面向自主Web服务的注册中心模型及其实现技术

张子龙 毛新军 尹俊文 侯 富 陈 超
(国防科学技术大学计算机学院 长沙 410073)

摘 要 为了应对互联网环境的开放性和动态性,加强对Web服务态势的管控,并为自主Web服务应用的开发提供支持,对传统的SOA架构进行了扩展,提出了一个面向自主化Web服务的注册中心模型。该注册中心不仅支持Web服务的基本注册功能,还提供了对Web服务态势信息的管理能力。介绍了自主服务的生命周期模型以及描述模型,给出了自主服务注册中心的关键技术,并开展案例分析来验证模型和实现技术的可行性。

关键词 Web服务,自主化,注册中心
中图法分类号 TP311 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2014.11.023

Model and Implementation of Registry for Autonomy Oriented Web Services

ZHANG Zi-long MAO Xin-jun YIN Jun-wen HOU Fu CHEN Chao
(College of Computer, National University of Defense Technology, Changsha 410073, China)

Abstract In response to the openness and dynamics of the internet environment and in order to enhance the management of Web services posture and support the development of autonomy-oriented Web service (AOWS) application, we extended the traditional SOA, and proposed a model of Registry for autonomy-oriented Web services in this article. This Registry not only supports the basic register function, but also provides the functionality of managing posture of Web services. We introduced the model of lifecycle and description for AOWS, described the key technology of Registry for AOWS, and finally studied a case to validate the feasibility of models and implementation technologies.

Keywords Web service, Autonomous, Registry

1 引言

随着互联网的发展,越来越多的软件建立在互联网环境之上。针对互联网软件开发与应用的新技术层出不穷。Web服务就是一种在互联网环境中架构软件系统的技术。

Web服务[1]支持基于XML的接口定义、发布和调用,实现跨平台交互,并通过接口隐藏了实现的具体过程,从而使得实现与软硬件平台以及语言无关。它采用相对成熟的技术标准,具有较好的互操作性,支持异构平台间的交互。

随着Web服务的快速发展,Web服务越来越多,形式和复杂程度也各不相同,如面向软件的服务、面向硬件的服务、面向移动的服务等。因而传统的Web服务技术在有些情况下不能满足实际应用的需求。

由于互联网络的动态和开放的特点,运行在互联网上的服务的负载和状态也是随时变化的,因此,当进行服务选择时就不能仅仅依据服务的功能来选择,还需要根据服务当前的状态进行匹配和选择。这样,服务需要能够感知自身状态的信息,因此服务就要具有自主性[3],从而要对服务进行自主化[4]。同时,在对服务自主化的基础上为了能够对服务进行更好的管理和监控,就需要对Web服务的架构进行改进[5]。因此,需要研究面向自主化Web服务的服务架构和相关技术。本文主要研究在自主化Web服务架构中注册中心的模型及其实现。本文第2节介绍自主服务注册中心的功能模型;第3节介绍Web服务的自主化技术;在Web服务自主化的基础上,第4节介绍自主Web服务注册中心的实现模型;第5节进行相关的案例分析;最后对相关的工作进行比较和分析。

2 自主Web服务注册中心功能模型

2.1 传统注册中心功能的不足

当前的Web服务注册中心主要遵循的是UDDI标准[7],UDDI是Web服务中用来进行服务发布操作的关键技术,它对Web服务进行统一描述、发现和集成,同时UDDI和WSDL(Web Service Definition Language)技术组合用来实现Web服务的查找操作。

UDDI在逻辑上分为两部分:商业注册和技术发现。前者是用来描述Web服务的一份XML文档;后者则定义了一套基于SOAP的注册和发现Web服务的编程接口。

UDDI标准主要是针对服务的静态信息,其中注册和查询的是Web服务的功能,而对服务"干得怎么样"则缺少相应

到稿日期:2013-09-16 返修日期:2013-11-16 本文受国家自然科学基金(61070034)资助。
张子龙(1988-),男,硕士生,主要研究领域为面向Agent软件工程、面向服务架构,E-mail:ryyrzzl@gmail.com;毛新军(1970-),男,博士,教授,CCF会员,主要研究领域为面向Agent软件工程、自适应软件;尹俊文(1969-),男,博士,教授,主要研究领域为软件工程;侯 富(1988-),男,博士生,主要研究领域为面向Agent软件工程、面向服务架构;陈 超(1982-),男,硕士生,主要研究领域为面向Agent软件工程、面向服务架构。

的支持。因此，当我们查询服务信息的时候，只能查到服务具体的功能，而对服务当前状态等动态信息就无能为力了。这显然不能满足我们在动态开放的互联网环境下 Web 服务系统的支持需求。

另外，UDDI 仅提供服务信息的注册和查询，缺少对 Web 服务的管控能力[6]。而在动态开放的互联网环境下，对 Web 服务的管理和控制能力是必须的。

2.2 面向自主 Web 服务注册中心的功能模型

针对自主化 Web 服务的需求分析，面向自主化服务的注册中心与传统的注册中心相比有很多不同之处。面向自主 Web 服务注册中心的功能主要有以下几个方面：

1. 服务静态信息的注册与查询

即 Web 服务功能的描述信息，主要描述 Web 服务具有何种功能、能够提供什么样的服务，与传统的 Web 服务注册中心注册的内容相类似，包括描述服务的操作、每个操作的输入参数和输出参数，以及服务的调用方法和协议。同时，面向自主化服务注册中心还要能够自主地对这些静态信息进行验证与分析，保证信息的可靠性和准确性。

2. 服务态势信息的记录与查询

即 Web 服务状态的描述信息，比如服务当前是否在线、服务当前是否出现故障、服务的最大最小反应时间等，注册要能够对这些信息进行记录和查询，从而可以在运行时根据服务的运行状态，自主地选择合适的服务提供给服务使用者。这里需要解决的问题主要是服务运行状态的获取以及记录。关于服务运行状态的获取，我们在服务提供端增加了自主化能力，使得服务提供者成为一个自主化的服务提供者，能够对服务的状态信息进行监控，并向服务注册中心进行注册。

3. 自主化 Web 服务的生命周期管理

由于服务运行状态的动态变化，因此，注册中心需要增加对服务生命周期的管理，针对处于不同生命周期的 Web 服务进行不同的管理操作。比如，当服务处于忙碌状态时，不再向用户提供该服务的信息，因为即使提供了该服务也不能马上调用。有关自主化 Web 服务的生命周期管理会在后面详细介绍。

3 Web 服务的自主化技术

3.1 自主化 Web 服务的软件模型

要建立自主化的服务平台，首先就要对 Web 服务进行自主化封装。我们提出了 Web 服务的自主化软件模型 ASM (Autonomous Service Model)，实现了对 Web 服务自主化封装，解决了如何实现服务自主化的问题。

自主服务的软件模型(见图 1)由两部分组成：服务软件部件和自主管理部件(AMC)。自主管理部件 AMC 采用反应式的结构实现对服务软件部件的管控。它提供了服务管控的自主化决策和服务态势监控两方面的基本能力。

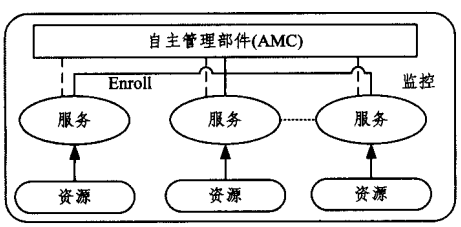

图 1 自主服务的软件模型

AMC 与服务软件部件的管控关系在运行时可根据具体情况进行动态调整，支持建立起面向任务的服务软件部件管控群。它使得所开发的自主服务软件具有以下几个方面的基本能力：服务运行态势的在线监控能力、自主服务的环境感知能力、面向应用协同的自主服务能力。

3.2 自主服务管控的架构体系

基于自主的服务软件模型，我们通过对现有的面向服务建构进行扩展，提出了对自主服务进行管控的架构体系(见图 2)，解决了如何对自主服务进行有效管控的问题。

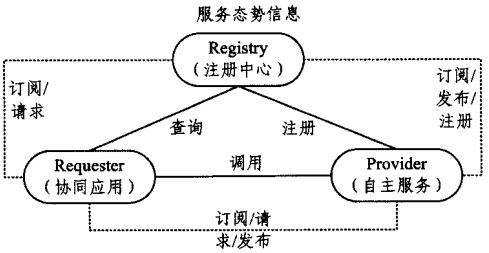

图 2 服务管控的架构体系

该扩展主要体现在以下几个方面：(1)自主的服务注册器 ASR(Autonomous Service Registry)。它除了提供传统 SOA 架构中服务基本信息的注册和查询之外，还支持对服务态势信息的组织、管理和发布。(2)自主的服务注册器 ASR 与自主服务以及任务应用之间的协同。项目提出了一组预定义的服务管控协议，从而实现 ASR 对服务态势信息的订阅、发布、获取、通知等方面的能力。项目提供了支持上述能力的协同机制。

在该模型中，自主服务除了传统 SOA 架构中的向注册中心登记服务基本信息之外，还负责通过 AMC 对服务的态势信息进行监视，将监控所获得的服务态势信息发布给协同应用或者服务注册中心，并对协同应用提出的服务提供请求进行自主决策和响应。服务注册中心除了提供服务基本静态信息的注册之外，还负责维护服务的态势信息模型，并根据订阅情况进行服务态势信息的发布。服务使用者可以通过服务注册中心查询服务、订阅服务态势信息，也可以直接与自主服务进行交互以订阅态势信息或者请求获得所需的服务。

注册中心实际上是一个特殊的自主化服务，它提供了系统中服务的注册/查询、服务态势订阅/发布等方面的功能。注册中心可以向其中注册的服务发出请求，以获取这些服务的态势信息，进而建立起系统中服务的态势模型，形成系统中服务的态势视图。当有协同应用或者其他自主服务向服务注册中心查询服务或者订阅服务态势信息时，它将根据请求提供和发布相关的信息。

服务请求者旨在获得所需的服务进而实现响应的任务。当应用请求者提出服务调用请求时，服务请求者会根据应用所订阅的服务及其态势描述信息向注册中心查询可供调用的服务，并根据返回的服务的态势信息结果进行计算，然后选择出其中最优的服务，最终向该服务发起调用。

4 面向自主化服务的注册中心的实现模型

4.1 自主化服务注册中心模型

面向自主化服务注册中心的模型如图 3 所示。

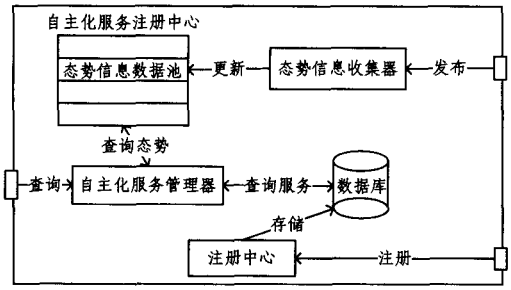

图 3　注册中心的实现模型

其中，服务注册中心提供的接口有服务态势信息的发布接口、服务静态信息的注册接口、服务的调用查询接口。自主化服务注册中心各个部件的功能如下：

1. 静态信息注册中心部件负责对从注册接口注册进来的自主化服务静态信息进行验证然后存储到数据库中；

2. 态势信息数据池部件是服务态势信息的存储部件，负责存储服务的态势信息，而对态势信息的支持是在元语言层次的支持，具体的描述将在下节进行介绍；

3. 态势信息收集器部件负责读取自主服务发布的服务态势信息，并将获得的服务态势信息存储到态势信息数据池中；

4. 自主化服务管理器是自主化服务注册中心的核心部件，负责处理用户的查询操作，自主化服务管理器会从数据库中查询服务的静态信息，以及从态势信息数据池中查询服务的态势信息，综合服务的态势信息和静态信息，将查询的结果返回给用户；另一方面，自主化服务管理器还要负责对数据库中服务静态信息和态势信息数据池中的服务动态信息进行检查，以保证信息的正确性和准确性。

4.2　自主服务的生命周期模型及其管理

为了加强对自主服务的有效管理，促进对其灵活管控，掌握其态势的变化，我们设计了一种自主化服务的生命周期模型（见图 4）。该模型包含了自主服务的以下状态：部署状态、自主化运行状态、失败状态、卸载状态。自主化运行状态是一个复合状态，它包括等待、开始和停止 3 个状态。等待状态是指服务当前已经被部署到了运行环境中，但是并没有进行相关业务的调用或者运行监控的一种状态。开始状态是指服务正处于业务调用或者运行监控的一种状态。停止状态指服务当前处于一种被禁止提供业务调用和运行监控能力的状态。

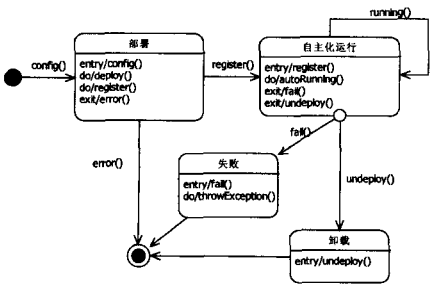

图 4　自主化服务生命周期模型

当一个服务被部署到其运行环境中后，该服务就处于部署状态，在此期间服务自主管理部件对服务进行部署设置，同时将服务的自主化描述信息注册到自主化服务的管控部件中。当完成自主化服务注册任务时，服务变为自主化运行状态。当服务进入到自主化运行状态后，它首先处于等待状态；当有业务调用或者运行监控时，服务由等待状态变为运行状态；当该服务被禁止提供业务调用或者运行监控时，服务由运行状态变为停止状态；当服务重新被允许进行业务调用或运行监控时，服务由停止状态变为等待状态。在自主化服务运行过程中，当业务调用或者运行监控失败时，服务运行状态变为失败状态。同样在自主化服务运行的过程中，当服务被卸载后服务的运行状态变为卸载状态。

4.3　自主服务的描述模型

针对开放动态互联网环境下 Web 服务的动态性、持续变化和自主化的特点，为了支持对自主化 Web 服务的高效、规范化的组织和管理，加强不同 Web 服务协同之间的交互和互操作，我们设计了自主化资源的描述模型 ASDM（Autonomous Service Description Model），以实现对应用协同的服务资源多维度描述。自主服务的描述模型包括 3 个部分：服务基本信息、自主化特征和态势信息。ASDM 与 WSDL 的区别在于 ASDM 中加入了服务自主化描述和态势描述两部分内容。

（1）服务基本信息描述

服务基本信息（见图 5）主要实现了对服务的数据类型、操作接口和服务标识信息的定义。其中，类型定义描述了实现服务功能的操作所需的各项参数（输入参数和输出参数）的类型；而操作接口定义则是服务将其对外所提供的功能操作以方法接口的方式进行的定义和描述，在该定义中主要描述了操作接口的名字、输入参数、输出参数和出现故障时所抛出的异常；服务的标识信息则是对服务本身能力的一种简介，它主要包括服务名称和服务描述两部分。服务基本信息中的数据类型和操作接口的描述与 WSDL（Web Service Description Language）中数据类型和操作接口的定义相类似。

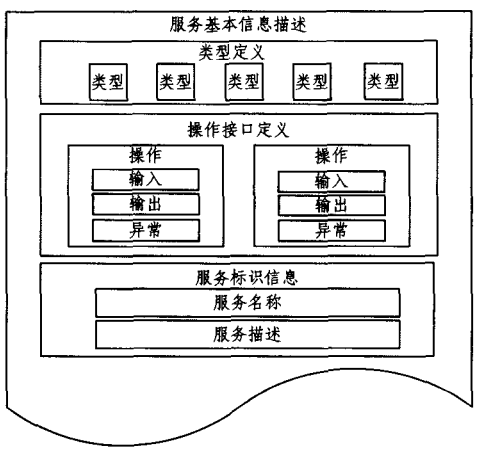

图 5　ASDM 的服务基本信息模型

（2）自主化特征描述

SDM 中的自主化特征描述刻画了自主化服务的基本标识信息，它主要描述了实现服务自主化封装的自主管理部件的标识信息（见图 6），包括服务自主化封装部件的名称、描述和地址的描述。对自主化特征信息的描述有助于实现对自主服务的高效管理、监控和调用。

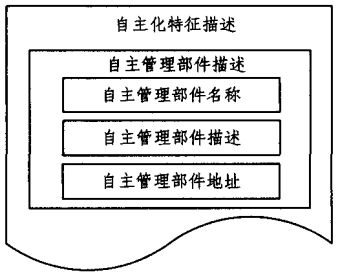

图 6　ASDM 的自主化特征模型

(3)态势信息描述[8]

态势信息描述刻画了 Web 服务的动态变化信息，它由两部分内容构成：与业务相关的态势信息和与业务无关的态势信息(见图 7)[9]。其中与业务无关的态势信息是指自主化服务的基本态势属性，主要包括服务的响应时间信息、服务状态信息、请求数量信息、响应数量信息、调用出错数量信息和最大实例数信息，其中响应时间又分为最大响应时间和最小响应时间，而服务状态信息则包括部署状态、等待状态、运行状态、停止状态、卸载状态和失败状态(见图 8)。与业务相关的态势是对一组与自主化服务本身业务功能相关的动态变化信息(即态势属性)的刻画，态势属性包括态势的属性名称、属性值和属性说明 3 个方面的信息。需要强调的是，在态势描述中，这种态势信息不管是业务无关的还是业务相关的，都需要对新信息的监视端口做出声明。

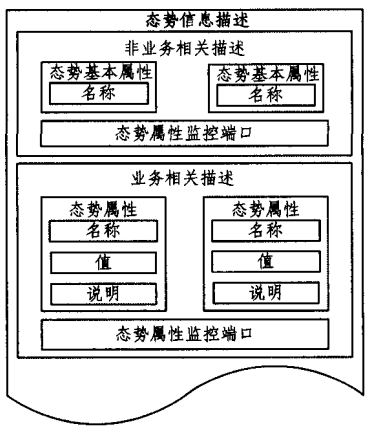

图 7　ASDM 的服务态势信息描述

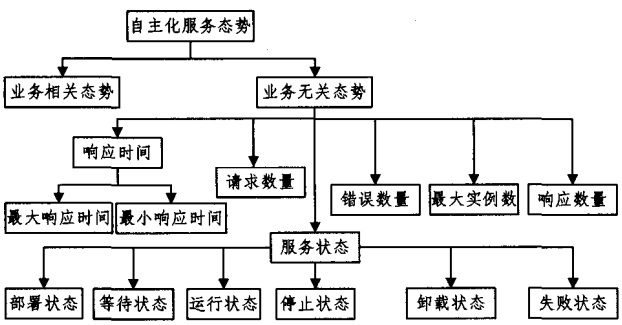

图 8　自主化服务的态势模型

通过该描述模型，不仅可以实现对异构资源静态信息进行描述，而且有助于实现对异构资源的动态信息进行刻画，从而加强对服务资源的发布、组织、监视、控制和管理，同时为自主化服务自主决策和按需监控奠定基础。需要强调的是，ASDM 模型是开放的，允许根据应用需求进行扩展和调整。

通过对自主化服务注册中心模型、自主化服务的生命周期管理和态势描述模型的设计，就可以对面向自主化服务的注册中心进行实现。下面就分别从自主化服务静态信息的存储与查询、自主化服务态势信息的存储与查询以及面向自主化服务的注册中心整体架构 3 个方面对面向自主化服务的注册中心的实现进行介绍。

4.4　自主化服务基本信息和自主化信息的存储与查询

由于自主化服务的基本信息和自主化信息通常是不会发生改变的，在服务部署时就可以确定，故将服务基本信息和自主化信息作为服务静态信息。我们将自主化服务静态信息存储在 MySQL 数据库中。静态信息的存储和查询模块的架构如图 9 所示。

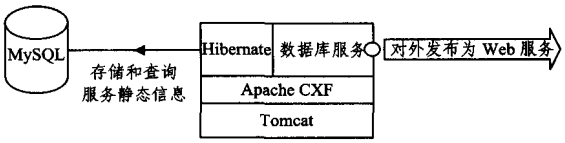

图 9　静态信息存储查询模块

MySQL 中存储服务的静态信息，使用 Hibernate 框架来与数据库进行交互。在这个基础上我们使用 Apache CXF Web 服务平台建立数据库服务。数据库服务对外提供对自主化服务静态信息的存储与查询的接口，并通过 Hibernate 与 MySQL 数据库进行交互。同时，数据库服务对外发布为普通的 Web 服务，这样就能统一系统中的所有部件，从而简化系统的管理开销。

4.5　自主化服务态势信息的存储与查询

自主化服务的态势信息主要是服务的运行状态等信息，是会随时变化的，因此，我们将自主化服务的态势信息存储在内存中。态势信息的存储和查询模块的架构如图 10 所示。

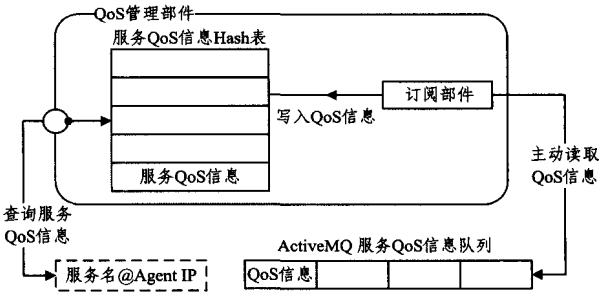

图 10　态势信息存储和查询模块

由于每个 Web 服务总是与其管理的自主化部件相联系，因此我们使用“服务名@Agent IP”来标识每一个自主化服务。这样，就可以使用 Hash 表来存储服务的态势信息，用“服务名@Agent IP”作为索引。之所以用 Hash 表存储是因为对态势信息的查询主要是通过标识来查询，使用 Hash 表可以更快地查找到所需的信息。

我们使用 Apache ActiveMQ 消息总线来传递服务的态势信息，服务提供端将服务的态势信息发布到 ActiveMQ 消息队列中，订阅部件主动地从态势信息消息队列中获取服务态势信息，再将态势信息存入服务态势信息 Hash 表中。

4.6　面向自主化服务的注册中心整体架构[10]

注册中心的整体架构如图 11 所示。

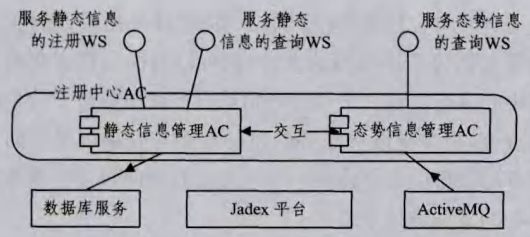

图 11　注册中心总体架构

我们借助 Jadex 平台提供的 AC(Active Component)[11]来实现管理部件的自主化能力，静态信息管理 AC 通过数据库服务来存储和查询服务静态信息，同时，也对信息进行检查和验证，态势信息管理 AC 从 ActiveMQ 中获取服务的态势信息，并存储下来。注册中心对外提供服务静态信息和态势信息的查询和存储接口。同样地，为了便于系统的管理同样将这些接口都对外发布为 Web 服务。

5　案例分析

考虑如图 12 所示的一个自主化 Web 服务案例，即互联网环境下的网络订票网站。

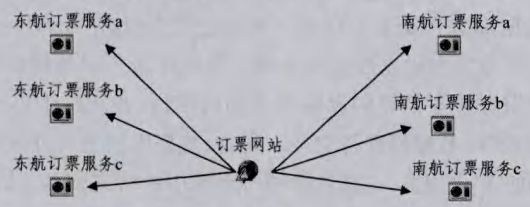

图 12　网络订票网站

该订票网站可以订购东航和南航的机票，同时，东航和南航各提供 3 个 Web 服务供该网站使用，分别是东航订票服务 a、b、c，南航订票服务 a、b、c。由于互联网络的动态和开放的特点，这些订票服务的负载和状态也是随时变化的，因此，当用户订票时，订票网站为了能够为用户提供更好的服务，就需要根据每个订票服务当前的状态信息来选择合适的服务以满足用户的需求。而且，服务的状态在运行时会发生变化，比如某个订票服务比较繁忙，或者某个订票服务出故障不能使用等。因此，订票网站不能像传统的 Web 服务一样对 Web 服务进行绑定操作，而要在运行时根据注册中心查到的服务状态动态地选择和调整服务，从而完成用户的需求。

自主 Web 服务与一般 Web 服务的区别主要在于：一般的 Web 服务只需要考虑 Web 服务的功能描述以及调用；而自主 Web 服务(比如案例中的网站订票服务)除了要考虑 Web 服务的功能以外，还要对服务的动态信息进行监控，从而能够根据服务的当前状态向用户提供更好的服务。

我们在开发的自主 Web 服务的注册中心及相关平台之上对这个案例进行了模拟实现。由于自主化 Web 服务能够监控自身的态势信息并向注册中心发布，因此，注册中心能够了解服务的态势信息，并在运行时根据服务态势信息选择合适的服务供订票网站使用。

当某个订票服务请求过多时，可以看到该服务的平均反应时间上升，注册中心也会选择另外的服务供订票网站使用。而当我们停止某个订票服务时，注册中心中的数据也可以动态更新、变化，从而保证数据的可靠性和可用性。

图 13 是我们实现的服务管控工具界面，通过该工具可以实时显示服务的动态信息，能够显示的动态信息有最小反应时间、最大反应时间、响应次数、当前访问次数、最大访问次数、错误次数、服务地址、服务状态等。经过实验，在系统运行中可以实时获得当前服务的态势信息，并根据服务的态势信息选择合适的服务进行调用。

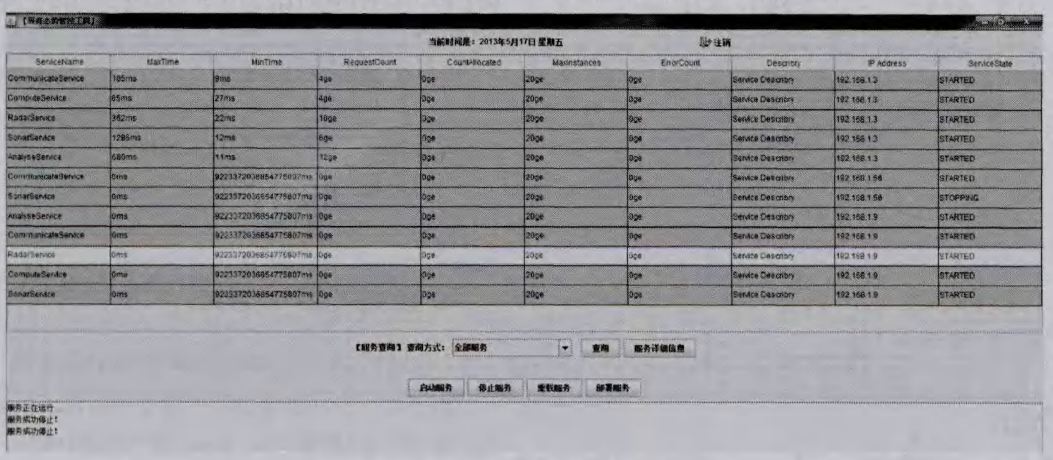

图 13　服务态势信息显示

这个案例验证了我们提出的自主 Web 服务注册中心的模型以及实现技术的正确性和可用性。应用该自主 Web 服务注册中心的模型和实现技术框架，能够对自主 Web 服务的动态信息进行实时的监控，并将其记录在注册中心之中。在此基础上，可以根据服务动态信息选择合适的服务进行响应。

6　相关工作比较

比较相关研究工作，比较有代表性的主要是 Dino[12]和 AFAWS[13]。

Dino(Dynamic and Adaptive Composition of Autonomous Services)是由 Arun Mukhija，David S. Rosenblum 等人提出的一种方法。Dino 为自治服务提供了能够动态和适应地组合服务的运行时支持。

Dino 对 Web 服务的能力和需求基于 OWL-S 的语义进行了详细的描述和规约，能够根据语义描述自动地发现和组合服务从而完成需求的功能。同时，Dino 还设计了对 Web 服务 QoS 的监控，从而能够根据 Web 服务的当前状态选择合适的服务。

AFAWS(An Agent based Framework for Auton-omic Web Service)是由 Walid Chainbi、Haithem M-ezni 以及 Khaled Ghedira 等人提出的一个基于 Agent 的自治服务框架。这个框架的目的是用来加强 Web 服务运行中的一些自治能力。

AFAWS 对传统的 Web 服务三角结构进行了扩展,在 Web 服务提供者端和 Web 服务注册中心端增加了两个基于 Agent 技术的自主化管理部件,通过这两个部件对 Web 服务进行管控和监视,收集 Web 服务的 QoS 信息,从而能够对 Web 服务的行为进行调整和适应,以增强 Web 服务的自治能力。在注册中心端增加了 ARM(Autonomic Registry M-anager),ARM 对注册中心进行 QoS 以及服务信息的自主化管理。在服务提供者端增加了 ASM(Autonomic Service Manager),ASM 对 Web 服务提供者进行监视和自治调整。

综合现有的研究工作,目前解决的问题主要围绕以下几个方面进行:

1)Agent 与服务之间的互联互通;

2)如何提供自主化的服务;

3)如何根据需求动态的选择、组合服务。

尽管上述工作对如何在动态网络环境中构建服务系统进行了有力的探索,但是这些工作还存在着如下方面的不足:

Dino 方法能够对 Web 服务进行监控,能够根据 Web 服务的能力和当前的 Web 服务状态来动态地发现、选择和组合 Web 服务。但是 Dino 由于缺少自主化的管理部件,不能支持具有自主化能力的 Web 服务。

AFAWS 框架对 Web 服务进行了自主化和自治化的封装,通过引入管理 Web 服务的 Agent 来增强 Web 服务在动态开放的网络环境中的自主能力和自治能力。但是,AFAWS 仅仅是一个自治 Web 服务的框架模型,缺少构建自主化 Web 服务系统的技术路线和实现手段,没有实现真正的应用平台。

结束语 本文从 Web 服务的自主化需求出发,举例分析了当前互联网环境下 Web 服务系统遇到的问题,提出了自主化服务平台,并着重对面向自主化服务注册中心进行了设计和实现的研究。本文提出了面向自主化服务注册中心的软件模型,并针对自主化服务的特点,设计了自主化服务的生命周期模型和态势描述模型。从而在注册中心能够对自主化服务进行静态信息和态势信息的查询与注册,以及对自主化服务的监控。最后,验证并实现了面向自主化服务的注册中心,从而能够有效地支持互联网环境下 Web 服务系统的开发与运行。

下一步工作主要包括:通过引入语义能力提高对服务静态信息和态势信息的描述能力与匹配推理能力,以进一步提高系统应对开放动态互联网环境的能力。同时应用语义的推理能力提高服务信息查询的准确性。另外,对服务态势信息做进一步的完善,增强对更复杂态势信息的支持。

参 考 文 献

[1] W3C Web Service Architecture Working Group . Web Service Architecture[EB/OL]. 2004-02-11[2010-09-18]. http://www.w3.org/TR /2004/NOTE- ws- arch- 20040211

[2] Duggan D. Service Oriented Architecture: Entities, Services, and Resources[M]. John Wiley & Sons, 2012

[3] Nguyen X T, Kowalczyk R. WS2JADE: integrating web service with jade agents [C]// AAMAS 07/ SOCASE 07 Proceedings of the 2007 AAMAS Internat Ional Workshop and SOCASE 2007 Conference on Service-Oriented Computing: Agents, Semantics, and Engineering. Belin: Springer-Verlag, 2007: 147-159

[4] Dickinson I. Wooldridgel M. Agents are not(just)Web services: considering BDI agents and web services[R]. HPL-2005-1233. Utrecht, 2005

[5] Jin X, Zhao X. Research on Enterprise Application System Integration Based on Web Services and Agent[M]// Informatics and Management Science III. Springer London, 2013: 353-359

[6] Bellwood T. Universal Description, Discovery and Integration specification (UDDI) 3. 0[EB/OL]. 2002-07-19[2009-02-15]. http://PPuddi. orgPpubsPuddi-v3.00-published-2002 0719. htm

[7] Tamilarasi K, Ramakrishnan M. Design and Development of an Enhanced UDDI for Efficient Discovery of Web Services[M]// Advances in Communication, Network, and Computing. Springer Berlin Heidelberg, 2012: 109-114

[8] Zheng Z, Lyu M R. QoS-Aware Fault Tolerance for Web Services[M]// QoS Management of Web Services. Springer Berlin Heidelberg, 2013: 97-118

[9] Zeshan F, Mohamad R, Ahmad M N. Quality of Service Ontology Languages for Web Services Discovery: An Overview and Limitations[M]// Human Interface and the Management of Information. Information and Interaction Design. Springer Berlin Heidelberg, 2013: 400-407

[10] Gawinecki M, Cabri G, Paprzycki M, et al. Evaluation of structured collaborative tagging for Web service matchmaking[M]// Semantic Web Services. Springer Berlin Heidelberg, 2012: 173-189

[11] Braubach L, Pokahr A. Jadex Active Components Framework-BDI Agents for Disaster Rescue Coordination[J]. Software Agents, Agent Systems and their Application, 2012, 32: 57-84

[12] Mukhija A, Rosenblum D S, Dingwall-Smith A. Dino: Dynamic and adaptive composition of autonomous services[OL]. www.cs. ucl. ac. uk /research/dino/, 2007

[13] Chainbi W, Mezni H, Ghedira K. AFAWS: An Agent based Framework for Autonomic Web Services[J]. Multiagent and Grid Systems, 2012, 8(1): 45-68