多 Agent 系统的技术研究

张 林[1],徐 勇[1],刘福成[2]
(1. 安徽财经大学 信息工程学院,安徽 蚌埠 233041;
2. 安徽财经大学 管理学院,安徽 蚌埠 233041)

摘 要:多 Agent 系统是由多个智能 Agent 组成的有机系统,这使得它具有比单个 Agent 更强大的处理能力。它表现出自组织性、鲁棒性、分布性以及很强的复杂行为。文中论述了 Agent 和多 Agent 系统的有关理论、方法和技术。主要包括智能 Agent 的特性、结构和推理;介绍多 Agent 系统的体系结构分类和常见的几种通信机制;以及面向 Agent 的程序设计的现状和发展。

关键词:Agent;多 Agent 系统;体系结构
中图分类号:TP18 **文献标识码**:A **文章编号**:1673-629X(2008)08-0080-04

Research of Multi-Agent System Technology

ZHANG Lin[1], XU Yong[1], LIU Fu-cheng[2]
(1. School of Information Engineering, Anhui University of Finance & Economics, Bengbu 233041, China;
2. School of Management, Anhui University of Finance & Economics, Bengbu 233041, China)

Abstract: A multi-agent system (MAS) is a system composed of several software agents, collectively capable of reaching goals that are difficult to achieve by an individual agent. MAS can manifest self-organization and complex behaviors even when the individual strategies of all their agents are simple. Summarizes the theory, method and the key technologies of agent and multi-agent system. The main topics are as follows: the properties, structure and reasoning of intelligent agent; the architecture and communication method of multi-agent system; current status and developing trends of agent oriented programming.

Key words: agent; MAS; architecture

0 引 言

Agent 俗称"智能体"、"智能代理",它的研究最早起源于分布式人工智能(DAI),它为分布式开放系统的分析、设计和实现提供一个崭新的途径,被誉为"软件开发的又一重大突破",在许多领域得到了更为广泛的应用。另一方面,Agent 技术作为一门设计和开发软件系统的新方法已经得到了学术界和企业界的广泛关注。Agent 的研究大致可分为智能 Agent、多 Agent 系统(Multi-Agent System, MAS)和面向 Agent 的程序设计(Agent Oriented Programming, AOP)这 3 个相互关联的方面。智能 Agent 是多 Agent 系统研究的基础,有关智能 Agent 的研究可以认为是统一在多 Agent 系统的研究之下的。智能 Agent 可以被看成是多 Agent 系统研究中的微观层次即主要研究 Agent 个体的行为和特征;而有关 Agent 之间关系的研究则构成了多 Agent 系统研究的宏观层次。智能 Agent 和多 Agent 系统的最终实现和成功应用必须要借助于 AOP 及其开发工具这一媒介。多 Agent 系统是由多个 Agent 组成的系统,它在 Agent 理论的基础重点研究 Agent 的相互操作性以及 Agent 间的协商、协作等问题,基于 Agent 的系统(Agent-Based system, ABS)是指使用了 Agent 思想或技术的系统。ABS 可能只包含一个 Agent,如用户接口 Agent 或软件秘书等,但通常是指多 Agent 系统的应用实例。Jennings 等人还指出,ABS 可以是只借用 Agent 概念而用其他技术(如 OOP)实现的系统。

收稿日期:2007-11-04
基金项目:安徽省自然科学研究项目(2006KJ011A, KJ2007 B246);安徽省高校青年教师"资助计划"项目(2007jql083);中华全国供销合作总社科研项目(GXZSKY06012zd)
作者简介:张 林(1975-),安徽宿州人,硕士,讲师,研究方向为数据挖掘、算法设计;刘福成,教授,硕士生导师,研究方向为人力资源管理。

1 Agent 与智能 Agent

1.1 Agent 的特性

由于 Agent 技术应用在不同领域的系统中,即使

在同一个系统中也处于不同层次和结构中,因此Agent的定义和表现就大相径庭了。那么究竟什么是Agent? 它又有那些特性? 这些是Agent研究者和应用者最为关心的问题,也是目前Agent研究领域一直在争论和探讨的热点问题之一。著名学者Hewitt曾指出:在面向Agent计算领域回答"什么是Agent?"这一问题,就像在人工智能领域回答"什么是智能?"一样令人困惑和为难。在有关Agent特性的研究中,最经典和广为接受的是Wooldridge等人有关Agent的"弱定义"和"强定义"的讨论[1]。一些典型的研究报告和应用系统中[2],在对Agent的描述或定义的基础上指出,一个Agent的最基本的特性应当包括:反应性、自治性、面向目标和针对环境性。每个Agent首先应具备这4条最基本的特性,然后再根据其应用情况拥有其他特性。Agent可以拥有的其他特性包括:移动性、自适应性、通信能力(包括协商、协作等能力)、理性、持续性或时间连续性、自启动、自利等特性。一些研究人员还从Agent的特性进行了更为深入的研究。

对Agent的典型定义大都来源于定义者设计和开发的一些Agent实例,根据他们各自的设计需要反映出Agent的一些特征和侧面,但他们都无法全面地描述和表达智能Agent的完整特性和性质。

实际上,对于任何一个系统,研究和开发人员不可能也没必要构建一个包括上述所有特性的Agent或多Agent系统,他们往往是从应用的实际需要出发来开发包含以上几部分特性的Agent系统。但既然是称为面向Agent的技术或系统,那么就应当满足上述提到的4条最基本的特性。可以根据Agent的特性给出一个Agent的简单定义:Agent是一类在特定环境下能感知环境,并能自治地运行以代表其设计者或使用者实现一系列目标的计算实体或程序。

1995年,Wooldridge和Jennings对Agent作出了权威性的定义[1]:Agent是一个基于软件或硬件的计算机系统。它拥有以下特性:自治性、社会能力、反应性、能动性、学习性、通信性和移动性。a.自治性,是Agent最基本的特性,指行动上的独立性。Agent一经初始化后,可不受干预直接执行。Agent控制着自己的外部行为和内部状态,可以被授权去做某种决定,完成一些事情。b.反应性,是指Agent清楚所处的环境,能感知其所处的环境,并能对环境发生的改变及时作出响应。c.能动性,是Agent能采取主动的以目标为导向的行为,适时地对流程作出调整,而不必等待环境发生变化,可提高敏捷性。d.学习性,是指基于历史活动的执行情况指导未来行为,Agent这种对时间上的适应性称为学习性。e.通信性,是指Agent有能力与其他Agent交互。一些Agent可形成Agent群,它们之间的接口和联系不是固定的,而是随任务的不同变化的,其最大的好处在于耦合小。Agent可以最小的代价加入系统或从系统中移出。f.移动性,是指Agent有能力在一个网络上随时随地、自主地从一台主机移到另一台上,Agent将数据封装在执行的一个线程中,每个Agent独立于其他Agent。

1.2 Agent的结构和推理

Agent的基本结构一般分为思考型Agent、反应型Agent和混合型Agent三种[3]。在实际系统开发时,可以根据Agent的具体功能要求在一种基本结构的基础上进行设计和拓展。

(1) 思考型Agent。

思考或慎思型Agent(deliberative Agent)是将Agent看作是一种特殊的知识系统,即通过符号AI的方法来实现Agent的表示和推理。这是建造Agent的最经典方法。

思考型Agent的最大特点就是将Agent看作是一种意识系统[3],即把它们作为人类个体或社会行为的智能代理,模拟或表现出所谓的意识态度,如信念、意图、目标、承诺等等。这有助于研究者们以一种自然直观的方法来理解、描述和规范基于Agent系统的内部结构、运行规律和变化状态等。

根据大多数通用的慎思方法,认知构件基本上由两部分组成:规划器和世界模型。这种方法中有一个基本的假设:对认知功能进行模块化是可能的,即可以分开来研究不同的认知功能(如感知、学习、规划和动作),然后把它们组装在一起构成智能自治Agent。从工程角度看,功能模块化降低了系统的复杂性[4]。

Wooldridge曾经指出,在Agent形式化方面,经典的命题逻辑和一阶谓词逻辑是不合适的。目前最常用的形式化工具是模态逻辑(包括各种时态逻辑)和可能世界语义,即将意识态度看成是一种模态。关于模态逻辑和可能世界语义的研究已形成一整套的相关理论,成为表示和推理智能Agent和多Agent系统的最有力的形式化工具。意识态度的其他表示方法还有元语言与对象语言以及具有符号解释结构的演绎模型等。

(2) 反应型Agent。

对于思考型Agent过多强调其意识系统,尽管在智能上取得种种突破,但是也使得Agent结构过于复杂,反映变慢,而且其形式化的描述也不是很完善,这就导致了反应型Agent的出现。反应型Agent起源于Brooks的思想,即Agent不依赖任何符号表示,直接根据感知输入产生行动,从而提出Agent智能行为的"感

知－动作"模型。他们认为这就像昆虫一样,不需要知识,不需要表示,也不需要推理,也一样可以生存在自然环境之中。Agent 的行为在现实世界与周围环境的交互作用中表现出来。也有人认为反应型 Agent 是来自下面的假设:Agent 行为的复杂性可以是 Agent 运作环境复杂性的反应,而不是 Agent 复杂内部设计的反应。MIT 的 R. Brooks 提出一种不同于符号 AI 的,称之为子前提结构来建造 Agent 的控制机制。该结构是由用于完成任务的行为来构成的分层结构,这些行为相互竞争以获得对机器人的控制权。这种虽然简单的结构在实践中被证明是非常高效的,它甚至解决了传统符号 AI 很难解决的问题。

(3) 混合型 Agent。

反应型 Agent 能及时而快速地响应外来信息和环境的变化,但其智能程度较低,也缺乏足够的灵活性。思考型 Agent 具有较高的智能,但无法对环境的变化作出快速响应,而且执行效率相对较低。混合型 Agent 是上述两种体系结构的结合,综合了二者的优点,具有较强的灵活性和快速响应性。

混合结构的系统通常被设计成至少包括如下两部分的层次结构:高层是一个包含符号世界模型认知层,它用传统符号 AI 的方式处理规划和进行决策;低层是一个能快速响应和处理环境中突发事件的反应层,它不使用任何符号表示和推理系统。反应层通常被给予更高的优先级。比较著名的典型实例有 PRS (procedural reasoning system), TouringMachine 和 InterRRap 等;也有环型体系结构,如 Will 等。

2 多 Agent 系统(MAS)

多 Agent 系统的协作求解问题的能力超过单个 Agent,这是多 Agent 系统产生的最直接的原因。导致多 Agent 系统研究逐渐兴起的其他原因还包括与已有系统或软件的互操作性、提高系统效率和鲁棒性等。

如果把单个的 Agent 设计成一个完整的专家系统,那就失去 Agent 设计的意义了;一般单个的 Agent 主要用于模拟人的智能行为,拥有的知识库一般都较小,因此处理问题的能力较弱,对一些复杂的问题往往求解困难,有些甚至不能求解。从弱小的昆虫的通过群体实现高智能社会这一自然现象出发,也可以通过多个 Agent 之间的通信和协调或协作形成了一个多 Agent 系统,它能克服单个 Agent 的局限性,灵活地求解复杂问题。

多 Agent 系统是由一组在逻辑上或物理位置上分布的 Agent 组成的,它们通过网络连接,共享资源,为完成共同任务而形成一个有组织的系统,它具有社会性、自治性、协作性。各个独立的 Agent 都有自己的作用,同时与其他 Agent 以及它所处的系统环境交互,获得必要的信息和服务,并与其他 Agent 形成相互依赖和相互组织的关系。与单个 Agent 相比,多 Agent 系统具有如下特点:每个成员 Agent 仅拥有不完全的信息和问题求解能力,不存在全局控制,数据是分散或分布的,计算是异步、并发或并行的;这就使得 MAS 具有分布性、可靠性(具备恢复单个故障元件或性能老化元件功能的能力)、可扩展性(系统的扩充只需向系统中加入不同种类的 Agent 即可,增加的 Agent 只要遵循共同协议标准,就可以与系统内的其他 Agent 协同工作)以及容错性(单个 Agent 的错误可以通过其它 Agent 进行处理)等。

多 Agent 系统的各 Agent 之间的通信和运行控制模式不同,严重影响到整个系统的性能。

2.1 多 Agent 系统体系结构

从运行控制的角度来看,多 Agent 系统的体系结构可分为:集中式、分布式和混合式[5]。

(1) 集中式结构。

如图 1 所示,类似网络的星型结构。将 Agent 分成若干个组,每个组内的 Agent 采取集中式管理,即每一组 Agent 提供一个控制 Agent ,通过它来控制和协调组内不同 Agent 的合作,如任务规划和分配等等。整个系统采用同样的方式对各成员 Agent 组进行管理。集中式能保持系统内部信息的一致性,实现系统的管理、控制和调度较为容易。此方式的缺点是:随着各 Agent 复杂性和动态性的增加,系统结构层次较多,除了增加控制的管理成本外,也让数据传输过程中出错的概率大大增加;而且一旦控制 Agent 崩溃,将导致以其为根的所有 Agent 崩溃。

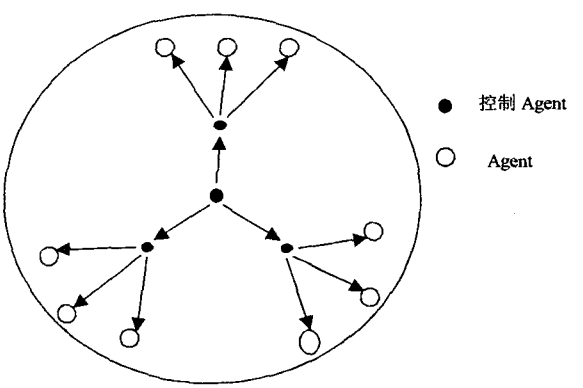

图 1 集中式结构

(2) 分布式结构。

如图 2 所示,各 Agent 组之间和组内各 Agent 之间均为分布式结构,各 Agent 组或 Agent 无主次之分,处于平等地位。Agent 是否被激活以及激活后做什么

动作取决于系统状况、周围环境、自身状况以及当前拥有的数据。此结构中可以存在多个中介服务机构,为 Agent 成员寻求协作伙伴时提供服务。这种结构的优点是:增加了灵活性、稳定性,控制的瓶颈问题也能得到缓解;但仍有不足之处:因每个 Agent 组或 Agent 的运作受限于局部和不完整的信息(如局部目标、局部规划),很难实现全局一致的行为。当 Agent 的数目过多的时候,则带来了维护成本的增加。

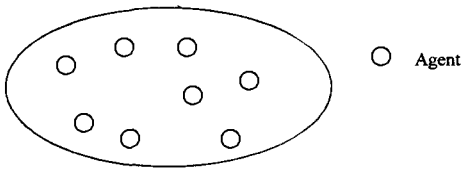

图 2　分布式结构

(3) 层次式结构。

如图 3 所示,实际上是集合了集中式和分布式两类结构,它包含一个或多个层次结构,每个层次结构有多个 Agent,这些 Agent 可以采用分布式或者集中式。相邻层之间的 Agent 可以直接通信,或者利用控制 Agent 进行广播通信。这样通过分类,就把功能相似的 Agent 归类为某一层,有利于对同类型 Agent 以某种方式进行统一管理,参与解决同类型 Agent 之间的任务划分和分配、共享资源的分配和管理、冲突的协调等。如果不相邻的层次进行某种通信,可以增加一些协作 Agent,来处理相关信息。此种结构平衡了集中式和分布式两种结构的优点和不足,适应分布式多 Agent 系统复杂、开放的特性,因此是目前多 Agent 系统普遍采用的系统结构。

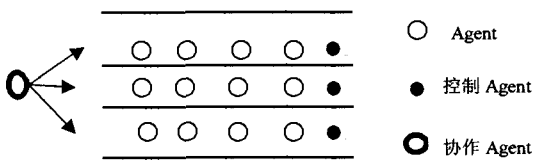

图 3　层次结构

2.2　多 Agent 系统的通信机制

在多 Agent 系统中,常用的通信机制有直接通信、广播通信、联邦系统(Federation System) 和公共黑板系统(Blackboard System)[6]。

(1) 直接通信用于需要通信的 Agent 已确切知道通信的另一方是谁,二者之间进行直接交流。这种通信的方式特点是简单、高效;但是它只能实现一对一的通信,而且需要明确知道通信的双方。

(2) 广播通信用于 Agent 特别是控制 Agent 将消息广播给同组的所有成员或者不知道通信对方的地址。其缺点是控制 Agent 不知道消息是否被接受,而且该方式也造成了 Agent 间数据传输量的大大增加。

(3) 联邦系统通信:各 Agent 之间的交互是通过联邦体来实现的,Agent 可以动态地加入联邦体,接受联邦体提供的服务。这些服务包括:接受 Agent 加入联邦体并进行登记;记录加入联邦体的 Agent 能力和任务;为加入联邦体的 Agent 提供通信服务;对 Agent 提出的请求提供响应;在联邦体之间提供知识转换与消息路由等等。

(4) 公共黑板系统是一种集中控制的方式,每个 Agent 把信息放在其它所有 Agent 都可以存取的黑板上,供其它成员共享。

3　面向 Agent 的程序设计

目前,国际上还没有统一的多 Agent 系统开发语言标准。已经设计出了许多种专门面向 Agent 的程序设计语言,主要有 Shoham 提出的 AgentO,Rao 提出的 AgentSpeak,Koen V Hindriks 等人提出的 3APL 和 GOAL 等。但现有的面向 Agent 程序设计语言远不如面向对象程序设计语言那样通用,所以在实现 Agent 时通常采用成熟的设计语言和技术。目前人们在实现软件 Agent 方面还是主要采用 Java、Agent 通信语言 KMQL(Knowledge Query and Manipulation Language) 进行开发,这使得它们逐渐成为事实上的标准。

由于 Java 具有基于网络和平台的无关、内置的多线程、面向对象性、通信能力和安全性等几个方面的特性,使得它特别适合于 Agent 的开发,所以 Java 是目前开发多 Agent 系统较为理想的编程语言。随着多 Agent 技术的研究和应用,一些公司和研究机构专门研究了多 Agent 系统开发平台,如 Reticular 系统公司的 AgentBuilder、国际知识系统公司的 AgentX 等。这些平台均采用 Java 实现,具有计算机平台无关性等特点。但总的来说,这些平台现在仍不够完善。

知识查询与操纵语言 KQML 是目前最通用的 Agent 通信语言,是一种用于 MAS 中 Agent 之间交换信息和知识的语言和协议,为表达消息和处理消息提供了标准的格式。KQML 包含了一系列可扩充的行为原语,行为原语定义了 Agent 对知识和目标的各种操作,在其上可以建立 Agent 互操作的高层模型。KQML 消息是线性的字符流,语法简单,消息内容可是非 ASCII 的二进制流。主要优点有:具有灵活的结构和良好的可扩充性;独立于网络传输机制;独立于内容表达语言;能够满足相互传递的基本要求。由于 KQML 还在进一步的发展和完善,因此存在着一些不足之处,如不同的 KQML 系统互相兼容,没有一个固定的规范标准去实现通信机制等等。

(下转第 87 页)

tion'><
位置把菜单显示出来。

```
function showMainMenu(menus){
var menuText="";
<!- $('')是DWR框架里定义的工具函数,等价于Javascript脚本语言里的document.getElementById(""),显然前者比后者更简单实用->
//var mainmenu=document.getElementById("menuposition");
var mainmenu=$('menuposition');//获取显示菜单位置
for(var i=0; i< menus.length; i++){
if( menus[i].parent==0){
menuText=menuText+"<li id='foldheader' name='"+menus[i].id+"'>"+menus[i].name+"</li>"+"<ul id='foldinglist' style='display:none'></ul>";
}
}
mainmenu.innerHTML=menuText;
}
```

配置dwr.xml、定义并访问后台Java类和回传Javascript函数,完成了一个AJAX动态调用机制的循环,实现了不提交整个页面也可以刷新信息的卓越用户体验。在无限级的菜单使用中,每展开一级菜单才从数据库中读取其下级菜单的内容生成并显示出来。当不访问菜单时,页面只包含并显示第一级菜单的内容。这样避免了前台一次载入数据过大,造成响应缓慢的问题,并且在不使用时,不会读取大量数据,避免给服务器带来不必要的负担,提升了页面响应速度。

4　结束语

介绍了AJAX技术的优势特点,以及AJAX框架出现的缘由。提出一种使用AJAX框架DWR来实现无限级树型菜单的方法,并给出实现代码。此方法既具有AJAX动态显示数据的特点,轻松实现了无限级扩展,又隐藏了异步通信编程工作,提高了开发效率,在实际项目中有普遍推广和深化开发的意义。

参考文献:

[1] 游丽贞. AJAX引擎的原理和应用[J]. 微计算机信息,2006,22(23):205-207.

[2] 徐　驰. Ajax模式在异步交互Web环境中的应用[J]. 计算机技术与发展,2006,16(11):228-233.

[3] Zakas N C. Javascript高级程序设计[M]. 北京:人民邮电出版社,2006.

[4] Asleson R. AJAX基础教程[M]. 北京:人民邮电出版社,2006.

[5] 余翔宇. AJAX技术及其框架实现[J]. 软件导刊,2006(9):28-30.

[6] Carneiro C. AJAX made simple with DWR[EB/OL]. 2005. http://www.javaworld.com/javaworld/jw-06-2005/jw-0620-dwr.html.

(上接第83页)

4　结束语

面向Agent技术为复杂、开放、分布式系统的开发和实现提供了新途径,在开发新型应用中将起着重要作用。随着应用系统的日益复杂,系统需要的数据和信息的规模日益扩大,以及对应用系统人性化、智能化的日益需求,面向Agent技术,特别是多Agent技术将会受到越来越多的重视。但是Agent目前还有一些问题需要解决:

(1)缺乏有关基于Agent的软件系统开发方法学,这使Agent的开发无规范可言。开发高性能Agent及多Agent系统必须有一套科学化的方法来解决:个体Agent的可靠性和扩展性?整个系统的Agent如何组织和协作?基于Agent软件系统的可扩展性?等等。

(2)基于Agent软件系统的设计方案缺乏评估、比较的标准。由于缺乏开发方法,则系统开发自然仁者见仁智者见智了,对于系统设计方案的评估、比较也就难以有一个统一的标准了。

(3)与可重用技术和分布式对象技术的结合不够紧密。由于目前基于Agent的软件系统在实践上还是使用面向对象语言作为开发平台,客观上要求软件系统能够与可重用技术和分布式对象技术的结合紧密结合一起。

(4)不成熟的面向Agent的程序设计语言。虽然目前有众多的面向Agent的程序语言,但都是不太成熟的,使得目前基于Agent的系统还是多以成熟的面向对象语言作为开发语言。

参考文献:

[1] Wooldridge M J, Jennings N R. Intelligent Agent: Theory and Practice[J]. Knowledge Engineering Review, 1995, 10(2):115-152.

[2] Yang Kun, Liu Da-you. Agents: properties and classifications[J]. Computer Science, 1999, 26(9):30-34.

[3] 刘大有,杨　鲲,陈建中. Agent研究现状与发展趋势[J]. 软件学报,2000,11(3):315-321.

[4] 王汝传,徐小龙,黄海平. 智能Agent及其在信息网络中的应用[M]. 北京:北京邮电大学出版社,2006.

[5] Wooldridge M. An introduction to multi-Agent systems [M]. New York: John Wiley & Sons, Inc.,2002.

[6] 蒋云良,徐从富. 智能Agent与多Agent系统的研究[J]. 计算机应用研究,2003(4):31-34.