一种支持动态管理的SCA服务模型研究

龙 军 赵贵虎 张祖平
(中南大学信息科学与工程学院 长沙 410075)

摘 要 为解决SCA和OSGi的结合在分布式环境下不能很好支持运行时组件模型动态管理的问题,在分析二者传统结合方式的基础上,建立了一种基于OSGi的SCA服务模型——DOSGi_SCA。DOSGi_SCA以分布式OSGi为基础,构建了服务注册中心来管理本地服务和远程服务,实现了在分布式环境下支持运行时组件模型的动态管理。应用实例表明,该模型实现了SCA和OSGi的优势互补,充分发挥了各自的优点,弥补了各自的不足。

关键词 SCA,动态管理,组件模型,分布式OSGi,DOSGi_SCA
中图法分类号 TP311 **文献标识码** A

Research of SCA Service Model for Dynamic Management

LONG Jun ZHAO Gui-hu ZHANG Zu-ping
(School of Information Science and Engineering, Central South University, Changsha 410075, China)

Abstract The traditional methods based on the combination of SCA and OSGi are not able to support the dynamic management of component modules in the distributed environment at runtime. Based on the analysis of the traditional methods, this paper proposed an OSGi based SCA service model -DOSGi_SCA. To support dynamic component module management at runtime in the distributed environment, DOSGi_SCA constructs a service registry center to manage local and remote services based on the distributed OSGi. The practical example demonstrates that this model takes advantages of both SCA and OSGi while avoiding their deficiencies.

Keywords SCA, Dynamic management, Component model, Distributed OSGi, DOSGi_SCA

1 引言

在软件工程中,为了提高软件的复用技术和软件的开发效率,需要把一个复杂的系统进行抽象,将其分解成为一个个可独立处理的可管理组件单元。但如何使组件单元能够具有动态可插拔性,并使系统保持各个部件的松耦合性等问题仍然是软件开发中所面临的重要问题[1]。

SCA提供了一个模型基础,用于创建基于面向服务架构(Service-Oriented Architecture, SOA)的应用程序和解决方案。SCA所依托的理念是将业务功能作为一系列服务组件提供,从而创建能满足特定业务需求的解决方案。这些服务组件可以包含为已有系统中的应用程序和业务功能创建的新服务,以及作为复合应用的一部分重用的应用程序。

OSGi(Open Service Gateway Initiative)最初是为了解决家庭网络或者嵌入式设备由于本身的限制(CPU、内存和带宽等)而提出的一个解决方案,是一个轻量级的框架。并且,由于OSGi在网络及动态性方面的优势,其已经应用于移动通讯、嵌入设备和企业开发等众多领域。

SCA和OSGi在关注点上有很大不同:SCA关注于组件与组件之间、服务与服务之间的交互和合成[2],而OSGi更关注于运行环境中组件和服务的动态更新特性[3]。因此,SCA与OSGi的结合可以实现SCA和OSGi的优势互补,充分发挥各自的优点,并弥补各自的不足[4]。

OSOA组织在2007年3月发布SCA正式规范SCA1.0的同时,还发布了一些扩展规范,其中包括描述SCA和OSGi结合的白皮书《Power Combination: SCA, OSGi and spring》。在这本白皮书中,描述了SCA与OSGi结合的基本理论,但是目前业界并没有提供一种实用的解决方案[5]。OSOA提出的白皮书也只是提供了理论上的支持,而实现的方法并没有提供[6]。

本文在分析SCA和OSGi传统结合方式的基础之上,深入研究二者的运行机制,建立了一种基于OSGi的SCA服务模型,以解决SCA和OSGi结合在分布式环境下不能很好支持运行时组件模型动态管理的问题,实现SCA和OSGi的优势互补,充分发挥各自的优点,弥补各自的不足。

2 SCA模型

Component(组件)是SCA组合构件中业务功能的基础元素。其通过SCA组合构件(Composite)联合成完整的业务解决方案。Component是实现的可配置实例。Component提

到稿日期:2012-02-15 返修日期:2012-06-08 本文受国家自然科学基金(60873081,M0921005,60970095)资助。

龙 军(1972-),男,博士,副教授,主要研究方向为网络资源管理与服务技术,E-mail:125953182@qq.com;**赵贵虎**(1983-),男,硕士生,主要研究方向为服务组合技术;**张祖平**(1966-),男,教授,博士生导师,主要研究方向为信息融合与信息系统、参数计算与生物计算、网络容错路由算法及协议。

供和消费服务。多个 Component 可以使用和配置为同一个实现，而每个 Component 又可以对实现有不同的配置。如图 1 所示，一个 Component 包括 4 个部分：服务(Service)、实现(Implementation)、属性(Property)和引用(Reference)[7]。

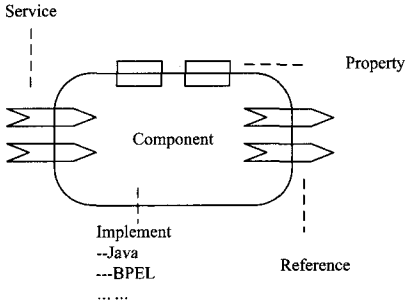

图 1 SCA Component

Implement(实现)是提供服务或引用其他地方所提供的服务的业务功能的具体实现。另外，某个 Implement 可以有一些可设置的属性值。SCA 允许从广泛的实现类型中选择任何一种技术，比如 Java、BPEL 或 C++。每种类型都代表了特定的实现技术。该技术不仅仅简单地定义实现语言，如 Java，也可以定义使用某个特定的框架或运行时环境，例如包括使用 Spring framework 或 Java EE EJB 技术的 Java 实现。

Component 通过服务提供它的业务功能，其他的 Component 通过访问该 Component 服务的形式来获取它的业务功能。如果 Component 要引用其他 Component 的服务，就必须要通过 Reference(引用)的形式。通常一个 Component 需要引用多个其他的 Component 的服务。Property(属性)允许通过外部设置的数据值来配置实现。数据值可通过组件提供，也可能源自此组件的上层组合组件的属性。

Composite 是 SCA 中另外一个非常重要的概念。Composite 是由一些有关联的 Component 组合而成，部署时以 XML 文件的形式构成。因为它是一个独立部署的单元，所以在分解系统的时候能够提供很高的灵活性。例如在更新一个系统时，只要保持接口不变，就可以随意替换发生变化的 Composite，而其他部分不会受到影响。

3 OSGi 模型

在 OSGi 规范中，bundle 是部署基于 Java 的应用的唯一实体。OSGi 服务由 bundle 提供，服务部署也是以 bundle 作为载体，bundle 可认为是一个软件构件，因此 OSGi 的计算模式也可被称为“面向服务的构件模式(Service Oriented Component Model)”。OSGi 框架的一个最重要的特性是动态更新。在一个动态扩展的 OSGi 环境中，OSGi 框架管理 bundle 的安装和更新，同时也管理 bundle 和服务之间的依赖关系。在一个 bundle 的生命周期中可能处于以下 6 个状态，如图 2 所示：

INSTALLED：安装完成，本地资源成功加载。

RESOLVED：依赖关系满足，该状态意味该 bundle 要么已经准备好运行，要么是被停止了。

STARTING：bundle 正在被启动，BundleActivator 的 start()方法已经被调用但是还没有返回。

STOPPING：bundle 正在被停止，BundleActivator 的 stop()方法已经被调用但是还没有返回。

ACTIVE：bundle 被成功启动并且在运行。

UNINSTALLED：bundle 被卸载且无法进入其他状态。

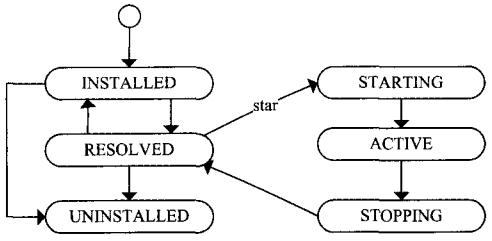

图 2 Bundle 状态图

从开发平台方面，OSGi 为动态扩充、修改系统功能和改变系统行为提供了支撑，而在传统的开发方式下，要实现系统的动态扩充、修改以及改变是一件很困难的事。从组件规范方面，OSGi 带来了规范化的组件组织以及统一的开发方式，这为组件组织、组件开发以及组件积累提供了一种全新的指导以及支撑。OSGi 提供的这些支撑能给具体的应用带来优势。

4 DOSGi_SCA 模型设计

4.1 模型结合方式

SCA 与 OSGi 的结合在《Power Combination：SCA，OSGi and spring》白皮书中给出了 3 种不同的方式：(1) SCA 组件作为 OSGi 容器内的服务；(2) SCA 程序集或 SCA 域作为 OSGi 容器内的服务；(3) OSGi bundle 作为 SCA 域中的组件。

上述 3 种 SCA 与 OSGi 结合的方式，其实实质上是两种：其一，以 SCA 为基础，采用 OSGi bundle 实现 SCA 服务，OSGi bundle 作为 SCA 域中的组件就属于这种方式；其二，以 OSGi 为基础，将 SCA 组件或者程序集作为 OSGi 中的组件，前两种都是属于这种方式。

OSGi bundle 作为 SCA 域中的组件是采用 SCA 的 OSGi Implementation Type 方法，是以 SCA 为基础，以 OSGi bundle 作为 SCA 组件，这样做可以充分发挥 SCA 自身的特点。就 SCA 而言，这是变动最小、最容易实现的方法。但是这种方法也存在局限性，没有发挥出 OSGi 的特点。组件之间的连接是采用 SCA 的 references 方式，没有应用到 OSGi 中的动态装载组件的特性，这就限制了它的灵活性。此外对于已有的 OSGi 服务，如何实现重用也是一个很大的问题，因为 OSGi 服务中的 bundle 可能很多，关系也很复杂，势必会造成 SCA 中的 scdl 描述很复杂。

以 OSGi 为基础，将 SCA 组件或者程序集作为 OSGi 中的组件，可以实现 OSGi 服务与 SCA 组件的交互。但是这种模式也存在局限，SCA 组件必须与 OSGi 服务在同一个 OSGi 容器中运行。这实际上是用 SCA 来实现 OSGi，OSGi 存在仍然不足。但是 OSGi 宿主不具备 SCA 所有的跨平台与跨语言特性，与非基于 OSGi 平台的应用程序的交互也不如 SCA 架构方便。

OSGi4.2 中分布式的 OSGi 规范的提出，为实现 SCA 和

OSGi 结合提供了更好的条件。2009 年 9 月最新发布的 OSGi 4.2 聚焦于分布式的 OSGi，这使得 OSGi 中装配的服务不是必须存在于同一个 JVM。使用分布式的 OSGi，可以使部署在一个 JVM 中的 OSGi 服务组件能够使用部署在其他 JVM 中的 OSGi 服务组件，甚至其他 JVM 中非 OSGi 服务组件，如 SCA 服务组件等。

4.2 DOSGi_SCA 模型

在分析传统结合方式的基础之上，建立一种基于分布式 OSGi 的 SCA 服务模型——DOSGi_SCA。在 DOSGi_SCA 模型中构建了服务注册中心，负责系统中的服务注册、注销及更新等的管理。通过引入服务注册中心实现了服务的统一管理，即将所有的分布式节点上的服务统一管理。

图 3 描述了在分布式环境下存在几个节点。其中 Node1 为服务中心，在 Node1 中构建一个服务注册中心。当应用启动时，Node1 中的服务注册中心以广播的形式将其地址发布到网络中，分布式环境中的其他节点接收到后记录下地址；接下来其他节点就可以根据地址提交要注册的远程服务的相关信息给服务注册中心。

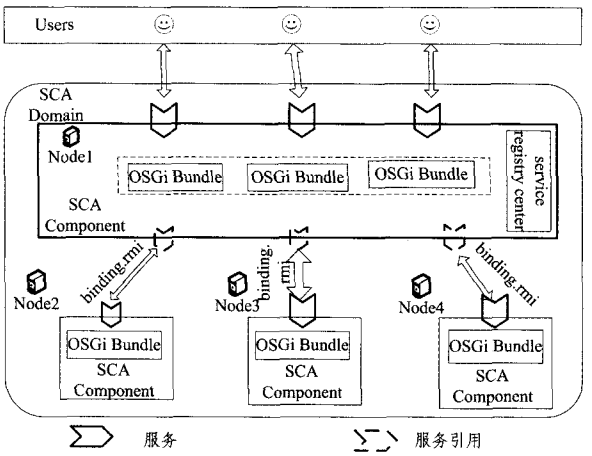

图 3 基于分布式 OSGi 的 SCA 服务模型

在 Node1 节点中的 SCA 服务是由分布式 OSGi 通过 Implementation Type 方式来实现的。并且这些服务通过服务注册中心查找要引用的服务，如果存在则服务注册中心返回满足要求的服务信息。接着根据返回的信息通过 binding. rmi 的方式引用分布在其他节点中的服务。

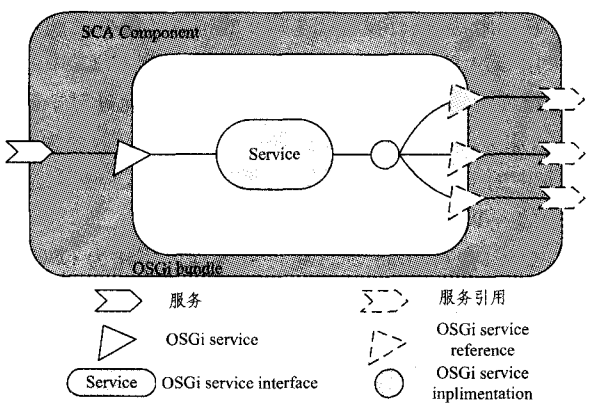

图 4 分布式 OSGi 实现 SCA Component

图 4 是图 3 在 Node 上的 SCA Component，这里的 SCA Component 是由分布式 OSGi 通过 Implementation Type 方式实现的。这里 SCA Component 对外提供的服务实际上是由内部的 OSGi bundle 提供的，在 OSGi bundle 内部又分为服务的接口与服务的实现，并且 OSGi bundle 可以通过服务注册中心查找引用 Node 节点以外的服务。

4.3 服务注册中心

本文在 DOSGi_SCA 模型中构建了服务注册中心，其结构如图 5 所示，包括以下几个部分。

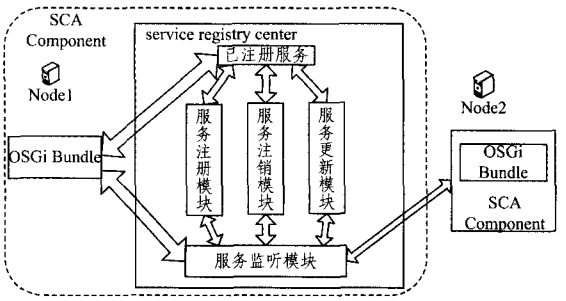

图 5 服务注册中心

已注册服务：负责记录 SCA 容器中所有已注册服务。其中每个服务对应唯一一条记录。在服务监听模块发现有新的服务增加时会通知服务注册模块，服务注册模块就会将新的服务记录在已注册服务表中。其中服务的描述、接口、状态等信息都记录在每条记录中。

服务监听模块：是服务注册中心的核心模块，当某个服务的状态发生改变时，与服务注册中心的其他模块通信。如果监听到了服务的更新，就调用服务更新模块，并以广播的形式通知所有的服务有服务更新。这样服务就可以及时地更新所引用的服务。服务监听模块类图如图 6 所示。其中 EventMonitor 对象为服务事件的监听对象，在增加新的服务时，会新建一个服务监听对象来对此服务的状态进行监听。如果被监听的服务注销后，该监听对象也会被销毁。在服务的状态发生变化时，其他模块就会收到该对象发送的消息。服务监听模块中对服务的启动、服务的停止与服务的注销 3 种状态进行监听。当新增的服务被注册时，该服务处在停止状态，需要用户启动服务才能正常使用，而当服务被注销时，与该服务对应的监听对象也会被释放。

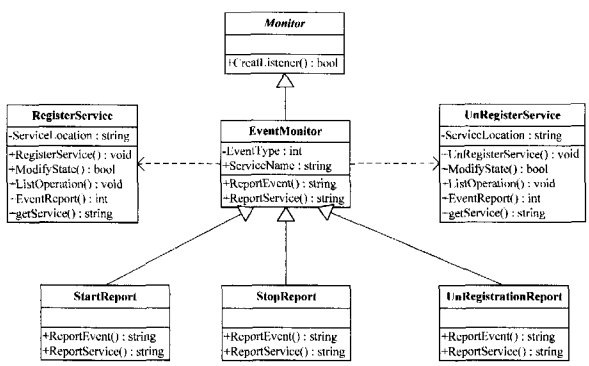

图 6 服务监听模块类图

服务注册/注销模块：模块用以注册/注销容器中新增的服务。经过注册的服务将会在已注册服务中对应一个唯一的记录。当服务监听模块监听到已注册服务的注销事件后，会调用服务注销模块将已注册服务注销。图 7 为服务注册与注销模块类图。RegisterService 对象与 UnRegisterService 对象

负责服务的注册与注销。

服务更新模块：该模块用以更新容器中的服务。当服务监听模块监听到已注册服务的更新事件后，会调用服务更新模块将已注册服务更新。图 8 为服务更新模块类图。ServiceUpdate 对象负责具体服务更新操作。

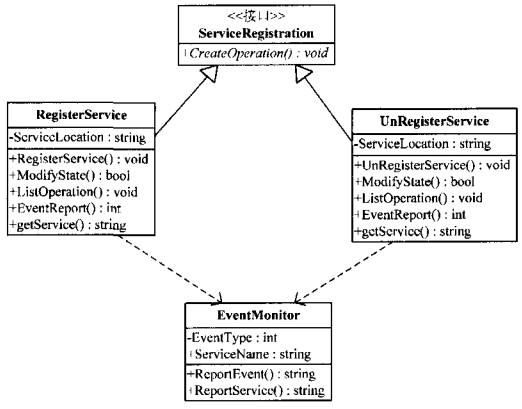

图 7　服务注册/注销模块类图

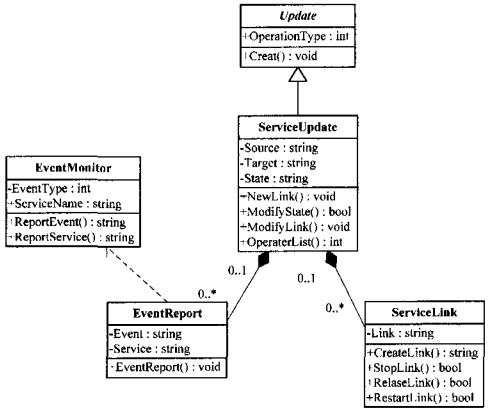

图 8　服务更新模块类图

5　DOSGi_SCA 的应用

由中南大学网络评审系统工程研究所主持开发的科技评价系统服务平台，通常包含查看项目材料、专家打分、专家投票、撰写意见、专家留言、查看评审结果、查看政策法规等功能。在这里将这些功能提取出来，分别抽象为独立的组件。其中，专家打分和专家投票往往需要灵活控制，有的时候需要专家打分，而有的时候需要专家投票，或者有时会在评审期间临时改变，例如由专家打分变为使用专家投票。这就需要系统能够在系统运行时动态地改变系统功能，基于分布式 OSGi 的 SCA 架构能够很好地满足要求。图 9 描述了整个系统架构。

在基于 DOSGi_SCA 模型的科技评价平台中，把复杂的系统通过抽象，分解成一个个可单独处理的可管理组件单元。本文实现了查看项目材料、专家打分、专家投票、撰写意见、专家留言、查看评审结果、查看政策法规等功能组件，可以将这些组件分别部署在不同的服务节点上，通过服务的形式对外提供功能。如果不需要打分组件时，就发送注销事件，服务监听模块在监听到事件后就会调用服务注销模块将已注册服务注销。这样就使软件具有很好的灵活性、扩展性和重用性。

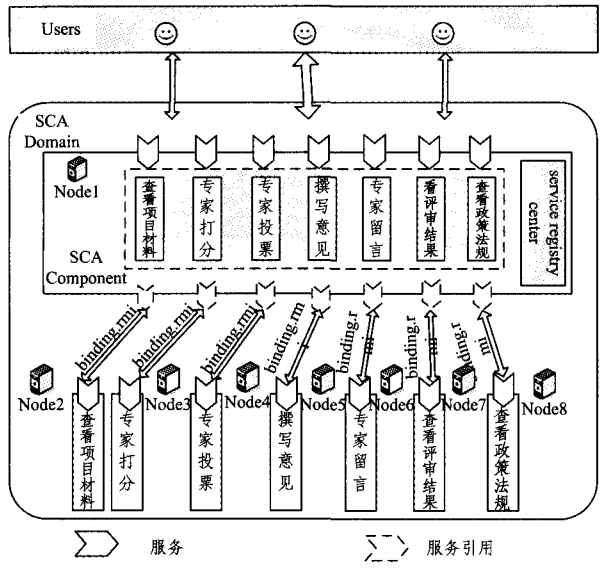

图 9　基于 DOSGi_SCA 的科技评价平台架构

结束语　在分析传统结合方式的基础之上，建立了一种基于分布式 OSGi 的 SCA 服务模型——DOSGi_SCA。DOSGi_SCA 以分布式 OSGi 为基础，构建了服务注册中心来管理本地服务和远程服务；改善了以往以 OSGi 为基础的 SCA 与 OSGi 结合不能很好地适应分布式环境的不足，很好地体现了 SCA 与 OSGi 自身的优点，使 SCA 服务引用具有动态性和灵活性，完善了 SCA 架构对 SOA 的实现，提高了 SCA 的灵活性和实用性。在后续的工作中，将近一步完善 DOSGi_SCA 模型中服务的安全性问题。作为评审工作使用的系统，安全性是必须要考虑的问题，此问题要在后续的工作中重点研究。

参 考 文 献

[1]　Krafzig D, Banke K, Slams D. Enterprise SOA[Z]. Prentice Hall PTR, 2004

[2]　Karmarkar A, Malhotra A, Booz D. Service Component Architecture(SCA) Tutorial[EB/OL]. http://www.osoa.org/download/attachments/250/SCA_OASIS_Tutorial_artl.pdf? version=1, 2007

[3]　Rui yi. Application and Research for the OSGi-based Service-Plug Oriented Architecture[D]. Jiangsu University, 2008(6)

[4]　Zhu Lei. A Dynamic Loading Supported Extensional Solution of SCA Framework and its Application[D]. Hefei University of Technology, 2008(5)

[5]　OSOA Collaboration. Power Combination: SCA, OSGi and Spring [EB/OL]. http://www.osoa.org/download/attachments/250/Power_Combination_SCA_Spring_OSGi.Pdf? version=3, 2007

[6]　Penugonda S. 开发示例 SCA 应用程序[OL]. http://www.ibm.com/, 2007-01

[7]　Barack R. Service Component Architecture Java Component Implementation V1.00[EB/OL]