以功能需求为驱的资源聚合方法与实现

郭皓明 马世龙

(北京航空航天大学 计算机学院, 北京 100083)

摘 要: 针对面向服务的体系结构(SOA, Service Oriented Architecture)环境中资源聚合的特点,提出以功能需求为驱动的资源聚合机制,在这一机制中,资源不再是静态的组件对象,应用流程与任务模型实现了分离,任务所需的功能成为关注重点与组织对象. 聚合机制由3层构成:通过底层服务信息集合汇聚满足需求的资源服务;通过中间层的功能需求描述定义资源服务所具备的各种功能与属性;通过上层上下文环境满足任务执行过程中资源服务之间的约束. 在模型内部,通过逻辑数据,维护异构服务接口之间的数据依赖关系. 利用该机制实现以功能需求为驱动的应用构造. 基于该机制的工作流系统 NGGWf(National Geology Grid Workflow)已应用于国家地质调查应用网格中的地质勘测.

关 键 词: 面向服务的体系结构; 资源聚合; 工作流

中图分类号: TP 393

文献标识码: A **文 章 编 号**: 1001-5965(2008)05-0576-04

Functional-driven resource aggregation: method and its implementing

Guo Haoming Ma Shilong

(School of Computer Science and Technology, Beijing University of Aeronautics and Astronautics, Beijing 100083, China)

Abstract: A dynamic and non-persistent mapping resource aggregation mechanism was proposed, through which functional-driven workflow system was formed in SOA (service oriented architecture) environment for application aggregation requirement. In the mechanism, services in application model were not static component. Application process model and task were separated and functions needed by application were concerned. The mechanism consisted of three layers: In the bottom layer, service information collection was created to aggregate all services of same functionality needed by application; In the middle layer, function attributes of services was defined; In the top layer, application context was created to realize constrains between services and binding conditions while implementing task. In the model, logic data was adopted to maintain data relation between services for heterogeneous service interfaces. Through the mechanism, application was composed by functions needed. A workflow system based on the mechanism, NGGWf (national geology grid workflow), has been applied in geologic applications in NGG (national geology grid) environment.

Key words: service oriented architecture; service aggregation; workflow

1 SOA环境中资源聚合需求

基于面向服务的体系结构(SOA, Service Oriented Architecture)的分布式环境为应用提供一站式的资源访问与组织服务. 其应用与资源具有以下几个特点: ①功能实现接口的不一致性:不同组织对相同功能的服务提供不同的接口; ②应用内部服务之间存在依赖性:在一个应用任务的执行过程中,资源服务之间存在了依赖性与关联性; ③应用流程的确定性:应用流程是对一个应用执行过程的逻辑抽象,在这一抽象过程中将应用过程分解为若干步骤[1-2]; ④应用任务的不确定

收稿日期: 2007-04-20

基金项目: 国家863基金资助项目(2002AA104220)

作者简介: 郭皓明(1978-),男,河北石家庄人,博士生, guohm@nlsde.buaa.edu.cn.

性:在应用的执行过程中,应用任务所需要访问的服务是不确定的.应用根据使用者提交的内容将流程中各个步骤的内容实例化,绑定对应的不同资源服务调用执行.

从适应性角度出发,应用需要实现应用流程模型与应用任务的分离.流程模型实现应用所需功能定义与基本步骤的组织,在执行过程中根据使用者的需求将流程模型转化为不同的应用任务,调用不同服务.在这一过程中,任务所需的功能成为流程模型关注的重点.由于应用任务的不确定性,造成应用任务执行中功能需求转化为服务不确定性.一个功能需求转化为服务的映射约束条件受应用任务需求与其他服务调用的影响,因此,在应用模型中无法以静态的方式建立功能需求的转化条件以及调用接口之间数据关系的持久描述.在以功能需求为驱动的资源聚合过程中,应用模型中仅仅对功能需求描述的基本构成以及转化的方式进行定义.根据这一转化方式的定义,在不同的任务中,动态的建立正确的服务转化与映射关系[3].

在应用任务的执行过程中,逻辑上服务组件之间存在一定联系与约束,一个服务组件的执行会影响另一个服务组件的具体需求与绑定约束.在这一模式中一组满足特定属性约束的服务组件集合的逻辑表示构成一个功能需求[4-5],通过对功能需求的组织实现应用流程的构造.在应用内部,功能需求之间形成应用上下文环境,通过上下文环境规定了应用任务执行过程中一个功能需求的某类属性,这类属性影响该功能需求转化为抽象服务所需的约束信息.应用内部上下文环境是一个动态的非持久的约束集.

结合应用的特点,对功能需求进行抽象,通过应用内部上下文环境对功能需求转化方式建立动态的非持久的约束机制是实现以功能需求为驱动的应用构造的基础.

2　以功能需求为驱动的聚合机制

在应用模型过程中,流程由一系列活动构成,每一个活动表示应用所需的逻辑步骤,活动由一组功能需求构成[6-7].在一个活动步骤中,通过属性描述服务所具备的功能以及流程对服务功能的需求,在多个功能需求之间建立上下文环境.上下文环境对功能需求转化为服务的方式进行描述,在任务执行过程中根据这一描述动态地实现需求转化与服务的连接和绑定.另一方面,通过逻辑数据在功能需求之间建立服务调用的数据依赖关系.图 1中展示了这一基本机制的框架,在流程的活动中,通过属性定义描述服务功能,利用属性操作建立应用上下文环境,通过逻辑抽象数据在功能需求之间建立数据依赖关系.

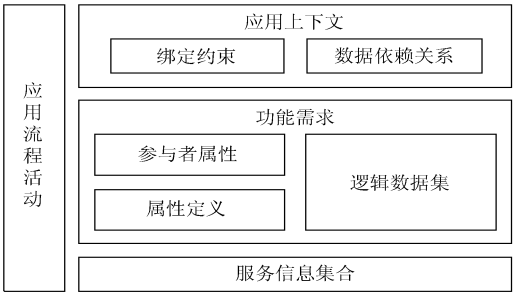


图 1　资源聚合框架

2.1　服务信息集合

服务信息集合是服务聚合的基础,其由一组服务信息构成:

$$SSet = \{ s_i \mid i = 1, 2, \cdots, n\}$$

其中 s_i 为服务信息.在服务信息集合中,服务信息由一个四元组构成:

$$s_i = \{ ID, add, D, SP\}$$

其中, ID为服务的标识; add为服务的访问地址信息,在执行过程中,根据 add的信息对服务实现远程访问; D 为服务调用数据信息集合, $D = \{ data_i \mid i = 1, 2, \cdots, n \}$, $data_i = \{ d, ld\}$其中 d 为服务接口数据, ld为其对应的逻辑表达; SP为服务的功能描述集合.服务功能描述集合由一组属性构成: $SP = \{ sp_j \mid j = 1, 2, \cdots, m \}$; 属性定义由一个三元组构成: $sp_j = (ID, domain, value)$. 其中, ID为属性的标识; domain为属性的定义域; value为属性的取值.

如前所述,一个属性用来对服务的某一功能进行表达,属性的定义域对该属性所表达的功能进行定义,属性的取值对该功能的实现进行约束.

根据属性定义,在聚合模型中,实现服务功能需求之间约束描述.属性不但用于对功能需求的描述,同时属性操作还用于逻辑数据之间约束关系的描述.属性操作包括赋值、比较 2种操作.当 2个属性 sp_1 与 sp_2 具有相同的定义域,可以对这 2个属性进行比较操作时:

$$cmpP(sp_1, sp_2) =$$

$$\{ sp_1.value \cap sp_2.value \equiv \varnothing \rightarrow sp_1 \neq sp_2;\ sp_1.value \cap sp_2.value \equiv sp_1 \rightarrow sp_1 \subseteq sp_2;\ sp_1.value \cap sp_2.value \equiv sp_2 \rightarrow sp_2 \subseteq sp_1;\ sp_1.value \cap sp_2.value \equiv sp_1 \,\&\&\, sp_1.value \cap sp_2.value \equiv sp_2 \rightarrow sp_1 \equiv sp_2;\ sp_1 \in s_1,\ sp_2 \in s_2;\ sp_1.domain \equiv sp_2.domain \}$$

通过属性的比较操作,判断2个服务 s_1 与 s_2 在该功能上是否具有一致性.

当2个属性 sp_1 与 sp_2 具有相同的定义域时,对属性的赋值操作有:

$$setP(sp_1, sp_2) = \{ sp_1.value = p_2.value, sp_1 \in s_1, sp_2 \in s_2; sp_1.domain \equiv sp_2.domain \}$$

在流程执行过程中,通过定义赋值操作,服务 s_1 的功能属性 sp_1 被传递给服务 s_2.

2.2 功能需求与服务转化

功能需求是应用流程所需服务功能的基本逻辑描述单元.功能需求的定义由一个三元组构成:

$$req = (ID, provider, dSet)$$

其中 ID为功能需求的标识;provider为参与者属性需求定义集合,其由参与者定义构成;dSet为数据集定义集合.

在功能需求定义中,通过参与者的定义对所需服务功能进行约束;在应用任务执行过程中,根据参与者属性定义内容在服务信息集合中发现满足需求的服务;数据集合则实现对功能需求所提供的入口数据、结果数据以及异常消息的定义.

参与者是功能需求中对服务属性的约束集,其由一组属性定义构成,其形式如下:

$$provider = \{p_i \mid i = 1, 2, \cdots, n\}$$

其中,p_i 为属性定义.在功能需求向具体服务的转化过程中,系统通过服务信息集合信息与参与者属性定义的匹配发现满足需求的服务集合.

$$findS(provider, SSet) = \left\{ s_k \middle| \sum_{i} \sum_{j} (cmpP(p_i, sp_j) \equiv p_i \subseteq sp_j), p_i \in provider, sp_j \in s_k, s \in SSet \right\}$$

功能需求转化为服务后就具有具体的调用参数、结果数据以及异常信息.功能需求是一组具有相似性服务的逻辑抽象,因此功能需求中抽象数据集合是这组相似服务相关数据最大集的逻辑表示.一个功能需求所对应的一组服务通常具有不同的调用接口和接口数据,这些接口之间,各类数据通过逻辑抽象在功能需求中建立统一的表达.

功能需求中抽象数据定义集合由一个三元组构成:

$$dSet = (input, output, fault)$$

其中,input为功能需求入口逻辑参数定义集合,$input = \{ld_i \mid i = 1, 2, \cdots, n\}$; output为功能需求出口逻辑参数定义集合,$output = \{ld_j \mid j = 1, 2, \cdots, m\}$; fault为功能需求异常逻辑信息定义集合.

逻辑数据定义由一个二元组构成:

$$ld = (ID, type)$$

其中,ID为逻辑数据的标识,该标识在功能需求中是唯一的;type为逻辑数据的类型.

服务信息集中的服务通过服务调用数据信息集中通过引用 ld的 ID标识实现服务接口数据与功能需求中逻辑数据的映射.

2.3 上下文环境与服务动态连接

应用上下文环境中对功能需求之间的关联与映射关系进行描述,同时建立功能需求之间的逻辑数据依赖关系.一个上下文环境由四元组构成:

$$context = (source, target, reqCons, dataLink)$$

其中,source为前驱功能需求;target为后继功能需求;reqCons为功能需求关联映射操作集合定义;dataLink为功能需求之间逻辑数据依赖关系定义.

在上下文环境中,一个功能需求的服务绑定操作由另一个功能需求某一属性或结果数据决定时,这2个功能需求就构成一个内部逻辑约束关系.该关系中,被约束的功能服务称为后继 target,约束其它服务绑定的功能需求称为前驱 source.上下文环境中,一个服务的最终执行需要经历外部绑定与内部连接2个过程.外部绑定指根据功能需求属性描述在服务信息集中发现满足需要的服务;内部连接指在上下文环境内部,前驱服务与后继服务之间的顺序执行操作.这一过程如下:

$$linkS(s_i, s_j) = s_i; s_j$$

$$s_i \in findS(source.provider, SSet)$$

$$s_j \in findS(target.provider, SSet)$$

在上下文环境中,2个功能需求通过关联与映射操作实现内部逻辑关系的建立,通过该内部逻辑关系实现前驱与后继功能需求之间连接约束操作.

在应用任务执行过程中,通常一个服务的调用会导致另一个服务的调用.在上下文环境中将这一约束称为映射 (Mapping)操作.通过映射,前驱需求影响后继需求绑定条件.映射操作由属性赋值操作构成:

$$map(p_1, p_2) = \{ setP(p_1, p_2), p_2 \in target, p_1 \in source \}$$

在流程执行过程中,一个功能需求会根据需要转化为多个具体服务执行.前驱与后继功能需求同时转化为多个服务时需要通过关联操作实现前驱需求服务与后继需求服务一对一的内部绑定.关联操作由属性取值比较操作构成:

$$realte(p_1, p_2) = \{ r \mid p_1 \cap p_2 \equiv p_2 \rightarrow r = true; p_1 \cap p_2 \neq p_2 \rightarrow r = false, p_2 \in target, p_1 \in source \}$$

映射与关联操作定义了前驱需求与后继需求

绑定操作的影响,在上下文环境中,处于前驱的需求绑定操作不能由后继需求的属性约束以免造成死锁. 在建立 2个服务连接关系过程中,上下文环境首先根据定义对后继需求的有关属性赋值,而后在服务信息集合中发现全部满足功能需求服务. 最后根据比较操作,通过属性取值,将前驱功能需求中的服务与后继功能需求中的对应服务之间建立正确的连接关系. 在这一过程中,服务连接关系有:

$\sum_i$ map (tp_i, sp_i); findS (target provider, SSet); $\sum$ realte (tp_i, sp_i); linkS (s_i, s_j), $tp_i \in$ target provider, $sp_i \in$ source provider, $s_i \in$ findS (source provider, SSet); $s_j \in$ findS (source provider, SSet)

2.4　非持久的数据依赖关系的转化操作

上下文环境中通过逻辑数据依赖关系定义来实现功能需求之间数据依赖关系的描述以及服务之间数据连接关系. 在功能需求定义中通过 dSet 建立逻辑数据描述,在上下文环境中,对前驱与后继需求的逻辑数据之间建立依赖关系:

$$dataLink = \{ link_i \mid i = 1, 2, \cdots, n \}$$

一个依赖关系由一个三元组构成:

$$link_i = (ID, sourceData, targetData)$$

其中, ID为依赖关系的标识; sourceData为数据来源,该来源可以为前驱功能需求中输出数据 ld_j, $ld \in$ source dSet output; targetData为连接目标,为后继功能需求中逻辑输入数据 ld_i, $ld \in$ target dSet input.

在服务转化过程中,功能需求之间的逻辑数据依赖关系通过映射转化为服务之间数据依赖关系. 在上下文环境中,逻辑数据之间形成数据依赖关系,后继功能需求完成服务转化后,需要根据依赖关系获取对应的调用数据. 在这一过程中,后继服务根据功能需求中逻辑数据定义以及其与前驱功能需求逻辑数据的依赖关系获取前驱功能需求中对应的逻辑数据. 这一过程如下:

implLink (source, target, dataLink) = { $\sum$ (td = sd), td ∈ target dSet input; td ≡ link. targetData; sd ∈ surce. dSet output; sd ≡ link. sourceData; link ∈ dataLink }

由于同一个功能需求可能转化为若干个不同接口的服务,因此在完成功能需求之间数据连接后,还要在处于前驱与后继功能需求中同属性的服务之间建立连接关系.

implLinkS (ts, ss) = { $\sum$ td = sd, td ∈ $link_i$. targetData; sd ∈ $link_i$. sourceData; td ∈ ts. input; ts ∈ findS (target provider, SSet); sd ∈ ss. output; ss ∈ findS (source provider, SSet); ss. SP ≡ ts. SP }

一个服务只有完成全部数据连接后方可实现调用执行,当一个数据连接发生错误时,该服务为不可调用.

3　在 NGG中的应用

在国家地质调查应用网格 (NGG, National Geology Grid)中,以功能需求为驱动的服务聚合机制满足了地质行业应用中服务动态聚合的要求,并由此形成工作流中间件系统 NGGWf (National Geology Grid Workflow),能有效满足地质调查行业中服务动态聚合的需求.

4　结 束 语

传统的工作流系统不能有效满足应用流程模型中资源动态聚合的需求. 本文提出的动态非持久映射的资源聚合机制,以功能需求做为构造流程模型的基本对象,满足资源动态聚合的需求.

参考文献 (References)

[1] Zhao W, Hu W H, Zhang S K, et al. Study and application of a workflow meta-model[J]. Journal of Software, 2003, 14(6): 1052 - 1059

[2] 范玉顺. 工作流管理技术基础[M]. 北京:清华大学出版社, 2002: 31 - 45
Fan Yushun. Fundamentals of workflow management technology [M]. Beijing: Tsinghua University Press, 2002: 31 - 45 (in Chinese)

[3] 孙瑞志, 史美林. 支持工作流动态变化的过程元模型[J]. 软件学报, 2003, 14(1): 62 - 67
Sun Ruizhi, Shi Meilin. A process meta-model supporting dynamic change of workflow [J]. Journal of Software, 2003, 14(1): 62 - 67 (in Chinese)

[4] Dong Jing, Sun Yongtao, Yang Sheng, et al. Dynamic web service composition based on OWL-S[J]. Science in China Series F-Information Sciences, 2006, 49(6): 843 - 863

[5] 张晓刚, 李明树. 基于工作流的知识流建模与控制[J]. 软件学报, 2005, 16(2): 184 - 193
Zhang Xiaogang, Li Mingshu. Workflow-based knowledge flow modeling and control [J]. Journal of Software, 2005, 16(2): 184 - 193 (in Chinese)

[6] 李红臣, 史美林. 工作流模型及其形式化描述[J]. 计算机学报, 2003, 11: 1456 - 1463
Li Hongchen, Shi Meilin. Workflow models and their formal descriptions[J]. Chinese Journal of Computers, 2003, 11: 1456 - 1463 (in Chinese)

[7] Hollingsworth D. The workflow reference model [M]. UK: Workflow Management Coalition, 1995: 72 - 110

(责任编辑:吴　琼)