engineering, Stanford University, CA 94305, USA*

Accepted 12 August 2004
Available online 12 October 2004

Abstract

Most mission critical software systems are being built by integrating multiple distributed components. A software service, which is self-described and managed, and can be discovered and invoked dynamically through the Internet, provides a new paradigm for the composition of software systems. As a consequence, services and service-oriented software architecture will play fundamental roles in autonomic computing, which promotes the concept of self-management for software systems. To support self-managed and service-based software systems, a critical issue is how to deal with the service-oriented architecture to support dynamically reconfiguration. A service-based dynamically reconfigurable system framework for supporting future self-managed software systems was proposed in the paper. In the framework, a service agent represents an intelligent service broker that offers a self-managed and integrated service to respond to the requests from the environment adaptively. A workflow engine in this framework coordinates these service agents to implement particular business functions. The structure of the service agent, including its plan model, the relevant reconfiguration method and a service optimization mechanism, were discussed in the paper. A case study and an implementation were also presented.

Keywords: Autonomic computing; Service; Service agent; Workflow

1. Introduction

Autonomic computing, as proposed by IBM (IBM, 2001), is an approach for a self-managed computing system with a minimum requirement of human interference. The term derives from the biological function of a body's autonomic nervous system, which controls key functions without conscious awareness or involvement of the body and the environment. The IBM autonomic vision seeks to solve some of the current complex IT software system development, deployment, and management problems by using eight principles of system design to overcome current limitations. These principles include the ability of systems to self-monitor, self-heal, self-configure, and improve their performances. Furthermore, autonomic systems should be aware of their environment, defend against attack, communicate with use of open standards, and anticipate user actions. Autonomic computing is an emerging research topic that offers a grand challenge for the entire IT community (Kephart and Chess, 2003).

Majority of today's mission critical business applications and systems are being built by integrating multiple distributed software components. However traditional component-based system structure is insufficient for supporting autonomic computing. For example, it is difficult to upgrade software components for new functionalities without shutting down the running system, and it is unable to optimize performances for

*Corresponding author. Tel./fax: +86 21 62933536.
E-mail addresses: cao-jian@cs.sjtu.edu.cn (J. Cao), jiewang@stanford.edu (J. Wang), sszhang@cs.sjtu.edu.cn (S. Zhang), li-ml@cs.sjtu.edu.cn (M. Li).

0952-1976/$ - see front matter © 2004 Elsevier Ltd. All rights reserved.
doi:10.1016/j.engappai.2004.08.030

both the existing and newly augmented components. In addition, to extend the functionalities of the current software system is usually not an automatic process and it requires plenty of additional programming works and re-configuration of the systems by experienced IT professionals. Thus, the cost of providing new functionalities for an existing system is high, and the risk of interrupting the currently running systems is unpredictable.

Services are well known building blocks in modern information systems. They are loosely coupled, mostly autonomous reusable software components that semantically encapsulate discrete functionality and are distributed and programmatically accessible over standard Internet protocols (Sleeper and Robins, 2001). Service is expected to bring forth the next paradigm for the evolution of information system. This new paradigm is derived from the concept that the functionalities can be represented as services, which are self-described and can be dynamically discovered and orchestrated by exchanging messaging through the network.

While service-oriented architectural might play a fundamental role in autonomic computing, more specifically, in the support of flexibility and self-reconfiguration of a software system, current service technology is still in its infancy for only satisfying the basic requirements of a truly self-managed systems. The major issues are: (1) service lacks the adaptability for meeting requirements that are not anticipated in the original design, and thus it is unable to support the on-demand service composition through activating improved services; (2) most services are designed without a defined ontology for supporting self-management and automatic service composition; (3) services are passive until they are invoked, and they are not able to provide alerts and updates when extra information of the service becomes available (Huhns, 2002). Other deficiencies include that the quality of the service can not be easily measured and optimized, and no fault handling mechanism is offered. As a result, it is unfortunate that the functions of supporting a self-managed component-based system are still largely missing in the current service-oriented architecture.

This paper presents an approach for enhancing the service-oriented architecture to support dynamic reconfiguration for service components, which is considered as a first step for implementing a self-managed autonomic software system. The ability to reconfigure system is useful for a variety of reasons, including adapting applications to changing environments, performing on-line software upgrades and optimizations, and extending basic application functionalities with additional services.

The framework proposed in the paper is based on an architecture consisted of workflow and service agents. The workflow engine plays the role of a global coordinator to control different agents to fulfill the business functions according to a pre-defined process model. A service agent stands for an intelligent service broker that can offer an integrated service to respond to the request from the environment. The service agent can also self-optimize its' service set and provide a fault tolerance mechanism to the service invocation. By adding new service information into the service agent, its' capability can be augmented and in turn the function of the whole system can be improved incrementally, adaptively and dynamically.

The paper is organized as follows. Section 2 presents an overview of the service-based reconfigurable system framework. Section 3 defines the structure of service agent in detail. A method that is based on a dynamic plan configuration approach for composing services is introduced in Section 4. Section 5 presents a service self-optimization mechanism of service agent. Section 6 provides a case study and the system implementation for a demonstration. Section 7 discusses related works, and finally, Section 8 briefly summarizes the paper and points out several directions for future work.

2. The framework of a service-based reconfigurable system

The framework of a service-based reconfigurable system is shown in Fig. 1. It is decomposed into two layers: design and deployment layer, and executing and monitoring layer.

In the design and deployment layer, there are altogether three tools, i.e., an agent-modeling tool, an agent deployment tool and a workflow modeling tool. By applying the agent-modeling tool, an agent model can be saved as a XML file. The tool is able to access different service directories, such as UDDI (OASIS, 2002) for web service and registry service of GLOBUS (Foster et al., 2002), to obtain the information about the available services. Agents are deployed to different computers using the agent deployment tool. Business processes are modeled as workflows by applying the workflow-modeling tool.

In the executing and monitoring layer, the workflow engine will drive process instances instantiated from workflow models. The activated tasks of a process instance are assigned to users or service agents through a message-oriented middleware (MOM). The workflow engine maintains a list for storing activated tasks, from which users (through a user interface) or agents can obtain a task (including the task information and the input data). Any information about the occurrences of the operations that can change the status of a task, such as startup or submission, is sent to the workflow engine. Basing on the workflow model and related events, the workflow engine updates the task list.