一种基于动态语境组装的分布式构件框架的设计与实现

谢德平,邢 阳,马晓星,曹 春,吕 建
(南京大学计算机软件新技术国家重点实验室,南京大学计算机软件研究所,江苏南京 210093)

摘 要: 提出一种新的分布式构件框架 ACF,在传统的以共享协同信息为主的语境机制基础上,引入了共享协同对象的动态语境机制,直接支持基于运行时软件体系结构的协同逻辑的表达和动态演化. 该构件框架已在集成开发环境 Artemis Studio 中得到初步实现和应用.

关键词: 分布式构件; 基于语境的组装; 协同; 动态演化
中图分类号: TP311 **文献标识码**: A **文章编号**: 0372-2112 (2009) 4A-057-08

Design and Implementation of A Dynamic Contextual Composition Based Distributed Component Framework

XIE De-ping, XING Yang, MA Xiao-xing, CAO Chun, LV Jian
(*State Key Laboratory for Novel Software Technology, Institute of Computer Software, Nanjing University, Nanjing, Jiangsu* 210093, *China*)

Abstract: We propose a novel distributed component framework named ACF, which introduce a dynamic contextual composition mechanism based on shared coordination object, contrastively based on shared coordination data traditionally. ACF directly supports explicit expression and dynamic evolution of coordination logic, by applying runtime software architecture. At present, ACF has been preliminarily implemented and employed in an integrated development environment (IDE) name Artemis Studio.

Key words: distributed component; contextual composition; coordination; dynamic evolution

1 引言

软件构件技术,特别是分布式构件技术[1]的发展,大大简化了大型的、企业级的软件系统的开发. 分布式构件技术为网络环境中的应用系统提供了远程通信功能的支持;另外,分布式构件技术还提供了资源池、事务、安全等基础设施,使得人们能够开发出伸缩性好、运行可靠及安全的应用系统. 当前主流的分布式构件技术包括 CORBA[2]、EJB[3]、DCOM[4]、Web Service[5]等. 但是,随着 Internet 技术的出现和普及,这些分布式构件技术还不能完全满足 Internet 计算环境的需求.

在 Internet 计算环境下,软件系统的开发更多地体现为组合基础软件资源[6]. 我们考虑的重点不是如何开发软件资源,而是如何组合、协同各个软件资源. 换句话说,协同逻辑是 Internet 计算环境下软件开发的重点. 考察传统的分布式构件技术,它们的着眼点在于构件业务功能的开发,对构件之间协同逻辑的考虑却比较少. Internet 计算环境要求,能够根据环境的具体情况和应用的特殊需求,对构件的协同逻辑进行一定的设计和定制,而传统分布式构件的交互模式是单一、僵硬的;传统分布式构件之间是一种静态绑定的关系,难以在运行时动态调整交互关系,而 Internet 计算环境"动态、多变"的特点,要求能够在运行时动态调整软件系统的结构[7].

针对上述问题,我们提出了一种新的分布式构件框架——ACF(Artemis Component Framework). ACF 遵循"基于语境的组装(Contextual Composition)"[8]的思想,将以共享传递协同信息为主的静态语境扩展为支持具有动态行为的共享对象的动态语境,以此显式表达 Artemis 构件的协同逻辑,为它们提供协同服务;同时,该共享对象解耦了 Artemis 构件之间的绑定关系,并且承载了演化行为,从而实现了协同逻辑的动态演化. 为了支撑基于 ACF 的应用开发与运行,我们设计和开发了 ACF 的实现 Artemis Server,及可视化的构件集成开发环境 Artemis Studio.

收稿日期:2008-10-22;修回日期:2008-12-01
基金项目:国家自然科学基金(No. 60736015);国家 863 计划(No. 2007AA01Z178);新世纪优秀人才支持计划(No. NCET-07-0419);江苏省六大人才高峰项目

2 基于动态语境的组装方式

2.1 构件的组装方式

构件是一种组装单元,它具有规范的接口规约和显式的语境依赖.构件可以被独立地部署并由第三方任意地组装[8].构件模型定义了构件的组装标准,主要描述了一组构件是如何组装的.目前存在的组装方式主要包括面向连接的组装、数据驱动的组装和基于语境的组装[8].Visual Basic 构件、JavaBeans 采用了面向连接的组装方式;基于消息队列的构件如 EJB 中的消息驱动 Bean,一般都采用数据驱动的组装方式;而大多数分布式构件,包括 CORBA 的 CCM、EJB(不包括消息驱动 Bean)、DCOM 等,都采用了基于语境的组装方式.

面向连接的组装是面向过程和面向对象程序设计范型中常见的组装方式.像远程过程调用(Remote Procedure Call,RPC)、对象方法调用、事件传递等,都属于面向连接的组装.这种组装方式的特点是调用者和被调用者之间总是通过建立绑定而"连接"起来.虽然间接的绑定可以带来一定的灵活性,但是我们认为这种灵活性还不够满足 Internet 计算环境的要求.数据驱动的组装的特点是,需要的数据到来时,构件才会执行相应的行为.这种组装方式很好地实现了构件之间的解耦,但是受限较多.基于数据驱动组装的构件的接口一般被定义为单向语义,即不包括返回值,发出请求的构件也不期望得到错误码或异常.

基于语境的组装是一种声明式的协同功能调用机制.开发者仅需声明构件所需的运行语境性质,而让构件框架自动调用相应服务实现相应的协同功能.在实现上就像在构件实例上建立一层外壳,即使用所谓构件容器,将构件实例包裹在里面,并截获构件实例与外界的一切通信.由于截获了与外界的通信,就有机会在构件实例运行时为其插入特定的服务.这些服务构成了构件实例的语境.同时,构件实例会收到各种形式的指向它的语境的引用,以在运行时能够主动地访问这些服务.

例如,EJB 就支持基于语境的组装方式.部署了的 Bean 被包裹在一个 EJB 容器中.EJB 容器截获了 Bean 与外界的一切交互,并根据部署描述文件的声明,自动地为 Bean 提供事务、安全等服务.这些服务构成了 Bean 的语境.从编程的角度看,Bean 的开发人员并没有编写使用这些服务的代码,只是在部署描述文件中进行了声明,而 EJB 容器在运行时自动地执行这些服务的代码.图 1 说明了 EJB 基于语境的组装方式.

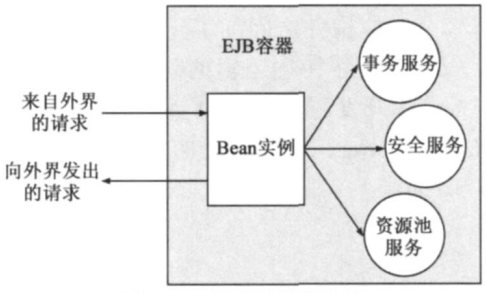


图1 EJB基于语境的组装

可见,基于语境的组装能够自动给构件增加额外的行为(即语境中的各种服务),而编写这些服务的难度往往很高.例如安全服务,一般需要相关领域的专家才能实现.现在,专家们负责在构件模型的中间件中实现这些服务,应用的开发人员只需关注业务功能,以声明的方式自动使用,或以引用的形式主动使用这些服务.这大大简化了应用的开发难度,也体现了"关注分离"[9]的软件工程原则.

2.2 动态语境

传统构件模型基于语境的组装方式,关注的重点是功能方面的服务,对协同逻辑的关注较少.像自动的事务操作、安全控制等,本质上仍然是构件功能的一部分.而在 Internet 计算环境中,应用需要丰富的协同逻辑.随着协同逻辑复杂性的提高,构件也需要一些基础的服务,来帮助它与其它构件交互.这些服务可能包括通信、协调、转换和一些辅助功能[10].另外,传统构件的语境是一组没有行为的共享数据,难以满足开放协同环境中动态演化的需求.为此,我们提出了 ACF,引入了具有动态行为的共享对象,作为 Artemis 构件的动态语境,为 Artemis 构件提供支持动态演化的协同服务.

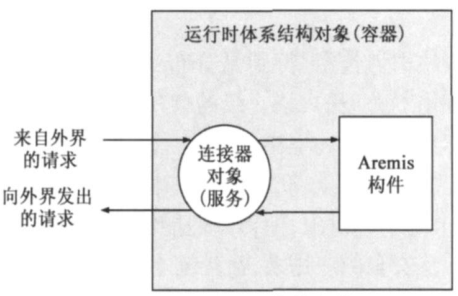


图2 Artemis基于语境的组装

我们在前期工作[11]提出了一种基于运行时体系结构的动态协同架构,说明了运行时体系结构在协同逻辑动态演化中的应用.在本文中,我们使用软件体系结构描述 Artemis 构件之间的协同逻辑:体系结构的拓扑描述了构件之间的交互关系,即"谁跟谁交互";连接器描述了具体的交互机制,即"怎么交互".我们将软件体系结构信息具体化为一个有行为的对象实体,称之为运行时体系结构对象.如图 2 所示,运行时体系结构对象作为 Artemis 构件的动态语境,截获了 Artemis 构件与其它构件的所有交互,根据其内置的体系结构信息,对构件之间的交互关系做出解释;运行时体系结构对象

包含了连接器对象,连接器对象承载了是交互机制方面的服务,运行时体系结构对象在 Artemis 构件运行时为其自动插入这些服务;同时,对运行时体系结构对象的修改会造成交互行为的重新解释,因此,我们可以采用修改和升级运行时体系结构对象的方式,实现协同逻辑的动态演化.

2.3　运行时体系结构对象

软件体系结构一般是指该系统的构件组成,构件之间的相互关系和连接方式,所形成的系统结构配置,及其动机、模式和约束等. 它本身原本是一个抽象的规约,最终分散而隐含的体现在系统各个构件及其间的交互行为中[12]. 可见,用软件体系结构来描述构件之间的协同逻辑,是一种合理和恰当的方法. 软件体系结构从全局的视角刻画了系统的配置状态,从而能够成为动态演化的重要依据. 运行时刻,体系结构约束的破坏,往往是动态演化的驱动因素;而对协同逻辑的动态调整可以在体系结构的抽象层面进行表达.

为了使体系结构层面抽象表达的动态演化能够在具体系统中得以实施,我们将软件体系结构具体化为一个有具体行为的、运行时可以操纵的运行时体系结构对象. 与一般的对象一样,运行时体系结构对象具有状态和行为. 它的状态就是软件体系结构信息,包括构件、连接器以及它们之间的连接关系. 它的行为包括两方面:对交互关系的解释,例如为发出请求的构件查找目标构件,并转发请求;对协同逻辑的修改,例如构件的增加、删除、替换,连接器的修改等.

运行时体系结构对象是 Artemis 构件交互机制的核心. 如图 3 所示,运行时体系结构对象封装了体系结构信息,知道 Artemis 构件之间的交互关系. Artemis 构件总是将请求交给运行时体系结构对象,运行时体系结构对象通过查询内置的体系结构信息,得知 Artemis 构件请求的目标构件,进而将请求转发给目标构件.

运行时体系对象的作用类似于 CORBA 中的 ORB、EJB 中的业务接口代理. ORB 和业务接口代理都实现了构件之间的远程通信和请求转发,但是都存在交互关系过于僵硬和交互机制不够丰富的问题. 而 Artemis 构件能够很好地克服这两个问题:

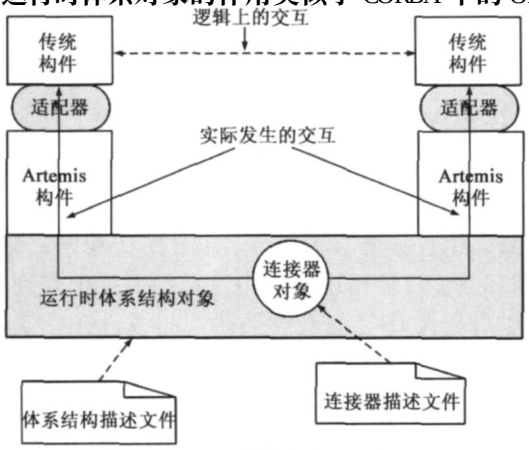


图3　Artemis构件的交互机制

(1) 运行时体系结构对象通过内置体系结构信息,实现了 Artemis 构件之间关联关系的松耦合.

(2) 运行时体系结构对象通过修改内置的体系结构信息,可以实现交互关系的动态演化.

2.4　连接器对象

在软件体系结构中,交互机制则集中体现为连接器的内在逻辑. 这意味着,我们在设计软件体系结构时,要提升连接器的地位,将其作为与构件同等重要的"一等公民". 在 ACF 中,我们将连接器作为运行时体系结构对象的一部分,参与系统的运行,在运行过程中为构件提供交互机制方面的服务.

随着交互机制复杂程度的提高,相应连接器对象的实现难度也随之增加. 我们研究发现,很多应用中的交互机制存在共同部分,例如 RPC、消息传递、接口适配、特定的加密算法、特定的压缩算法等. 这些共同的交互机制实现是可以重用的. 因此,我们应该将交互机制模块化,尽可能复用已有的交互机制实现.

为此,我们提出了一种基于连接器组合的交互机制实现方案. 我们的思路是:

(1) 归纳出若干简单的、通用的交互机制,分别实现为简单连接器.

(2) 将复杂的交互机制分解成若干简单的交互机制,这些交互机制已经被实现为简单连接器. 通过组合简单连接器,来生成复杂的连接器.

(3) 连接器的功能采用拦截器模式实现,连接器的组合可以通过连接拦截器来实现.

开发人员在设计软件体系结构的时候,需要对连接器进行分析. 如果某个连接器的交互逻辑比较复杂,则应该对其分解,并通过连接器组合图来描述连接器的组合情况.

3　ACF 的编程模型

ACF 的编程模型描述的是如何使用 Artemis 构件技术及其支撑平台和工具开发应用系统. 在 ACF 的编程模型下,构件的编程不是重点,重点是如何组合构件,使它们协同工作以构成一个完整的应用系统.

在 Internet 中,有许多软件资源作为服务可第三方使用,其中不乏基于传统构件技术开发的构件. Artemis 构件编程模型的目标之一,就是将这些传统构件集成在一起,让它们协同工作. 需要注意的是,我们不对已有的传统构件做任何修改. 这些传统构件只需要与某种形式公布出来,使得构件集成者能够发现这些构件.

首先,构件集成者根据应用的功能性需求和非功能性需求,设计软件体系结构图,确定需要哪些功能的 Artemis 构件,以及它们之间的协同逻辑;然后,构件集成者在 Internet 上搜索符合要求的传统构件,将它们的相关信息,如网络地址、访问接口等,绑定到体系结构图中的 Artemis 构件上,并将体系结构图转换成体系结构描述文件;接着,构件集成者对连接器进行设计,为每个连接器关联相应的拦截器类,必要时进行连接器组合,最终得到连接器描述文件;最后,构件集成者将体系结构描述文件和连接器描述文件发送给传统构件所在的各个节点,这些节点上运行的 Artemis Server 开始部署,从而创建运行时体系结构对象和连接器对象. 至此,Artemis 构件及其语境都被创建好,Artemis 构件之间可以交互、协同,整个应用可以运行.

3.1 Artemis 构件的基本形态

前面提到,ACF 使得异构的传统构件之间能够互操作. 异构的传统构件的基本形态是不同的,这为实现互操作带来了一定的难度. 我们抽象出传统构件基本形态的共性,将其作为 Artemis 构件的基本形态,如图 4 所示.

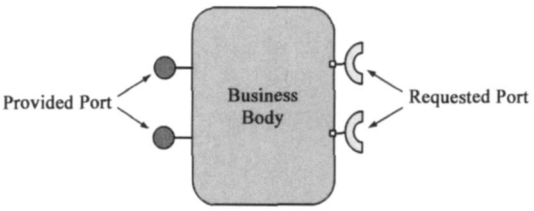


图4 Artemis构件的基本形态

Artemis 构件并不执行业务功能. 真正的业务功能由 EJB、CORBA、Web Service 等传统构件完成,Artemis 构件只是它们的一个逻辑代表,解决它们之间的互操作问题. 如图 3 所示,适配器在 Artemis 构件和传统构件之间,为两种不同的接口形式提供转换服务. 我们为每种传统构件提供了相应的适配器. 通过增加新类型的适配器,我们可以很容易地增加对新构件的支持.

Artemis 构件拥有两种类型的接口:提供端口和请求端口. 提供端口表示 Artemis 构件提供的功能,请求端口表明 Artemis 构件需要依赖其它构件. 通过连接提供端口和请求端口,可以将 Artemis 构件组装在一起. 图 5 表示了 Artemis 构件的一个组装场景.

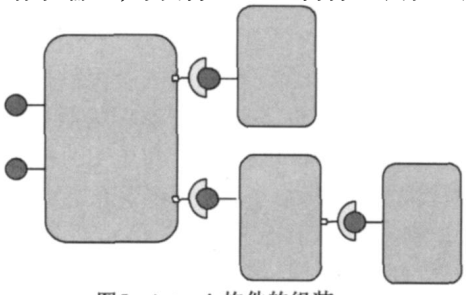


图5 Artemis构件的组装

3.2 体系结构描述文件

体系结构描述文件作为 Artemis 构件可部署单元的一部分,包含了体系结构相关的信息. 这些信息分为两类,第一类是构件之间的拓扑信息,即连接关系;第二类是构件的属性信息,前面提到的"绑定"传统构件,其实就是将传统构件的相关信息存放到构件的属性中.

体系结构描述语言(Architecture Description Language,ADL)是一种建立体系结构的规范方法. 因此,我们采用 ADL 来编写体系结构描述文件. ADL 提供一种针对图形的文本形式的句法表达,有比较严格的句法和语法. 软件集成者可以直接使用 ADL 来编写软件体系结构,也可以借助于转换工具,将软件体系结构图直接转换成 ADL 描述文件.

我们选择 Acme[13] 作为 ACF 的 ADL,基于以下三个理由:

(1) Acme 的设计借鉴了其他 ADL 的经验,语言简洁,语法简单,可扩展以支持复杂的体系结构.

(2) Acme 提供了一个可视化设计环境 Acme Studio,直接支持设计软件体系结构图,并能将其转换 Acme ADL 描述文件.

(3) Acme 提供了一套类库 Acme Lib,为每种体系结构元素如构件、连接器等提供了 Java 封装类,并提供了解析 Acme ADL 描述文件的帮助类. 这为我们开发 ACF 实现,提供了很大帮助. 如果使用 Acme ADL 来描述 C/S 应用的软件体系结构,则如图 6 所示.

```
System CSSystem = {
    Component Client  = {
        Port sendRequest  = {
        }
    }
    Component Server  = {
        Port receiveRequest  = {
        }
    }
    Connector CSConnector  = {
        Role clientSide  = {
        }
        Role serverSide  = {
        }
    }
    Attachment Client.sendRequest to CSConnector.clientSide;
    Attachment Server.receiveRequest to CSConnector.serverSide;
}
```

图6 Acme描述的C/S体系结构

3.3 连接器描述文件

连接器描述文件作为 Artemis 构件可部署单元的一部分,包含了连接器对象的相关信息. 前面提到,连接器对象采用拦截器模式实现,其主要功能由拦截器完成,需要在连接器描述文件中指定拦截器的实现类. 连接器组合机制其实是通过拦截器的成链操作实现的. 因此,复杂连接器的功能由它的拦截器链完成. 如果该复杂连接器具有多路复用(Multiplexer)功能,则它的功能由拦截器树完成. 为了便于编写和分析连接器描述

文件,我们采用 XML 作为连接器描述文件的编写语言. 如图 7 所示,它描述了一个具有数字摘要验证和远程同步方法调用功能的复合连接器.

```
<?xml version="1.0" encoding="UTF-8"?>
<connectors>
  <connector name="Connector0">
    <interceptor-tree name="Caller">
      <interceptor
        className="SynMethodInterceptor"/>
      <interceptor
        className="DigestClientInterceptor"/>
      <interceptor
        className="InvokerInterceptor"/>
    </interceptor-tree>

    <interceptor-tree name="Callee">
      <interceptor
        className="DigestServerInterceptor"/>
    </interceptor-tree>
  </connector>
</connectors>
```

图7　连接器描述文件

4　ACF 的原型实现

为了兼容 EJB、Web Service 主流软件构件和软件服务技术,我们基于工业界广泛采用的开源软件 JBoss Application 和 Apache Axis 开发了一个 ACF 的原型实现——Artemis Server,提供了对基于动态语境组装机制的支持,从而支持灵活的协同逻辑和协同逻辑的动态演化. 另外,我们还提供了一个基于 Eclipse 通用开发平台的、支持构件集成的开发环境——Artemis Studio. Artemis Server 与 Artemis Studio 一起,为 Internet 计算环境中基于构件集成的软件开发,提供了一套方法和支撑软件.

4.1　Artemis Server 的系统实现

图 8 描述了 Artemis Server 的总体结构,主要包括远程通信模块、体系结构部署模块、连接器部署模块、动态演化模块、运行时体系结构对象管理模块和本地构件库管理模块.

远程通信模块提供了 Artemis Server 与外界通信的功能. Artemis Server 需要与 Artemis Studio 进行通信,以接收协同逻辑和演化命令;同时,Artemis Server 之间也需要相互通信,以转发 Artemis 构件发出的请求和传递执行结果.

体系结构部署模块和连接器部署模块的主要功能是接收体系结构描述文件和连接器描述文件,创建运行时体系结构对象. 构件集成者在设计体系结构的时候,可以对运行时体系结构对象类进行定制,为其引入特定的行为,例如预制的演化行为. 然后在体系结构描述文件中指定自定义的运行时体系结构对象类;体系结构模块使用反动态类加载技术和反射机制,来创建定制的运行时体系结构对象. 由于一个应用总是包括多个节点,因此体系结构的部署是一个分布式的过程. 任何一个节点的部署失败,都导致整个体系结构的部署失败. 因此,我们采用了事务机制,保证部署过程的原子性.

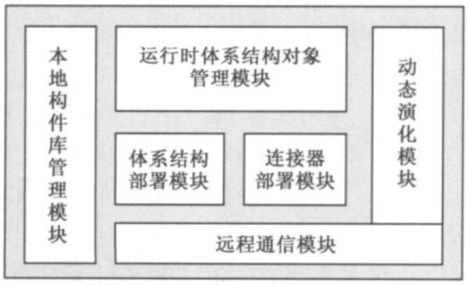


图8　Artemis Server的总体结构

动态演化模块的主要功能是接收演化命令,执行演化操作. 我们采用了命令(Command)模式和组合(Composite)模式,将构件集成者对体系结构图的修改封装成一个(组)演化命令对象. 而具体的演化行为,包含在运行时体系对象中. 动态演化模块根据接收到的演化命令,并对之进行分析,然后调用运行时体系结构对象的相应方法,完成运行时体系结构对象的升级,从而实现整个应用的协同逻辑的演化. 同样,动态演化也是一个分布式的过程,需要保证原子性和一致性. 我们利用两阶段提交机制,来实现原子性;利用锁定机制,保证系统的一致性,使得系统在演化的过程中,不会给用户错误响应.

运行时体系结构对象管理模块提供了对所有运行时体系结构对象的管理和维护功能. 由于每个节点可以参与到多个应用中,所以一个 Artemis Server 中可能同时存在多个运行时体系结构对象. 当一个体系结构被部署后,相应的运行时体系结构对象添加到管理模块中;当一个体系结构被卸载时,管理模块将删除相应的运行时体系结构对象.

本地构件库管理模块提供了对 Artemis Server 所在节点上传统构件的管理和维护功能. 传统构件被存放在本地构件库中,体系结构部署模块会根据体系结构描述文件中的相关信息,从本地构件库中寻找相应的传统构件,并将其部署. 图 8 Artemis Server 的总体结构

4.2　Artemis Studio 的功能

Artemis Studio 为构件集成者提供了一个可视化的构件集成开发环境. 构件集成者以图形化的方式绘制体系结构图和连接器组合图,如图 9、10 所示. Artemis Studio 可以将它们分别转换为体系结构描述文件和连接器描述文件. 当所有这些都完成之后,构件集成者通过 Artemis Studio,将体系结构描述文件和连接器描述文件,发送到服务构件所在的各个节点上.

如果系统在运行的过程中,由于用户需求或运行环境发生了变化,构件集成者觉得有必要对系统进行动态调整,他可以通过 Artemis Studio 对体系结构对和连

接器组合图进行调整. Artemis Studio 自动捕获用户的调整操作,生成相应的演化命令,然后将演化命令发送给相应的节点. Artemis Server 接收到这些演化命令,并执行演化操作,从而实现协同逻辑的动态演化.

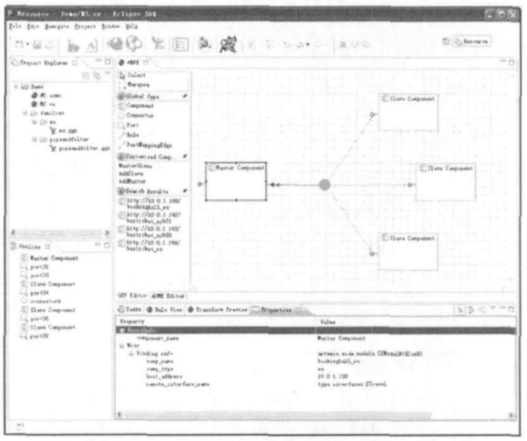


图9 Artemis Studio体系结构设计界面

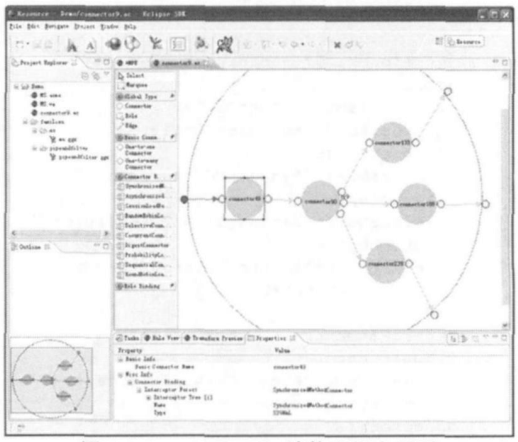


图10 Artemis Studio 连接器组合界面

4.3 应用示例

为了展示 ACF 对协同逻辑动态演化的支持,我们在该框架下开发了一个简单的电子商务应用. Trade Service 构件提供交易服务,Supplier 构件提供商品信息,消费者通过 Trade Server 购买 Supplier 提供的商品. 如图 11 所示,此场景中可能出现两种演化行为:增加 Supplier 构件,以提供更多种类的商品;增加 Trade Service 构件,以分担交易量持续增加带来的负载. 前者是同一体系风格类的演化,后者是不同体系结构风格间的演化.

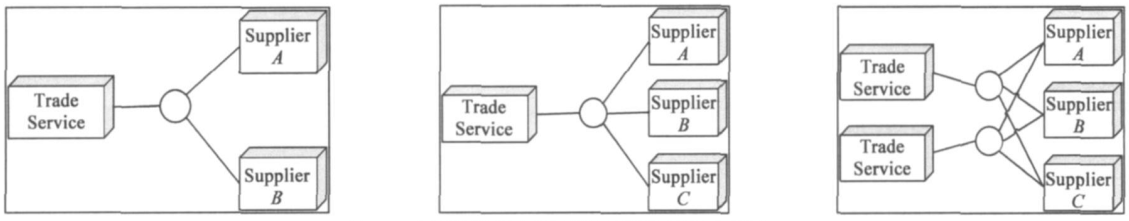


图11 电子商务应用的初始配置及其演化行为

5 相关工作

随着 Internet 和分布式计算技术的发展,分布式构件技术已经成为构建分布式应用系统的关键技术. EJB、CORBA、DCOM、Web Service 都是当前主流的分布式构件技术. 这些构件技术的关注点都在于构件本身的功能和构件之间的远程通信,而没有着重考虑构件之间复杂的协同逻辑. 目前已经有一些技术开始解决这个问题,如 BPEL4WS、WS-Coordination 和 WS-Transaction 等[14],它们研究如何组合已有的 Web Service 以构成新的服务. 但是,放到 Internet 计算环境中,这些技术还是不能完全符合需求. 例如,BPEL 按照工作流视图来组合 Web Service,组合出来的服务的结构过于平坦. 平坦的结构不利于描述复杂的协同逻辑,也限制了协同逻辑的演化方式. 本文提出的 Artemis 构件框架 ACF,以体系结构视图作为组合构件的指导,使用软件体系结构来描述协同逻辑. 软件体系结构真实地反映了组成软件的各个软件实体之间的相互关系,用它描述协同逻辑是非常恰当的. 另外,就软件体系结构的描述功能来说,学术界的研究比较成熟,有现成的理论和工具可以帮助我们实现 ACF.

在软件构件领域,基于语境的组装(Contextual Composition)是一个重要概念[8]. EJB、CORBA、DCOM 等主流构件框架中的事务、安全等通用功能都依赖于基于语境的组装,以语境的形式提供给构件使用. ACF 中的运行时体系结构对象和连接器对象,可以看作是 Artemis 构件的语境. 只不过传统构件的语境都是提供功能性的服务,而 Artemis 构件的语境提供的是协同逻辑方面的服务. 而且,传统构件都是由构件模型实现提供语境,用户很难定制;而在 ACF 中,用户可以提供自定义

经测试,ACF 能够很好地支持这两种演化行为.

在实验过程中,我们着重考察了 ACF 的性能开销和演化效率. 由于引入了动态语境,ACF 需要执行额外代码,性能开销自然要大一些. 但由于分布式系统中,构件之间通信的时间开销很大部分是网络通信带来的,因此,ACF 增加的性能开销是可以接受的. 由于运行时体系结构对象内置于软件系统之中,从系统内部调整协同关系,演化效率也还是不错的.

的连接器,通过连接器组合机制,为 Artemis 构件提供定制的语境.这样,用户拥有足够的自由度,设计出灵活、多样的协同逻辑.

近几年来,基于体系结构的动态演化方法成为软件演化的研究重点,卡内基梅隆大学的 Rainbow 项目[15]是其中具有代表性的工作之一.首先,Rainbow 项目设计出体系结构描述语言 Acme,将其作为描述软件体系结构的基本工具.这个工作在前文已经提到,不再赘述.鉴于 Acme 强大的功能和友好的用户界面,我们也使用它作为支持 Artemis 构件模型的体系结构描述语言.其次,在动态演化方面,Rainbow 使用了外置的运行时体系结构管理器,探知运行时系统的状态,必要时对系统进行动态调整.这种方法的缺点是,对体系结构的修改不能直接作用到运行中的系统内部,还需要借助其它机制,保持体系结构与真实系统之间的一致性.而 Artemis 构件模型使用了内置的运行时体系结构对象,将体系结构对象化,作为运行系统的内在部分.因此,对体系结构的修改,能够直接反映到运行系统中.

另外,文献[16]从过程(行为)的角度考虑组合服务的自适应演化,这种方法可以看作是基于软件体系结构的动态演化方法的补充.

在以往软件体系结构的研究工作中,连接器一直处于被弱化的位置.在 Internet 计算环境中,应用的协同逻辑处于一个非常重要的位置,我们不仅仅把连接器作为一个重要的设计元素,更要将其作为运行时实体,在系统运行的过程中发挥其作用.北京大学的 PKUAS 项目,借鉴了面向方面编程 AOP(Aspect Oriented Programming)思想[9],将连接器作为显式的实体,利用 Advice 实现连接器的功能,并通过元编程技术,将这些 Advice 同构件的业务功能编织在一起[17].PAUAS 项目重点解决了连接器与构件如何协同工作,而 ACF 还进一步对连接器的编程做了研究.我们提出了一种连接器组合机制,通过简单连接器的组合,生成复杂的连接器,并且为这种机制提供了图形化的支持工具.

6　总结与展望

Internet 的发展带来了新的应用模式,同时对软件的开发方法提出了新的需求.通过对 Internet 计算环境特点的分析,我们认为,Internet 计算环境中的软件开发应该体现为组合已有的软件资源.在组合的过程中,软件资源之间的协同逻辑尤为重要.当前主流的分布式构件技术已经不能完全满足 Internet 计算环境的需要.为此,本文提出了 Artemis 构件框架 ACF,引入了运行时体系结构对象和连接器对象,通过基于动态语境的组装方式,为 Artemis 构件提供了丰富的协同服务,并实现了协同逻辑的动态演化.另外,我们设计与实现了辅助 Artemis 构件开发的可视化开发环境 Artemis Studio,和支持 Artemis 构件运行的支撑中间件 Artemis Server,为 Internet 计算环境中的应用开发提供一套方法和支撑技术.

下一步,我们将继续围绕基于语境的组装,从组装推理(Composition Reasoning)[8]理论基础的角度深化我们的工作.我们将深化连接器组合机制,结合面向方面编程中侧面(Aspect)[9]的思想,从连接器分类、组合规则等角度出发,给出一个系统化的连接器组合理论和方案.

致谢:感谢 Artemis-NG 系统开发小组的其他成员:潘静,陈祥,徐胜强,吕书田,辛显龙,夏晨.

参考文献:

[1] Wolfgang Emmerich. Distributed component technologies and their software engineering implications[A]. Proceedings of the 24th International Conference on Software Engineering[C]. New York,NY,USA:ACM Press,2002. 537 - 546.

[2] Henry Balen. Distributed Object Architectures with CORBA[M]. New York. NY. USA: Cambridge University Press, 2000.

[3] Richard Monson-Haefel,Bill Burke,Sacha Labourey. Enterprise JavaBeans. Fourth Edition[M],Sebastopol,CA,USA:O'Reilly,2004.

[4] Brown Nat, Charlie Kindel. Distributed Component Object Model Protocol - DCOM/1.0[S]. Redmond,WA,USA:Microsoft Corporation,1998.

[5] Gustavo Alonso,Fabio Casati,Harumi Kuno,Vijay Machiraju. Web Services:Concepts,Architectures and Applications[M]. London:Springer,2004.

[6] 杨芙清,梅宏,吕建,金芝.浅论软件技术发展[J].电子学报,2002,30(12A):1901 - 1906.
Yang Fu-qing,Mei Hong,Lv Jian,Jin Zhi. Some discussion on the development of software technology[J]. Acta Electronica Sinica,2002,30(12A):1901 - 1906. (in Chinese)

[7] 吕建,陶先平,马晓星,胡昊,徐锋,曹春.基于 Agent 的网构软件模型研究[J].中国科学(E 辑),2005,35(12):1233 - 1253.

[8] Clemens Szyperski. Component Software:Beyond Object-Oriented Programming. Second Edition[M]. Upper Saddle River, NJ,USA:Addison-Wesley,2000.

[9] G Kiczales,J Lamping,A Mendhekar,C Maeda,C Lopes. Aspect-oriented programming[A]. Proceedings of ECOOP'7[C]. Paris:Springer Verlag,1997. 220 - 242.

[10] Nikunj R Mehta, Nenad Medvidovic, Sandeep Phadke. Towards a taxonomy of software connectors[A]. Proceedings of the 22nd International Conference on Software Engineering

[C]. Limerick, Ireland: ACM Prress, 2000. 178 - 187.

[11] 马晓星,余萍,陶先平,吕建. 一种面向服务的动态协同架构及其支撑平台[J]. 计算机学报, 2005, 28(4): 467 - 477.
Ma Xiaoxing, Yu Ping, Tao Xianping, Lv Jian. A service-oriented dynamic coordination architecture and its supporting system[J]. Chinese Journal of Computers, 2005, 28(4): 467 - 477. (in Chinese)

[12] M Shaw, R DeLine, DV Klein, et al. Abstractions for software architecture and tools to support them[J]. IEEE Transactions on Software Engineering, 1995, 21(4): 314 - 335.

[13] David Garlan, Robert Monroe, David Wile. Acme: Architectural description of component-based systems[A]. Gary T Leavens and Murali Sitaraman, editors, Foundations of Component-Based Systems[M]. New York, NY, USA: Cambridge University Press, 2000. 47 - 68.

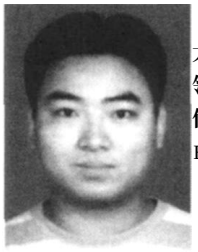
[14] Francisco Curbera, Rania Khalaf, Nirmal Mukhi, Stefan Tai, Sanjiva Weerawarana. The next step in Web services[J]. Communication of the ACM, 2003, 46(10): 29 - 34.

[15] David Garlan, Shang-Wen Cheng, An-Cheng Huang, Bradley R. Schmerl, Peter Steenkiste. Rainbow: Architecture-based self-adaptation with reusable infrastructure[J]. IEEE Computer, 2004, 37(10): 46 - 54.

[16] Wei Song, Xiaoxing Ma, Wanchun Dou, Jian L ü Toward a model-based approach to dynamic adaptation of composite services[A]. Proceedings of 6th IEEE International Conference on Web Services[C]. Washington: EEE Computer Society Press, 2008. 561 - 568.

[17] 曹东刚,梅宏,曹建农. 在中间件中支持用户自定义连接子[J]. 软件学报, 2005, 16(8): 1378 - 1385.
Cao Donggang, Mei Hong, Cao Jiannong. Supporting user-defined connector in middleware[J]. Journal of Software, 2005, 16(8): 1378 - 1385. (in Chinese)

作者简介:



谢德平 男,1983 年出生于江苏南京,南京大学计算机科学与技术系硕士研究生,主要研究领域为 Internet 软件技术、软件体系结构、软件构件技术.
E-mail: xbp@ics.nju.edu.cn

邢 阳 男,1984 年出生于江苏南京,南京大学计算机科学与技术系硕士研究生,主要研究领域为 Internet 软件技术、集成开发环境.
E-bensson@gmail.com

马晓星 男,1975 年出生于江苏南通,南京大学计算机科学与技术系博士、副教授,主要研究方向为 Internet 软件技术、软件体系结构.
E-mail: xxm@nju.edu.cn

曹 春 男,1978 年出生于江苏无锡,南京大学计算机科学与技术系博士、讲师,主要研究方向为访问控制技术、服务计算、软件 Agent 技术. E-mail: caochun@ics.nju.edu.cn

吕 建 男,1960 年出生于江苏南京,南京大学计算机科学与技术系博士、教授、博士生导师,研究方向为软件自动化、并行程序形式化方法、面向对象语言和环境. E-mail: lj@nju.edu.cn