

文章编号:1001-9081(2010)10-2754-04

支持多类终端与服务定制的 SaaS 软件服务架构

陈小兵¹, 武泽旭²

(1. 东南大学 计算机科学与工程学院, 南京 210000; 2. 四川大学 电气信息学院, 成都 610065)

(chenxiaobin@midea.com.cn)

摘要:由软件即服务(SaaS)模式中软件服务的概念与特点分析,可知现有的面向服务架构(SOA)不能完全支持软件服务的实现。此外,为了拓展软件服务的应用范围,实现多样化的服务,SaaS模式的软件服务必然要满足用户定制服务的需求,以及对多类用户终端的支持。通过对SOA的扩展,在其中引入了软件服务终端、软件服务端口、软件服务注册等模块,提出了一种支持多类终端与服务定制的SaaS软件服务架构,给出了该架构的结构与工作流程,并通过实验证明了该架构可以实现SaaS模式的软件服务,而且能够实现对上述功能需求的支持。

关键词:软件即服务;面向服务架构;软件服务;多类终端;服务定制

中图分类号: TP311.5 **文献标志码:** A

Architecture of software services based on SaaS model supporting multi-terminals and service customization

CHEN Xiao-bing¹, WU Ze-xu²

(1. School of Computer Science and Engineering, Southeast University, Nanjing Jiangsu 210000, China;

2. School of Electrical Engineering and Information, Sichuan University, Chengdu Sichuan 610065, China)

Abstract: Analyzing the conceptions and characteristics of the software services based on Software as a Service (SaaS) model concludes that the current Service-oriented Architecture (SOA) cannot be used to realize software services directly. In addition, with the view of expanding software service application and realizing diversity of software services, software services based on SaaS model should support service customization and multi-terminals. By extending the SOA, software service terminal, software service port and software service register model were introduced. An architecture of software services based on SaaS model which supports multi-terminals and service customization was presented. The structure and work flow of the architecture were also presented. The experimental results prove that the architecture can realize software service based on SaaS model, and also supports the requirements mentioned above.

Key words: Software as a Service (SaaS); Service-Oriented Architecture (SOA); software service; multi-terminal; service customization

0 引言

随着软件架构技术的不断发展,软件即服务(Software as a Service, SaaS)模式作为一种新的软件服务提供模式引起了各方面的广泛重视。SaaS模式是指由软件服务商提供基于网络的软件服务,完全负责维护软件服务所需的软硬件平台等后台工作,而用户则采用租赁的方式租用软件服务供应商提供的软件服务,并通过网络来使用这些服务^[1]。

SaaS模式改变了传统的基于软件版权的销售模式及软件服务的提供模式^[1],也改变了传统的软件生命周期模式。基于SaaS模式的软件服务具有一些重要的特点,主要的有:可扩展性(Scalable)、高效的多租赁支持性(Multi-Tenant-Efficient)及可配置性(Configurable)等^[2]。

在SaaS模式中强调的主体概念是软件服务,将软件功能通过网络以一种服务的形式提供给用户。基于软件服务的概念,在软件服务的架构与实现方面,本文认为现有的面向服务架构(Service-oriented Architecture, SOA)软件架构模式,不能够实现对SaaS模式的完全支持。主要原因在于:虽然SOA是基于网络的、可提供远程应用服务的分布式架构,但它并不

能满足SaaS模式所提出的软件服务概念。

根据OASIS组织对SOA的定义^[3]:SOA是一种组织与使用位于不同所属域的分布式功能的规范,可知在该定义中突出SOA的主体是分布式的功能,因此SOA更侧重于如何组织、使用分布式应用服务,而非SaaS模式中所强调软件服务。

SOA不能完全支持SaaS模式也可以由Web Services技术得到证明,Web Services是目前实现SOA的主流技术,而W3C组织在其定义中直接表明Web Services技术是用于机—机操作的软件系统^[4],但SaaS模式中提供的软件服务显然要实现人—机交互。

因此要实现SaaS模式的软件服务需要对SOA进行扩展,将人—机交互部分纳入SOA中。该扩展从应用功能的角度来看,是将人—机交互模块也作为应用服务加入SOA中;而从系统部署范围角度来看,是将SOA从服务器一端扩展到用户端。为了对SOA实现这样的扩展,本文提出一种支持多类终端与服务定制的SaaS软件服务架构,该架构实现了底层的软件服务功能与用户终端的松散耦合,从而使同一软件服务能适用于桌面终端、Web终端与移动终端等多类终端,并支持用户个性化配置与软件服务定制,从而满足SaaS模式下软

收稿日期:2010-04-02;修回日期:2010-05-15。

作者简介:陈小兵(1974-),男,江西湖口人,工程师,硕士,主要研究方向:软件工程、服务计算;武泽旭(1990-),女,北京人,硕士,主要研究方向:软件工程、信息系统。

件服务的多租赁性与可配置性特点要求。

1 SaaS 软件服务架构

1.1 SaaS 软件服务架构的框架

本文提出的 SaaS 软件服务框架如图 1 所示,图中所示虚线部分模块为本文扩展模块。从图 1 可以看出本文所提出的软件服务框架针对文献[5]中所提出的 SOA 进行了扩展,其中主要的扩展部分有以下两个方面。

1)在原 SOA 中的应用服务功能部分进行了扩展,主要有:应用服务注册扩展为软件服务注册,且在 SOA 中用于协同应用服务的业务流程模块。如 BPEL 引擎之上,添加了软

件服务端口与软件服务注册信息 (Software Service Registry Information, SSRI) 模块,其中:SSRI 用以存储与管理描述软件服务的注册信息模块,软件服务端口基于 BPEL 之上,提供软件服务功能 Web 服务端口,其端口采用 WSDL 定义与描述,用户通过软件服务终端组件与软件服务端口交互,实现软件服务的定制与使用。

2)用户终端部分,该部分实现了 SOA 扩展到用户端的功能,主要包括六个模块。

身份认证支持 主要包括提供给软件服务终端组件所需的用户身份认证信息或相关可信认证服务,如移动终端认证所需的 PIN 码,桌面终端所需的唯一标识等数据。

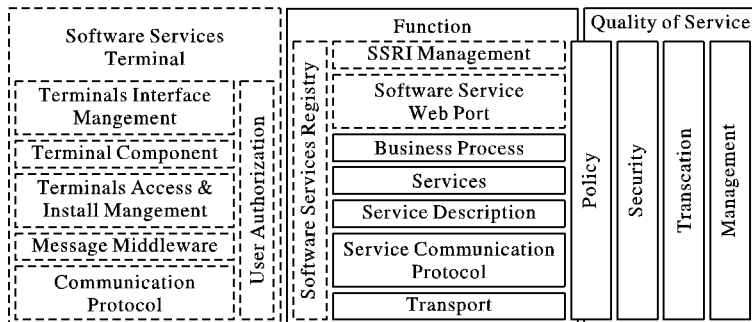


图 1 SaaS 软件服务框架

软件服务的界面管理 该模块主要提供各类用户终端界面的布局管理,向用户提供各种风格、可个性化配置的人—机交互界面。

软件服务终端组件 软件服务终端组件是用户定制的软件服务在用户端的代理组件,其功能是实现与其绑定的软件服务交互,以及向用户展示人—机交互接口。通过接收用户的操作消息,将其转换为软件服务请求,再通过消息中间层向目标软件服务转发请求数据,以及接收消息中间层发送来的软件服务响应数据,并向用户显示处理结果。

软件服务终端获取与安装管理模块 该模块是用户终端部分的核心模块,主要的功能是接收用户的软件服务定制请求,负责获取定制软件服务的终端组件与安装。

消息中间层模块 用于接收软件服务终端组件发送来的用户操作消息与转发软件服务响应消息到相应的终端组件。

通信协议 用于向界面组件管理框架中的界面组件提供底层的通信协议支持,包括多种网络通信协议,如 TCP、UDP、HTTPS、HTTP、SOAP 等或加密机制等。

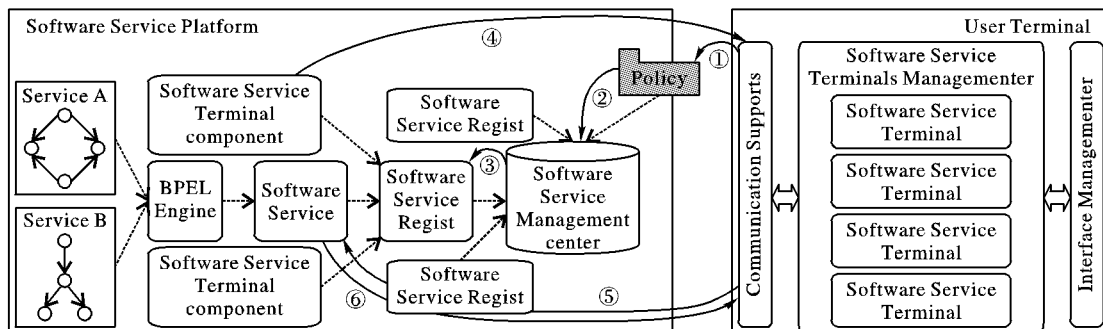
以上 SaaS 软件服务平台框架所包含的功能模块,不同的

用户终端类型可能包含的功能模块并不一致,如 Web 终端中并不包含用户身份认证模块,这部分的功能将由软件服务平台中的应用服务来完成;而桌面终端中则根据软件服务的不同,需要对桌面终端进行认证。

1.2 软件服务平台的工作流程

本文所提出的 SaaS 软件服务架构中各模块的协作关系如图 2 所示。

从图 2 中可以看出,在软件服务架构的服务器端中,底层是由 Web 服务构成的,而基于 Web 服务之上是工作流引擎,通过工作流引擎来协同多个 Web 服务来形成特定的软件服务功能,并且通过 Web 服务端口提供相应的软件服务功能。在软件服务架构中,每个软件服务功能都对应面向不同用户终端的软件服务终端组件。在 SaaS 软件服务架构中,通过软件服务注册信息 SSRI 描述软件服务端口的定义以及与其对应的软件服务终端组件信息,包括提供终端组件的 Web 服务、组件的类型等信息。在软件服务架构的服务器端中,软件服务的实现、注册、用户获取软件服务组件及安装的工作流程如图 2 中的虚箭头线所示。



注: ①软件服务终端获取与安装管理模块发出服务定制请求; ②经过安全策略检查,将定制服务信息发送至软件服务管理中心; ③由软件服务管理中心根据用户定制服务需求,检索软件服务注册信息; ④根据用户定制服务需求返回服务终端组件; ⑤、⑥由软件服务终端组件实现用户发送服务请求与接收服务响应; -----> 表述 SSRI 信息的组成与注册过程。

图 2 软件服务平台模块工作流程

软件服务架构中用户终端的各部分模块之间的协作以软件服务终端获取与安装管理模块为核心,由其访问提供终端

组件的 Web 服务,并在获取终端组件后实现终端组件的安装,而软件终端则负责处理消息事件,并通过消息中间层模块

与软件服务架构中的软件服务交互,最终用户显示服务的处理结果。

软件服务架构的工作流程如图2中的实箭头线所示。其中工作流程主要分成两部分。

1) 用户定制软件服务:当用户通过软件服务终端的界面管理模块定制软件服务时,软件服务终端获取与安装管理模块通过消息中间件模块向软件服务架构发出请求,软件服务架构根据相关策略判定用户具有定制该项服务的权限后,通过软件服务注册管理模块查询用户请求的定制服务信息,并返回提供定制服务终端组件的 Web 服务信息。软件服务终端获取与安装管理模块根据用户使用的终端类型,通过该 Web 服务下载定制服务的终端组件,然后完成定制服务终端组件的安装与初始化等工作。

2) 用户使用软件服务:用户通过软件服务终端组件与软件服务进行交互,终端组件通过绑定的软件服务端口等信息,由消息中间层模块向软件服务发送用户服务请求,接收消息中间层模块转发的软件服务消息,通过用户终端界面管理模块向用户显示软件服务的处理结果。

由上述流程可知,本框架的软件服务定制功能是以软件服务终端获取与安装管理模块为核心实现的,该模块根据用户输入的软件服务信息、查询到目标软件服务的终端组件,获取并安装组件,一旦软件服务终端组件安装与初始化完成,用户就可以通过界面管理模块与该组件实现交互,以该服务终端组件为中介,访问和使用定制的服务。

由图2中 SSRI 信息对软件服务与多类终端的绑定可知,本框架实现了同一服务绑定多类终端组件。用户可以根据其使用的终端类型选择适当的终端组件。终端组件本质上是用户终端与软件服务的中介,对于软件服务屏蔽了不同终端类型的差异性,因此在框架中对已有的软件服务,可以开发出多类终端组件,借助 SSRI 信息,不需要对该软件服务进行任何变更就可以实现该软件服务对多类终端的支持。

1.3 软件服务注册信息

基于上述的 SaaS 软件服务框架可知,软件架构中注册信息是整个平台的核心,用户终端管理模块正是通过软件服务注册信息,获取其定制的软件服务的终端组件,并通过终端组件实现定制服务的使用。因此软件服务架构中需要实现软件服务终端组件与软件服务的绑定,并提供相关信息使用户可以获得软件服务终端组件。这类信息在本文中称为软件服务注册信息(SSRI),在图2所示的软件服务架构中,SSRI 通过软件服务器注册器存储于软件服务管理中心。本文对于 SSRI 的定义如图3所示。

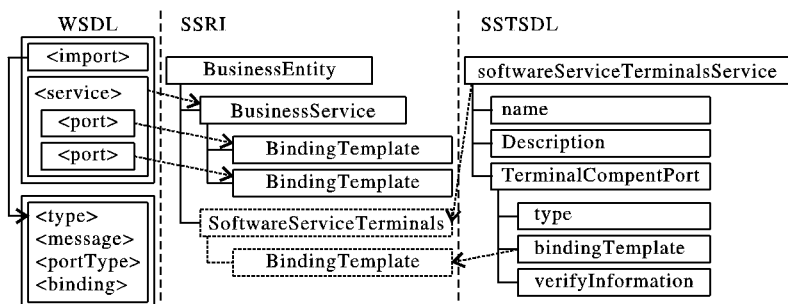


图3 SSRI与WSDL、SSTSDL的映射关系

SSRI 的数据来源主要包括两个部分:其一来源于 WSDL 文件,来自于被注册的软件服务所对应的 Web 服务;另一部

分则是软件服务终端服务定义文件 (Software Service Terminals Definition Language, SSTSDL)。其中:来自 WSDL 文件的 Web 服务定义信息映射到 SSRI 的 BusinessEntity 元素中,其映射的方式与 WSDL 映射到 UDDI^[6]是一致的;而 SoftwareServiceTerminals 元素则是来映射于 SSTSDL 文件中。SSTSDL 文件的 XML Schema 定义如下。

```
<element name="softwareServiceTerminalsService">
  <type content="elementOnly">
    <group order="seq">
      <element ref="name"/>
      <element ref="Description" minOccurs="0"
        maxOccurs="*" />
      <element name="TerminalCompentPort" minOccurs="1"
        maxOccurs="*" />
      <type content="elementOnly">
        <group order="seq">
          <element name="type">
            <simpleType>
              <restriction base="string">
                <enumeration value="WEB"/>
                <enumeration value="MOBILE"/>
                <enumeration value="DESKTOP"/>
              </restriction>
            </simpleType>
          </element>
          <element ref="bindingTemplate"/>
          <element ref="verifyInformation"/>
        </group>
      </type>
    </group>
  </type>
</element>
```

从中可以看出,SSTSDL 元素中除了一些描述信息之后,最主要的元素是 TerminalCompentPort,描述软件服务终端组件的信息,该信息包括三个元素,其中:

type 用以定义该终端组件所适应的终端类型,是一个值限定的元素,有三个可选项对应桌面、网络、移动三类不同终端。

bindingTemplate 与 UDDI 中对于 Web 服务的技术描述信息是一致的,该元素是描述提供软件服务终端组件的 Web 服务信息,包括提供终端组件的 Web 应用服务地址、宿主等信息。

verifyInformation 为校验元素,终端组件传递消息的端口信息,主要包括数据类型、消息类型、端口与端口绑定等相关信息。SSRI 使用校验元素的信息实现终端组件与其对应的软件服务端口一致性校验。只有校验一致的 SSTSDL 元素才会映射成 SSRI 中的 SoftwareServiceTerminals 元素。

从 SSTSDL 的定义可以看出,用户获取软件服务终端组件仍然由 Web 服务来实现,即用户通过软件服务终端管理模块访问 SSRI 注册信息,获取定制软件服务的 TerminalCompentPort 元素,由该元素中的 bindingTemplate 访问提供定制软件终端组件的 Web 服务,再通过该服务获取终端组件。由此可知,SSTSDL 定义提供了用户获取软件终端组件的途径。

SSRI 元素的定义中,软件服务对应的 Web 服务端口与提供终端组件的 Web 服务是分开展示的,从而实现软件服务与终端组件解耦,这使得软件服务发生变更时,不会对用户产生附带的关联影响。

此外,SSRI 元素中只定义了终端组件的获取相关信息,对于终端组件中采用何种方式绑定与其交互的软件服务并没有给出定义,因此在终端组件中需要由组件提供商或软件平台运营商提供绑定的相关信息。

2 软件服务架构实验

为了验证上文所述的软件服务架构,本文通过 AXIS 组件架设了一个查询天气的软件服务,分别由两个 Web 服务组成。

1) WeatherQuote。该 Web 服务中提供两个 Web 服务端口,在本文实验中,以这两个 Web 服务代表两个软件服务端口:其中一个软件服务端口提供天气情况的查询,返回查询地点当天、明天、后天的天气情况;另一个软件服务端口提供的是温度查询,返回查询地点当天、明天、后天的天气情况。

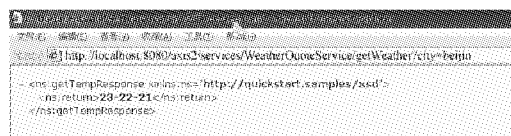
2) WeatherQuoteTerminals。该服务针对上述的两个软件服务端口分别提供 Web 终端组件与移动终端二类软件终端组件。

在本文实验中,Web 终端中界面管理模块、终端组件的获取与安装模块都由 Javascript 语言实现的一个 <div> 层元素管理器来实现。消息中间件模块则是由 AJAX 技术实现的一个服务请求与发送器来实现,该模块统一接收各服务终端组件的消息,通过 AJAX 向软件服务发送服务请求。而各服务终端组件则分别由一个 <div> 层来实现。

移动终端中终端组件的获取与安装模块、软件服务终端组件都是通过 Java ME 技术实现的移动应用程序,移动设备需要先安装终端组件的获取与安装模块应用程序,再由该程序通过终端组件服务端口下载终端组件后安装。

在实验中 SSRI 信息则是通过另外架设的一个 Web 服务来实现,该服务直接将 SSRI 元素提交给各终端的终端组件的获取与安装模块,由该模块解析后,通过绑定的 Web 服务地址来实现对终端组件的下载与安装。

图 4 分别显示了查询温度时,软件服务 Web 端口返回的 SOAP 消息以及用户使用获取的查询天气的 Web 终端组件访问天气软件服务时展示的软件服务响应数据及界面情况。



(a)



(b)

图 4 查询天气返回的 SOAP 消息与 Web 终端展示的界面

图 5 是用户同时定制温度与天气软件服务时,使用 Web 终端组件访问软件服务时显示的服务界面与数据界面,可以发现,查询天气的软件服务终端组件将软件服务的响应数据转换成图片的形式展示给用户,而查询温度的软件服务组件

则直接将回应的数据展示给用户。

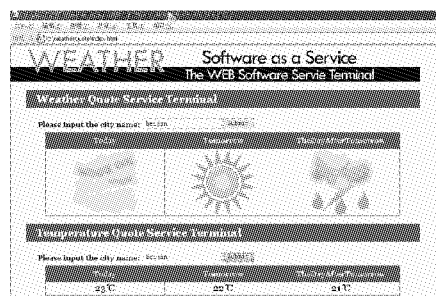


图 5 用户同时定制温度与天气查询时 Web 终端展示界面

图 6 显示的是使用 Java ME Platform SDK 平台模拟的移动终端访问温度查询软件服务时显示的文字界面。从图 5 与图 6 中可以看出,虽然 Web 终端组件与移动终端组件对于软件服务展示的界面不同,但它们访问的是同一个软件服务,只是各组件对软件服务数据的展示方式不同。

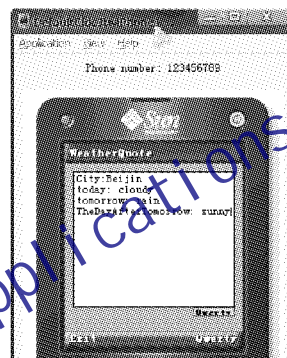


图 6 移动终端展示的天气软件服务界面

以上的实验可以看出,在本文的框架下,查询天气的两个 Web 服务并没有作任何改变,只是通过另一个 Web 服务提供的 SSRI 信息,用户分别获取了由移动应用程序与 Javascript 实现的移动终端与 Web 终端组件,就可以通过不同终端访问同一个查询天气的 Web 服务。因此,本实验表明了本文框架具有向用户提供基于 SaaS 模式软件服务的能力,同时也支持软件服务的定制与同一软件服务跨越不同用户终端的功能。

3 总结与展望

本文针对 SaaS 模式的软件服务所具有的特点,指出现有的 SOA 不能完全实现对 SaaS 模式的软件服务支持,且相对于传统的软件销售模式,软件服务提供商需要面对更多的挑战。

由于 SaaS 软件服务的提供商一般是针对某个行业或领域,如 ERP,提供多种软件服务,且面向需求变化,软件服务处于不断的变更之中,因而 SaaS 模式的软件服务除了具有多租赁性、可扩展性之外,更具有动态性与多样化,如软件生态^[7]的概念就反映了 SaaS 模式软件服务的这样的特点。软件服务提供商面对多样化的软件服务必然实现能满足用户需求的服务定制,以及面向多用户终端的软件服务,以进一步扩展软件服务的应用范围。

为了实现满足 SaaS 模式上述特点的软件服务,文献[8-10]都对 SaaS 模式的软件服务架构做出了研究;但文献[8]关注是基于 Web 服务技术如何实现与提供软件服务,并没有考虑到客户端方面的问题;文献[9]则侧重于实现一种可操

(下转第 2762 页)

法的有效性 with 实用性。今后工作将立足于深入探讨不同层次不确定性调和之间更合理的自身调和分布问题,并赋予 Agent

一定的动态智能特性,从整体上改进调和模型的自适应性与群智能性。

表1 传感不确定性分层调和分布实例表

Agent 层次	Agent 名称	直辖节点		节点失效		数据项缺失		数据集建模		个体冗余		属性不一致		属性约简		属性依赖		规则冲突	
		类型	数量	策略	调和量	策略	调和量	策略	调和量	策略	调和量	策略	调和量	策略	调和量	策略	调和量	策略	调和量
SA	S_1 等	/	/	S_{NR}	5	—	—	—	—	—	—	—	—	—	—	—	—	—	—
	S_2 等	/	/	—	—	S_{MD}	30	—	—	—	—	—	—	—	—	—	—	—	—
CAA	C_1	Ns	10	—	—	—	—	S_{DT}	10	—	—	×	×	S_{IR}	0	—	—	—	—
	C_2	Ns	15	—	—	S_{MD}	12	S_{DT}	15	—	—	—	—	S_{IR}	0	—	—	—	—
	C_3	Ns	15	—	—	—	—	S_{DT}	15	S_{IR}	20	—	—	S_{IR}	1	—	—	—	—
	C_4	Ns	20	—	—	S_{MD}	15	S_{DT}	20	S_{IR}	30	×	×	S_{IR}	1	—	—	—	—
SDA	$Sink_1$	Nc	4	—	—	—	—	—	—	S_{IR}	100	—	—	S_{IR}	3	S_{CR}	6	S_{DN}	3

注:“/”表示 SA 层次不应存在下级节点;“×”表示存在 I_A 不确定性但在 CAA 层次先不应作处理而留待 SDA 层次处理;“—”表示在监测实例中没有发现相应类型不确定性。

参考文献:

- [1] 李德毅, 刘常昱, 杜鹄, 等. 不确定性人工智能[J]. 软件学报, 2004, 15(11): 1583-1594.
- [2] 周傲英, 金澈清, 王国仁, 等. 不确定性数据管理技术研究综述[J]. 计算机学报, 2009, 32(1): 1-16.
- [3] HATFIELD A J, HIPEL K W. Understanding and managing uncertainty and information [C]// IEEE International Conference on Systems, Man, and Cybernetics. Washington, DC: IEEE, 1999, 5: 1007-1012.
- [4] LAPLANTE P A. The certainty of uncertainty in real-time systems [J]. IEEE Instrumentation & Measurement Magazine, 2004, 7(4): 44-50.
- [5] GASSER L. Perspectives on organizations in multi-Agent systems [C]// The 9th Advanced Course on Artificial Intelligence. LNCS 2086. Berlin: Springer-Verlag, 2001: 1-16.
- [6] AKYILDIZ I F, SU W, SANKARASUBRAMANIAM Y, et al. Wireless sensor networks: a survey [J]. Computer Networks, 2002, 38(4): 393-422.
- [7] 于海斌, 曾鹏, 梁韡. 智能无线传感器网络系统[M]. 北京: 科学出版社, 2006.
- [8] CHENG R, PRABHAKAR S. Managing uncertainty in sensor databases [J]. SIGMOD Record, 2003, 32(4): 41-46.
- [9] DAI ZHIFENG, LI YUANLIANG, ZHENG BOJIN, et al. A distributed coordination framework for adaptive sensor uncertainty handling [C]// The seventh International Conference on Computational Science. LNCS 4490. Berlin: Springer-Verlag, 2007: 1171-1174.
- [10] PAWLAK Z. Decision networks [C]// The Fourth International Conference on Rough Sets and Current Trends in Computing, LNAI 3066. Berlin: Springer-Verlag, 2004: 1-7.

(上接第2757页)

作的成熟模型来供软件服务商集成现有的应用到软件服务平台中,同样存在软件服务的实现问题;文献[10]则提出实现软件服务必需要在架构中引入客户端部分,研究了客户端在适应 SaaS 模式方面的需求,也提出了一个理论框架。

本文基于 SOA 的扩展,提出了支持多类终端与服务定制的 SaaS 软件服务架构,不同于文献[10]的理论分析,本文给出架构的结构与工作流程,并通过实验证明了本架构可以实现对上述需求的支持,架构中使用 SSRI 实现了终端组件与软件服务松耦合,使软件服务平台中软件服务动态变化对于用户透明,因而更进一步支持了软件服务平台的动态性。因此,本文所提框架更具有实际意义。

今后将以已有框架为基础,针对软件服务的注册工具及软件服务 Web 端口与终端组件 Web 端口的一致性校验工具等方面继续开发与研究。

参考文献:

- [1] CUSUMANO M A. The changing software business: Moving from products to services[J]. Computer, 2008, 41(1): 20-27.
- [2] CHONG F, CARRARO G. Architecture strategies for catching the long tail[EB/OL]. (2007-07-13) [2010-01-11]. <http://msdn.microsoft.com/en-us/library/aa479069.aspx>.
- [3] MACKENZIE C M, LASKEY K, MCCABE F, et al. Reference model for service oriented architecture 1.0[S/OL]// OASIS, 2006 (2006-10-12) [2009-12-16]. <http://docs.oasis-open.org/soa-rm/v1.0/soa-rm.pdf>.
- [4] STOJANOVIC Z, DAHANAYAKE A. Service-oriented software system engineering: challenges and practices[M]. Hershey, PA, USA: IGI Publishing, 2005.
- [5] ENDREI M, ANG J, ARSANJANI A, et al. Patterns: Service-oriented architecture and Web services, SG24-6303-90[R/OL]// IBM Corporation, 2004 (2004-03-31) [2010-01-06]. <http://www.chinagrid.net/grid/paperppt/Patterns-Services.pdf>.
- [6] SRINIVASAN N, PAOLUCCI M, SYCARA K. Adding OWL-S to UDDI, implementation and throughput [C]// Proceedings of the First International Workshop on Semantic Web Services and Web Process Composition. New York: [s. n.], 2004: 213-215.
- [7] SÄÄKSJÄRVI M, LASSILA A, NORDSTRÖM H. Evaluating the software as a service business model: from CPU time-sharing to online innovation sharing[C]// Proceedings of the IADIS International Conference E-Society. Qawra, Malta: IADIS, 2005: 177-186.
- [8] TURNER M, BUDGEN D, BRERETON P. Turning software into a service[J]. Computer, 2003, 36(10): 38-44.
- [9] KANG S, MYUNG J, YEON J, et al. A general maturity model and reference architecture for SaaS service[C]// DASFAA 2010: Proceedings of the 15th International Conference on Database Systems for Advanced Applications, LNCS 5982. Berlin: Springer-Verlag, 2010: 337-346.
- [10] XIN MINGDI, LEVINA N. Software-as-a service model: Elaborating client-side adoption factors [C]// ICIS 2008: Proceedings of the 29th International Conference on Information Systems. Paris, France: AIS Electronic Library, 2008: 86.