ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.19, No.6, June 2008, pp.1340−1349
DOI: 10.3724/SP.J.1001.2008.01340

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

一种可定制的自主构件运行支撑框架*

孙 熙, 庄 磊, 刘 文, 焦文品+, 梅 宏

(北京大学 信息科学技术学院,北京 100871)

A Customizable Running Support Framework for Autonomous Components

SUN Xi, ZHUANG Lei, LIU Wen, JIAO Wen-Pin+, MEI Hong

(School of Electronics Engineering and Computer Science, Peking University, Beijing 100871, China)

+ Corresponding author: E-mail: jwp@sei.pku.edu.cn

Sun X, Zhuang L, Liu W, Jiao WP, Mei H. A customizable running support framework for autonomous components. *Journal of Software*, 2008,19(6):1340−1349. http://www.jos.org.cn/1000-9825/19/1340.htm

Abstract: In this paper, a dynamically customizable framework for supporting the autonomy of components is introduced. To implement autonomous components, the framework adopts an approach to repacking existing components. By combining Agent technology into components, the framework can customize behavior rules and plans for components to enable components to adjust their behaviors according to the states of the environment. By integrating a rule-engine that can interpret and execute declarative rules, the framework supports the autonomous behaviors of components without recoding and re-deploying of components.

Key words: Internetware; autonomous component; Agent; running support platform

摘 要: 描述了一种动态、可定制的构件自主化的实现手段及运行平台,试图通过对实现自主构件的方法和手段的探索,为开发具有自主性的基于 Internet 的软件系统提供一定的实践基础和经验.在实现自主构件时,采取了改装已有普通构件的方式,将软件 Agent 技术和构件技术结合起来,通过为构件定制行为规则和规划来控制和调度构件的行为,使得构件能够根据环境的状态调整自己的行为.同时,通过集成可以解释和执行声明式规则的规则引擎,使得可以在不修改构件源代码和重新部署构件的情况下,动态定制和实现构件的自主性行为能力.

关键词: 网构软件;自主构件;软件 Agent;构件运行支撑平台

中图法分类号: TP311 **文献标识码**: A

Internet 的出现,使计算机软件所面临的运行环境开始从静态封闭逐步走向动态开放.基于 Internet 的软件系统可以看成是由一系列分布式的自主计算资源动态形成的,完成特定任务的软件联盟[1],与之相适应,软件系统呈现出柔性、多目标性、及连续反应性等多种新的系统形态.因此,基于 Internet 的软件系统具有了自主性、协同性、反应性、演化性和多态性等众多新特征(在文献[2]中,称这种新形态的软件为“网构软件”),这些新特性对于软件理论、方法和技术都提出了新的要求.

目前,面向构件的计算模型已被视为新一代的软件结构模型[3],构件也已成为新一代的软件开发范型.但是,

* Supported by the National Basic Research Program of China under Grant No.2005CB321805 (国家重点基础研究发展计划(973)); the National Natural Science Foundation of China under Grant Nos.60773151, 90412011 (国家自然科学基金)

Received 2006-02-21; Accepted 2006-11-03

传统的构件被视为用于构造应用系统的被动实体,构件无条件地参与系统组装;此外,构件的行为和状态是可预知的,构件的连接机制也是预先定制的,构件与构件之间的依赖关系在系统组装时确立,在运行时保持不变.由此可见,传统的构件缺乏对动态开放环境的适应能力,也缺乏对基于 Internet 软件自主行为能力的推理和支持机制.

要实现自主的、规模可伸缩的、适应动态开放环境的基于 Internet 的软件系统[4],软件系统的构成成分(即构件)必须是自主的[5].即构件一方面具有一定的独立性,能在没有外界干预的情况下独立地完成任务;另一方面,又要具有一定的主动性和自适应性,能够根据自身及外部环境的状态来调整自己的目标及行为.

在现阶段,针对软件实体的自主性研究主要集中在软件 Agent 领域.通常认为,软件 Agent 是处于特定环境下、为实现特定目标或完成特定任务的、具有自主性的软件系统[6].而软件 Agent 的自主性是通过 Agent 的行为以及控制其行为的行为规则来反映的,这些规则规定了 Agent 在不同的环境状态下所应采取的对策[7].然而,一方面,当前软件 Agent 技术尚不成熟,缺少公认的 Agent 定义,对 Agent 开发和运行平台的规范进行支持的实现提供商较少;另一方面,Internet 上的软件实体并非都是软件 Agent,对 Agent 互操作协议等的支持有限,从而单纯基于 Agent 技术来进行开发将在一定程度上限制其适用范围.

目前,构件技术已经得到了很大程度上的认可,产业界支持和推动了基于构件的软件开发,特别是在基础设施方面的建设,如构件库、构件运行支撑平台等都已经有很多实用的产品(如文献[8,9]).J2EE 等构件技术规范提供了较成熟的底层机制支持,如构件的部署、数据库连接、线程资源的管理等,其可靠程度和易伸缩性已得到肯定.如果可以在现有构件技术的框架下对构件进行改进,将 Agent 领域在自主性上的研究成果,如行为规则等,集成到传统的构件模型之中,并支持对这类具有自主性的构件的组装和部署,那么,我们将能实现具有自主性的构件(在下文中,将这种构件称为“自主构件”).这一方案在充分利用已有的构件运行支撑平台的前提下,使构件能在规则的驱动下对不同的环境作出反应,并自主地适应环境的变化;同时,规则的定义、部署、执行与维护应独立于构件原有的计算代码,从而可以方便地支持对传统构件进行复用,对提高网构软件的开发效率和应用范围具有重要意义.

当实现自主构件时,一种较为理想的策略是能够提供某种手段,使得自主构件的实现过程自动化;更进一步地,能够根据用户的需求在线动态地定制构件的自主行为.因此,本文在介绍自主构件的实现机理,包括如何在普通构件中加入行为规则、如何利用和改造已有的构件运行支撑机制来支持自主构件的运行之后,还将介绍如何利用已有的构件开发工具来在线定制自主构件,即用户可以动态地重新定制(或调整)构件的自主行为,无须重新编写或部署构件即可实现新的目标.

第 1 节给出自主构件的实现结构及在中间件平台上的具体实现,同时描述自主构件的自主行为的定制工具.第 2 节描述一个简单的应用实例,介绍在该应用实例中实现构件的自主化的具体过程,并就自主化传统构件的相关问题进行讨论.第 3 节比较相关的其他研究工作.第 4 节总结全文并给出进一步的研究计划.

1 自主构件及其运行支撑

1.1 自主构件的实现结构

传统的构件被认为是可以独立部署并组装的软件实体,由接口规约和实现构成[10].尽管自主构件需根据环境变化动态调整自身行为,但出于组装的需要,它仍应像传统的构件那样向外暴露其接口,并通过接口向外界提供服务.因此,自主构件与传统构件的差别在于其实现接口的方式发生了改变,从而其展示给用户的行为也随之发生了变化,即自主构件与传统构件的差别主要体现在其实现结构和运行时刻的行为语义上,如图 1 所示.其中:环境信息刻画了自主构件所关心的并能感知到的各类环境变量的相关信息,例如各类系统资源的使用状况等;感应器用来感知外界环境的状态变化,并根据环境的状态维护环境信息;自主构件的动作集即自主构件实现其业务逻辑的计算方法的集合.

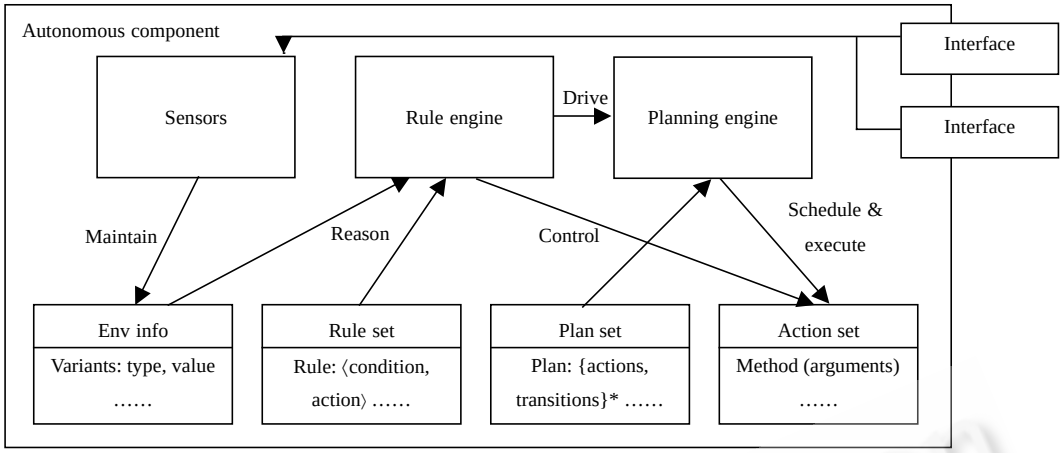

Fig.1 The implementation structure of autonomous components

图 1 自主构件的实现结构图

规则集保存自主构件驱动和控制其行为的规则,这些规则表示为“事实(fact)+动作(action)”,即当给定事实为真(即环境进入某种状态)的时候,构件应执行相应的动作.如调用某个方法,激活某个规划,甚至修改规则集本身.规则集中的规则可以划分为两类:行为驱动规则和行为限制规则.前者描述当条件满足时,自主构件应采取的行为;后者则描述对自主构件行为的约束条件,即构件的行为只有首先符合这种约束条件,才能得以授权顺利执行.

规划集中的规划刻画了实现服务的动作序列,即自主构件在提供服务时应该采取的任务步骤.规划可能是在线方式生成的,即在运行时刻由规划器动态生成并执行,也可能以离线方式根据不同的服务以及不同的服务质量制定,即事先定义好规划并将其保存在规划集中,运行时刻再由规划器调度执行.为了简化实现,当前模型采取的是离线预定义规划的方式.

规则引擎根据环境信息及自主构件的规则集推理自主构件的行为.自主构件只有在特定的规则被触发的情况下才会采取行动.规划引擎则由规则引擎驱动,选择特定的规划来实现服务.在实现服务的过程中,特定动作的执行(即方法的调用)受到构件规则的限制.例如,当行为与限制性规则互相冲突的情况下,该行为将无法执行.

这样,自主构件响应请求、提供服务的基本行为模式如下:

- 当收到服务请求时,规则引擎根据当前的环境状态及规则集对自主构件的行为进行推理.如果不存在可触发的规则,则不采取任何行为,即拒绝服务请求;否则,启动规划引擎,选择并执行相应的规划来实现服务.
- 在提供服务的过程中,如果所有的动作都能正确执行,且不存在某个动作违反限制性规则,则根据规划所规定的步骤执行以提供相应的服务;否则,产生服务失败例外.
- 即使是在没有外界服务请求的情况下,规则引擎也可能会根据自主构件所感知到的环境状态信息来触发行为规则,执行特定的行为.但在这种情况下,自主构件的内部行为不是为了为外界提供服务,而是为了调整其内部状态.

1.2 自主构件的运行支撑

从前面的描述中可以看出,自主构件在具备一些新特征的同时,对需要其提供服务的用户而言,其外部视图与传统的构件没有显著区别,从而确保了自主构件在使用方式上与传统构件的兼容性.正因为自主构件与传统构件在表现形式上的一致,本文选择扩展一种支撑传统构件的构件运行平台(即 PKUAS[8])来支撑自主构件的运行.在具体实现中,作者通过集成开源规则引擎(Drools[11])和设计相应的构件规划引擎,以提供自主构件所需

的规则和规划能力.

1.2.1 支撑平台及工具

PKUAS[6]是一个遵循 J2EE 规范的应用服务器,它是一个为部署和执行互操作构件提供运行支撑环境的中间件平台.PKUAS 实现了所有标准 J2EE 构件的容器,支持多种国际主流互操作协议,并提供了丰富的企业级公共服务.在具体实现中,容器一方面管理构件的生命周期,为构件的实例提供运行环境,另一方面为构件访问各种公共服务提供支持.每个构件实例都对应一个与其类型相适应的通信截取器,以截取和转换外界的服务请求,从而实现支持不同交互协议的构件之间的无缝连接.并且,基于其微内核、开放式的结构,PKUAS 可以通过自定义截取器的方式扩展其容器的能力.

Drools[11]是一个基于 Rete[12]匹配算法的规则引擎,它采用 Java 语言实现,提供面向对象的接口,从而很容易集成到基于 Java 的软件系统中.在 Drools 中,规则由“条件”和“结果”构成,它们都可以用一种类 Java 语句来描述,便于程序员编写和理解.另外,Drools 支持规则的声明式编程,即用户通过 XML 文件声明和定义规则,Drools 则实现对规则的解释和执行.

在人工智能领域中,规划强调在既定目标已经存在的情况下,如何生成一个个细分的子目标的过程.由于目前采用离线预定义规划的方式,因此,我们把实现重点放在子目标已经生成完毕,如何根据子目标的语义来执行规划上.现有实现参考了 Jade[13]系统中 Agent behavior 的概念,将规划定义为一组活动对象和一系列的变迁关系对象的组合,规划引擎通过对它们的解释和执行自动完成给定的规划.

在将传统的构件包装成自主构件时,需要为自主构件预定义行为规则集和规划集.为了实现最大程度的灵活性,我们开发了一个可以在线定制自主构件的规则和规划的可视化管理工具,提供了一种在不终止构件的运行或重新部署构件的情况下,动态改变自主构件的自主行为能力的手段.该可视化管理工具是系列工具 ABCTool[14]的一部分.ABCTool 以基于软件体系结构的构件组装方法(即 ABC 方法[15])为基础,将体系结构贯穿到软件开发的整个生命周期之中,为软件开发的各个阶段提供了不同的支持.在 ABCTool 中,用户可以浏览部署在 PKUAS 上的应用系统的体系结构,并可以任意选择某个构件,对自主行为进行定制.

1.2.2 自主构件的实现框架

网构软件作为一种新的软件形态,具有相应的覆盖软件生命周期的开发方法学,对领域知识、应用需求等的分析,对于识别需自主化的构件,定制自主策略均十分重要.本文重点关注在网构软件系统的开发和运行阶段,对自主策略、运行支撑框架的设计与实现,对方法学更完整的论述参见文献[16].

通过集成现有的规则引擎和规划引擎,我们在 PKUAS 上实现了如图 2 所示的自主构件运行框架.

在扩展运行支撑平台的工作中,其重点包括设计与实现新的自主构件容器,用于感知环境信息的容器截取器,以及对规则、规划进行处理的公共服务.自主构件容器为自主构件的实例提供运行空间,并管理自主构件的生命周期、负责自主构件之间的通信.容器内的截取器实现了自主构件的感应器,它一方面负责截获自主构件之间的交互,为支持不同交互协议的自主构件的互操作提供支持;另一方面捕捉环境状态信息,维护环境信息,为规则引擎提供用于推理和触发规则的数据.考虑到所有自主构件的实例中都存在规则和规划引擎,在实现自主构件时,规则和规划引擎被实现为中间件的公共服务.自主构件容器通过调用这些公共服务来控制自主构件的行为,实现自主构件的服务.

在运行阶段,当自主构件容器接收到服务请求时,会将请求转发给截取器.截取器首先根据自主构件的相关规约调用公共服务(如安全检查、规则推理)来确定是否响应服务请求,或实现构件的某些非功能性需求.在自主构件提供服务的过程中,截取器继续调用规则引擎和规划引擎来调度和调用自主构件的方法,实现具体的功能性需求,即提供服务.服务过程中或服务结束后,截取器也可能会调用相关的公共服务(如事务处理)将计算结果持久化,最后,通过自主构件容器将结果返回给服务请求者.在返回结果时,截取器可能还需要将特定于自主构件的交互转换成特定于服务请求者的交互,以实现异构构件之间的互操作.

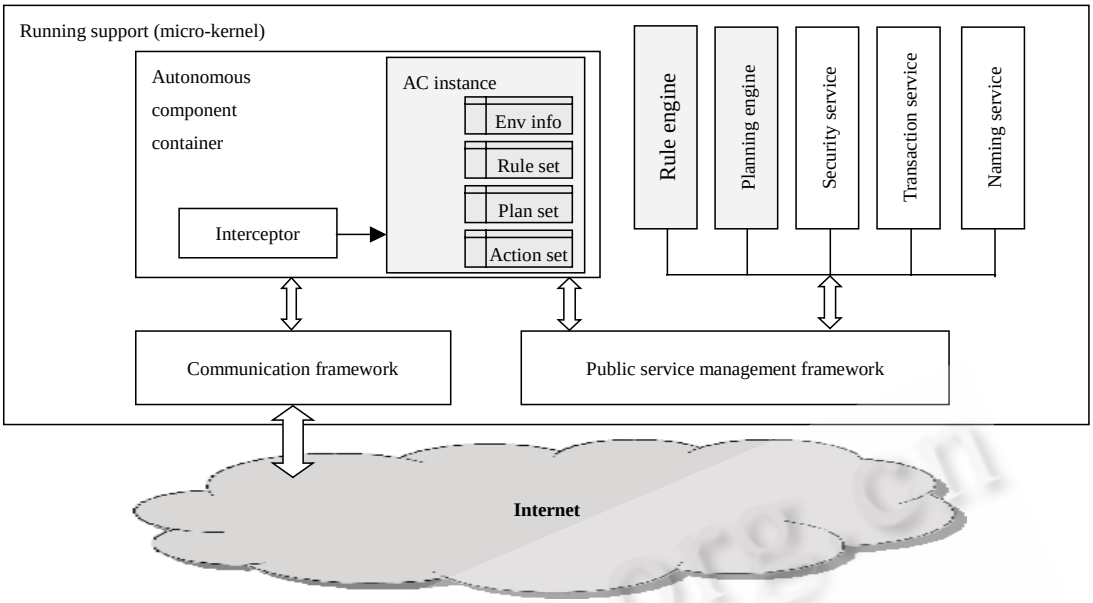

Fig.2 The PKUAS-based running support platform for autonomous components

图 2 基于 PKUAS 的自主构件运行支撑平台

2 应用实例分析

本节通过一个例子来展示具有自主性的应用开发过程.为了集中展示自主构件的实现及其特征,本文选择了一个典型的在线购物应用宠物商店 JPS(Java pet store)[17].JPS 是一个用 Java 开发的在 J2EE 平台上运行的应用实例,它通常用来演示如何利用 J2EE 平台提供的各种能力来开发灵活的、规模可伸缩的、跨平台的企业应用.在 PKUAS 中,用户可以在不修改 JPS 源代码的前提下,通过将 JPS 中某些构件自主化,来动态地加入行为规则来实现不同的商业策略,适应新的商业需求.

在改造 JPS 时,因为 JPS 相对比较简单,只需在其中加入行为规则就能让 JPS 具有某种程度的自主性.而该自主行为主要是由规则引擎来进行控制,规划引擎在其中的作用显得相对次要.所以受篇幅所限,下文介绍具体实现时将略去规划引擎.

2.1 定义自主规则

例如在自主化的 JPS 中,用户希望实现如下的销售策略:系统自主地根据销售情况实时地调整商品价格.即对于畅销的货品,适当提高其价格;而对于无人(或少人)问津的商品,则适当进行打折.利用 ABCTool,用户可以动态地编写如下的 Drools 规则,以控制该销售策略的实现(见表 1).

表 1 中,⟨parameter⟩说明规则内的标识符“invoice”与 JPS 中的构件 Invoice 相关联.⟨condition⟩为触发规则的条件表达式.在这里,各条规则的条件都是用构件 Invoice 的方法(如 isSalable()或 isUnsalable())来定义的,这些方法根据分析售货日志对商品是否畅销给出判断.⟨consequence⟩为规则将要触发的动作,它们都是通过调用 Invoice 的方法来实现的.第 1 条规则规定如果某商品很畅销,则提价 10%;第 2 条规则相反,它规定如果商品滞销,则通过降价 10%来促销商品.通过 ABCTool 定制的规则文件将被放置于 EJB 构件的部署描述文件所在目录,PKUAS 检测到后将自动为该构件加载相应规则.

Table 1 Behavior rules for autonomous components

表 1 构件的行为规则

⟨Rule name="increasePrice"⟩ ⟨parameter identifier="invoice"⟩ ⟨class⟩Invoice⟨/class⟩ ⟨/parameter⟩ ⟨Java: condition⟩ isSalable(invoice, saleLog) ⟨/Java: condition⟩ ⟨Java: consequence⟩ increasePrice(invoice.getItemID(),1.1); ⟨/Java: consequence⟩ ⟨/Rule⟩	⟨Rule name="decreasePrice"⟩ ⟨parameter identifier="invoice"⟩ ⟨class⟩Invoice⟨/class⟩ ⟨/parameter⟩ ⟨Java: condition⟩ isUnsalable(invoice, saleLog) ⟨/Java: condition⟩ ⟨Java: consequence⟩ decreasePrice(invoice.getItemID(),0.9); ⟨/Java: consequence⟩ ⟨/Rule⟩

2.2 自主构件工作流程

在为 JPS 中的"购物车"构件定制了上述行为规则后,当 JPS 接收到客户的购物请求时,JPS 将按如图 3 所示的方式采取行动.

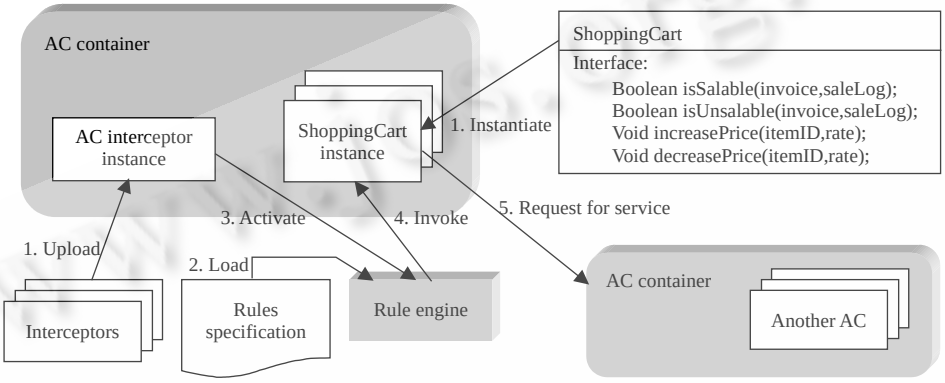

Fig.3 The execution course of the autonomous behaviors of the JPS

图 3 JPS 的自主行为过程

- 在部署阶段,构件容器首先实例化一个购物车(ShoppingCart)构件.
- 如果构件存在对应的规则定义文件,则将自主构件截取器(AcInterceptor)添加到 PKUAS 截取器列表当中,并上载到自主构件容器内.同时,启动规则引擎,加载规则文件,创建规则集.
- 当用户请求到来时,截取器将截获用户请求,并抽取用于规则驱动的相关信息,如客户所购买的商品种类和数量,供规则引擎使用.
- 规则引擎根据这些信息进行推理,以驱动下一步的行为.例如,如果是畅销商品,则会调用购物车构件的方法来提高商品的价格.
- 为了实现调价方法,购物车构件还可能请求调用其他构件的方法.

如上所述定制和部署自主构件后,在运行时,系统对于畅销的商品,将按照给定的策略自行调整其价格以获取更多的收益,且该策略可实时进行修改并反映到该应用中.

2.3 对实现的思考

在实现上述自主化宠物商店系统的过程中,我们发现,在不修改普通构件原有代码逻辑的前提下,为其提供自主行为能力,将受到多方面因素的限制:首先,构件本身的能力的限制:当构件提供的计算方法十分有限时,比如一种较极端的情况,货物构件不提供设置价格的接口,此时可以定义的规则行为显然也受到限制;其次,构件服务的实现方式的限制:在构件响应外界请求并提供服务时,如果整个服务的提供过程中的每一步都是环环相

扣、无法调度或调整的,那么,要改变服务的行为或服务质量也是非常困难的;第三,构件的支撑平台的限制:自主构件的规则引擎和规划引擎等服务的实现受到支撑平台实现的影响;第四,将构件改造成自主构件的手段的限制.在当前的实现中,自主构件截取器充当了自主构件的感应器,尽管通过截取器可以方便地获得用户的服务请求信息,但如果触发行为规则的信息并不仅仅来源于服务请求,就需要在方法截取器之外提供更多的感知环境的手段.

因此,要复用已有的构件,将普通构件改造成自主构件,就必须了解构件本身以及构件的运行支撑平台的局限性,结合构件自身的特征选择不同的自主化方式;在进一步的研究中,我们将对自主构件的行为进行分类,并相应采取不同的技术手段来实现构件的自主化.

一般来说,构件的自主化过程可以分为单个构件的局部行为的自主化和多个构件的协同行为的自主化.对于前者,要自主化单个构件的行为,构件至少应该具备如下特征:

- 能够获取构件的内部状态以及修改自身状态的能力.构件的内部状态可以利用构件的反射机制获得.
- 构件的方法具有适当的粒度、较好的独立性和可调度性.如果构件的服务总是由一个方法来实现,或者实现服务的方法之间的耦合度很大,那么,要改变服务的实现方式或服务质量就基本上没有了可能.自然,要让构件能适应不同的环境状态、满足不同的用户需要、提供自主的服务能力就很困难.

对于后者,自主化过程实质上是将多个构件组装在一起,形成新的自主构件,并通过调用构件的服务来实现新的自主服务.这时候,单个构件的内部状态以及它们如何实现自身的服务可能就都不再重要.要实现新的自主构件,只要能根据各个构件所提供的接口以及环境状态定制新的行为规则和规划即可.而这时,如何协调构件之间的行为及交互就成为实现自主构件的关键,这也是作者下一阶段要开展的研究工作之一.

3 相关工作比较

自主性的研究在软件Agent领域已有一些相关工作[18,19]和实验平台[13,20].与典型的Agent系统相比,本文的工作主要借鉴了其自主性方面的概念和技术,并将其整合到当前的构件模型和平台当中,这种方案保留了已较为成熟的构件系统的诸多长处.文献[21]对J2EE构件框架与符合FIPA标准的多Agent系统FIPA-OS进行了多方面的比较,从中可以看出,在当前的系统实现下,J2EE规范比Agent相关规范提供了更多的底层机制支持,如构件的部署、数据库连接、线程资源的管理等;并且,由于J2EE等构件技术规范拥有众多的实现提供商,其可靠程度和性能指标也比Agent系统更有优势.因此,本文从改造构件的角度入手,提供自动化的普通构件自主化机制,能够较好地复用现有的成熟技术,这对保证网构软件在实际应用中的开发速度和系统质量具有重要意义.

文献[22]基于移动Agent的原理、方法和技术,在面向对象方法与技术的基础上提出了一种适合于开放环境下网构软件需求的开放协同软件模型及其相关的技术体系框架,包括基于移动Agent的协同程序设计技术、多模式交互机制及基于Agent中间件模型和面向体系结构的协同程序设计方法等.相比较而言,本文的工作更多地关注于构件模型与相应的支撑机制,增强现有构件技术框架的开放性、灵活性和自主性,以满足网构软件的需求.

在现阶段构件领域的研究中,也有若干研究工作试图通过对构件原有的模型进行修改或者干脆重新建立新的构件模型来提高或实现构件的适应能力和自主行为能力,期望能够开发出可以更好地适应动态、开放的Internet环境的、具有自主性的复杂软件系统.

文献[23]提出了一种反射式的构件模型Fractal.在Fractal中,构件的反射能力并不是由模型固定的,它可以根据规划者的条件和目标动态地扩展和自我调整,从而可以更好地实现、部署和管理复杂的软件系统.在其当前实现中,用于反射构件元数据的控制对象(control object)需要构件开发人员自行开发.

HADAS(heterogeneous autonomous distributed abstraction system,异构的自主分布式抽象系统)[24]的目标是为了使开发以网络为中心的复杂系统变得更加容易.其提出者认为,开发这类系统的主要困难在于分布式的异构构件在管理上的自治性.从而,在HADAS中对构件的操作和维护不再由一个全局的管理者角色来完成,而是由构件的本地管理者来完成.构件不仅能动态自省,还能动态演化,构件的行为、甚至是构件的结构在运行时刻

都可以根据变化的环境进行适应性调整.

Gravity[25]实现了一种面向服务的构件模型,以管理运行时刻由服务的动态可用性引起的应用系统的变化.Gravity 通过两层管理机制来管理服务可用性的动态变化:构件实例管理和构件组装管理.前者从局部着眼,基于构件描述信息来管理构件实例的服务依赖;后者从全局视图着眼,根据组装描述信息来管理服务组装.

与这些相关工作相比,在模型层面,本文工作的侧重点不在于重新开发全新的自主构件模型并为其提供特定的运行支撑平台,而是在主流构件模型的基础上,在不修改构件源代码以及重新部署构件的情形下直接实现构件的自主化;同时,充分利用已有的构件运行支撑平台,为自主构件提供运行环境.该方案体现了构件及基于构件的运行支撑平台的可复用性,在实际应用中也更具可行性.

在实现层面,本文工作也使用了反射机制来支持自主构件的自省.但不同的是,构件的反射能力不是构件本身所具有的,而是因为我们所选用的构件运行支撑平台实现了反射机制,通过使用构建运行支撑平台的反射机制而让自主构件自动具有了某种程度的反射能力.这样,任意一个部署在该构件运行支撑平台上的构件都具有相同的反射能力,因而理论上利用该平台可以实现任意构件的自主化.但不足之处在于,通过这种反射机制只能得到一些通用的信息,如构件的接口、公共属性等,而无法获取特定于构件或特定于应用的信息,这可能会影响构件的行为规则和规划的定制.对此,在进一步的研究计划中,将考虑通过引入用户自定义感应器以支持对构件或应用特定信息的获取.

4 总结与展望

基于 Internet 的软件系统已成为一种新的软件形态,开发这样的系统面临着很多亟待解决的问题和挑战.本文描述了一种动态可定制的构件自主化的实现手段及运行平台,试图通过对实现自主构件的方法和手段的探索,为开发具有自主性的基于 Internet 的软件系统提供一定的实践基础和经验.

在实现自主构件时,我们采取了一种改装已有普通构件的方式,将 Agent 领域的自主技术和构件技术结合起来,通过为构件定制行为规则和规划来控制和调度构件的行为,使得构件能够根据环境的状态调整自己的行为,提供网构软件所需的自主性.另一方面,通过集成可以解释和执行声明式规则的规则引擎,既为动态定制行为规则提供了可能,又可以在不修改构件的情况下实现构件的自主性行为能力,使得构件的自主化过程显得较为简单.尽管采用其他方法,例如单纯基于 Agent 技术也能实现规则驱动的自主行为,但通过扩展业界主流的构件技术、复用已有构件和运行基础设施,本文的方法具有更高的可行性和更大的应用范围.

在现阶段,该工作所实现的应用实例较为简单,仅用于展示所提出的自主化方法的可行性.在下一阶段,将进一步探讨如何通过对多个构件的行为进行封装来实现更加复杂的自主构件,探讨如何协同多个构件之间的交互行为等.此外,还将通过实现更复杂的应用实例,进一步展示本文工作的先进性.

同时,我们还将在现有工作的基础上抽象出更加通用的自主构件模型,并开展基于自主构件的网构软件的开发及其开发方法学研究.

致谢 在此,我们要衷心感谢黄罡副教授,他在系统的实现过程中提出过很多宝贵的指导性意见和建议,也感谢杨杰、田田、杨瑜等同学,他们都参与过系统设计的讨论,也参加过部分系统代码的编写工作,其中,自主构件的定制工具就是由杨杰同学协助完成的.

References:

[1] Shaw M. Architectural requirements for computing with coalitions of resources. In: Clements P. Perry D. Ran A. eds. Proc. of the 1st Working IEEE/IFIP Conf. on Software Architecture. Netherland: Kluwer, 1999. 1−6.

[2] Yang FQ. Thinking on the development of software engineering technology. Journal of Software, 2005,16(1):1−7 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/16/1.htm

[3] Ning JQ. Component-Based software engineering. In: Nahouraii E. ed. Proc. of the 5th Int'l Symp. on Assessment of Software Tools and Technologies. Washington: IEEE Computer Society, 1997. 34−43.

[4] Suzuki J, Suda T. A middleware platform for a biologically inspired network architecture supporting autonomous and adaptive applications. IEEE Journal on Selected Areas in Communications, 2005,23(2):249−260.

[5] Kral J, Zemlicka M. Autonomous components. In: Hlavac V, Jeffery KG, Wiedermann J, eds. Proc. of the SOFSEM 2000. LNCS 1963, Berlin / Heidelberg: Springer-Verlag, 2000. 375−383.

[6] Wooldridge M, Jenings NR. Intelligent agents: Theory and practice. The Knowledge Engineering Review, 1995,10(2):115−152.

[7] Liu J, Jin X, Tsui KC. Autonomy-Oriented computing (AOC): Formulating computational systems with autonomous components. IEEE Trans. on Systems, Man, and Cybernetics—Part A: Systems and Humans, 2005,35(6):879−902.

[8] Mei H, Huang G. PKUAS: An architecture-based reflective component operating platform. In: Cooper B. ed. Proc. of the 10th IEEE Int'l Workshop on Future Trends of Distributed Computing Systems (FTDCS). Los Alamitos: IEEE Computer Society, 2004. 163−169.

[9] Pan Y, Wang L, Zhang L, Xie B, Yang FQ. Relevancy based semantic interoperation of reuse repositories. In: Proc. of the 12th ACM SIGSOFT Symp. on Foundations of Software Engineering (FSE-12). New York: ACM Press, 2004. 211−220. http://portal.acm.org/citation.cfm?id=1029924

[10] Szyperski C, Gruntz D, Murer S. Component Software—Beyond Object-Oriented Programming. 2nd ed., Addison-Wesley, 2003.

[11] Drools. 2008. http://labs.jboss.com/portal/jbossrules/

[12] Rete algorithm tutorial. 2008. http://www.cis.temple.edu/~ingargio/cis587/readings/rete.html

[13] Jade. 2008. http://jade.tilab.com/

[14] Xiang JL, Yang J, Mei H. A software architecture-based component composition tool: ABC tool. Journal of Computer Research and Development, 2004,41(6):956−964 (in Chinese with English abstract).

[15] Mei H, Chen F, Feng YD, Yang J. ABC: An architecture based, component oriented approach to software development. Journal of Software, 2003,14(4):721−732 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/14/721.htm

[16] Mei H, Huang G, Zhao HY, Jiao WP. An architecture centric engineering approach to Internetware. Science in China (Series E), 2006,36(10):1100−1126 (in Chinese with English abstract).

[17] JPS. 2008. http://java.sun.com/developer/releases/petstore/

[18] Bradshaw JM, Feltovich P, Jung H, Kulkarni S, Taysom W, Uszok A. Dimensions of adjustable autonomy and mixed-initiative interaction. In: Nickles M, Rovatsos M, Weiss G, eds. Agents and Computational Autonomy: Potential, Risks, and Solutions. LNCS 2969, Berlin: Springer-Verlag, 2004. 17−39.

[19] Castelfranchi C, Falcone R. Founding autonomy: The dialectics between (social) environment and agent's architecture and powers. In: Nickles M, Rovatsos M, Weiss G, eds. Agents and Computational Autonomy: Potential, Risks, and Solutions. LNCS 2969, Berlin: Springer-Verlag, 2004. 40−54.

[20] Nwana HS, Ndumu DT, Lee LC. ZEUS: An advanced tool-kit for engineering distributed multi-agent systems. In: Proc. of the Practical Application of Intelligent Agents and Multi-Agent Systems. 1998. 377−392. http://citeseer.ist.psu.edu/nwana98zeus.html

[21] Casagni M, Lyell M. Comparison of two component frameworks: The FIPA-compliant multi-agent system and the Web-centric J2EE platform. In: Lieberherr K, ed. Proc. of the 25th Int'l Conf. on Software Engineering (ICSE 2003). Los Alamitos: IEEE Computer Society, 2003. 341−350. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1201213

[22] Lü J, Tao XP, Ma XX, Hu H, Xu F, Cao C. Studies on agent-based Internetware models. Science in China (Series E), 2005,35(12):1233−1253 (in Chinese with English abstract).

[23] Bruneton E, Coupaye T, Leclercq M, Quema J, Stefani JB. An open component model and its support in Java. In: Crnkovic I. Stafford JA. Schmidt HW. Wallnau K. eds. Proc. of the 7th Int'l Symp. (CBSE 2004). LNCS 3054, Edinburgh: Springer-Verlag, 2004. 7−22.

[24] Ben-Shaul I, Holder O, Lavva B. Dynamic adaptation and deployment of distributed components in Hadas. IEEE Trans. on Software Engineering, 2001,27(9):769−787.

[25] Cervantes H, Hall RS. Autonomous adaptation to dynamic availability using a service-oriented component model. In: Finkelstein A. Estublier J. Rosenblum D. eds. Proc. of the 26th Int'l Conf. on Software Engineering (ICSE 2004). Edinburgh: ACM Press, 2004. 614–623.

附中文参考文献:

[2] 杨芙清.软件工程技术发展思索.软件学报,2005,16(1):1−7. http://www.jos.org.cn/1000-9825/16/1.htm

[14] 向俊莲,杨杰,梅宏.基于软件体系结构的构件组装工具:ABC tool.计算机研究与发展,2004,41(6):956–964.

[15] 梅宏,陈锋,冯耀东,杨杰.ABC:基于软件体系结构,面向构件的软件开发方法.软件学报,2003,14(4):721–732. http://www.jos.org.cn/1000-9825/14/721.htm

[16] 梅宏,黄罡,赵海燕,焦文品.一种以软件体系结构为中心的网构软件开发方法.中国科学(E 辑),2006,36(10):1100–1126.

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

孙熙(1982－),男,上海人,博士生,主要研究领域为软件工程,软件构件技术,自主计算.

庄磊(1982－),男,硕士生,主要研究领域为软件构件技术,自主计算.

刘文(1982－),男,硕士生,主要研究领域为软件构件技术,自主计算.

焦文品(1969－),男,博士,副教授,主要研究领域为软件工程,智能软件,构件技术.

梅宏(1963－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为软件工程,软件复用,软件构件技术,分布对象技术.