计算机科学 1996 Vol. 23 №. 1

# 软件 Agent

姚 郑 高 文

(哈尔滨工业大学计算机科学与工程系 哈尔滨 150001)

**摘 要** This paper talks about the difficulty in software agent research based on the discussion of the basic conception of software agent, and introduces the research works about software agent in the world.

**关键词** Software agent, Software agent construction, AOP, ABSE, Negotiation.

## 1. 引 言

近年来,Agent 一词变得越来越引人注目,有关 Agent 的各项研究在国外已得到迅猛发展。Agent 一词的译法在国内还没有定论,通常译为智能体、智能主体或代理者等,但这些译法都不能准确表达 Agent 一词的丰富内涵,故在本文中采用不译法。软件 Agent (Software Agent)作为 Agent 研究的主要方向,其研究人员来自许多不同的研究领域诸如软件工程、机器人、知识表示、基于知识的系统、数据库、问题求解、规划、机器学习、认知科学、人机接口等等。不同的研究者为了更好地描述自己的 Agent 已经发明了许多新的词汇如智能 Agent、智能接口、自适应接口(adaptive interface)、知识机器人(Knowbot)、软件机器人(softbot)、用户机器人(userbot)、任务机器人(taskbot)、个人 Agent 和网络 Agent 等等。尽管存在着这样的分歧,有关 Agent 的研究仍然给我们提供了这样一个机会,即将我们的技术成果直接送到最终用户手里,这就是 Agent 研究的基本思想——开发出能够吸引和帮助所有最终用户的软件系统。

## 2. 软件 Agent

软件 Agent 的研究者来自许多不同领域,使得软件 Agent 的含义也具有多重性,无论在软件 Agent 的特例、用途和理论动机上都有很大差别。

### 2.1 什么是软件 Agent?

究竟什么才是所谓软件 Agent? 问题的答案依赖于所问问的对象并且可能存在极大的差异。但有一点是明确的,即软件 Agent 是计算机程序,就象专家系统、字处理软件等等一样,并且其它计算机程序还不能做的事情,如用自然语言对话,软件 Agent 也不能做。那么,哪些程序是真正的软件 Agent 呢? 没有一种简单的必要性或充分性准则供我们做这样的判断,但是我们可以寻找一些软件 Agent 共有的特性作为判断依据,这些特性来自我们对一些典型方法的考察(不必是互相排斥的方法)。在做进一步讨论之前,我们首先考察软件 Agent 的一些典型的高层特性。

从抽象意义上,软件 Agent 是自主且持续运行的。Agent 可以代表某个特定的人采取行动(只要该行动符合该人的利益),因此,Agent 必须也是强壮的且能够安全地处理私人信息。Agent 应当是高度交互的——它们将自己的大部分时间用于与其它 Agent 及人类通信。Agent 是其所处的计算环境中的主动参与者,也就是说,它们对于全局系统状态发生反应和导致其变化。

从具体意义上,软件 Agent 是实际的和有用的,例如某些 Agent 可以使得一些简单的在线(online)、重复和费时的工作自动化。

下面我们分别讨论软件 Agent 的各种定义。

2.1.1 软件 Agent 是在线的伪人类(pseudo-people) 软件 Agent 是组成所谓 Agent 社团的成员。这种观点来自机器人和分布式 AI 的研究,对于软件 Agent 采用了拟人的描述方法,即 Agent 是包

高文 博士导师 863-306 专家组成员,主要研究兴趣:图象处理、多媒体技术、人工智能。姚郑 博士生,研究兴趣:面向 Agent 技术,Agent 协商模型。

含了信念(belief)、承诺(commitment)、义务(obligation)、意图(intention)等精神状态的实体。有关精神状态究竟应当包含哪些因素没有唯一正确答案，只能根据实际需要而定。比如 Shoham 认为信念、承诺、能力和决定是最基本的精神因素，而 Cohen 等人认为信念和目标(goal 或 choice)是基本因素[1]。

这些精神状态通过某种形式化语言进行了精确定义，其含义与我们通常所使用的意义相似，但并不完全相同，这源于某种实际需要如推理方面的考虑。这样，判断一个程序是否为软件 Agent 就要看它能否被上述精神状态所描述。尽管从理论上讲任何程序都可以通过精神状态来描述，但这种描述并不总是有益的。采用这种拟人描述所带来的好处是我们拥有了一种能够用于描述协作 Agent 之间的交往活动的启发性词汇，例如 Agent1 给 Agent2 发送 E-mail 是因为它觉得"有义务"这样做；同时这种高层描述使我们不必考虑一些具体实现问题，因而对于那些我们尚未十分了解的复杂系统如机器人来讲是非常方便的。

2.1.2 软件 Agent 是核心 AI 中其它领域研究的测试平台(testbed) 软件 Agent 是 AI 研究新的通用研究工具。随着 AI 的发展，专家系统和机器人在 AI 中的地位正在逐渐被软件 Agent 所取代。Etzioni 认为软件 Agent 是进行核心 AI 研究的理想基础[2]，在传统的核心 AI 研究中存在许多中心问题如规划、机器学习、知识表示等等，软件 Agent 为探索这些问题提供了一个非常合适的测试平台。例如，软件 Agent 消除了许多物理环境中不可避免的棘手问题，更进一步说，开发和试验软件 Agent 无论从资金的花费、精力的付出等方面都相对要低一些。同时，与仿真世界相比，软件环境更便于得到、经济实惠和真实。

比如，Etzioni 开发的 UNIX 软件机器人和互联网软件机器人主要用于系统管理方面，为探索 AI 中的许多中心问题，尤其是规划问题，提供了一个非常有趣的基础。软件机器人通过(低层的)UNIX 原语进行对话，其主要交互对象是其拥有者，不支持 Agent 之间的通信，因而其作为测试平台的通用性受到限制。此外，还有人提出了适用于更为广泛的核心 AI 研究的 Agent 模型，在此不作详细论述。

2.1.3 软件 Agent 是智能的在线助手 软件 Agent 是具有智能行为的人工助手，如人工秘书、人工设计者、人工导师等，它们将完成现实世界中对应角色的主要工作。人工秘书要帮助其拥有者完成 E-mail 的筛选、网络信息的浏览、制定约会备忘录等等，人工设计者将帮助设计队伍完成某些设计任务，人工导师则给其用户提供有关系统使用方面的各项建议以供参考。这种在线助手 Agent 在某些场合下是非常有用的，尤其当我们面临的是那些非常复杂的在线环境以及大信息量的场合如互联网。

接口 Agent[3]是一类特殊的在线助手，其设计目标是简化用户与某些特定的事先存在的应用系统之间的交互活动。这些 Agent 被设计成为能够学习和预测用户的行为和爱好。知识机器人[4]主要作为管理大量电子数据库的图书管理员，它们并不是通用目的的，而是高度复杂的特定系统。作为具体实例，GDS 是一个多 Agent 系统，它能够帮助设计队伍解决设计工作中的关键问题如突发性(emergence)。CAP 是一个日志管理者，它能够从实践中学习用户在日程安排上的偏爱。COACH 则是一个人工导师，它自动给用户提供所需要的有关系统使用方面的各项建议。*)

2.1.4 软件 Agent 是协商者

一组软件 Agent 能够通过协商作出某项决定或组成联盟，协商(negotiation)并非新事物，在经济学中对之已有深入研究。随着 Agent 研究的深入进行，人们更为重视的是 Agent 的动态特征和 Agent 之间的交互活动的研究，协商是解决 Agent 之间冲突的一种有用的技术。例如若需要组织一次具有复杂的时间限制的会议时，软件 Agent 可以帮助人们解决这件令人头疼的时间限制均衡问题，而且可能不会让任何人感到失望。

会议日程安排和资源重新配置是典型的软件 Agent 在协商中的应用。B. Moulin 等人开发的支持人机协同工作的多 Agent 系统可以自动解决会议日程安排问题[5]；A. Sathi 和 M. S. Fox 等人对于资源重新配置问题从算法上进行了研究并实现了多 Agent 间的协商[6]。另外，还有人开发了模仿人类协商

*)相关文献参见"Comm. of ACM, 37(7) 1994"

行为"协商者"Agent,被用于一个相关的称为"外交"的游戏程序里"

2.1.5 其它观点 本节综合讨论有关软件 Agent 定义的其它观点。R. Goodwin[7]从模型角度考察了 Agent,他认为一个 Agent 是一个被建立起来完成某项任务或一系列任务的实体,并给出了 Agent 的抽象模型,这对于实际开发工作具有指导意义。M. H. Coen 则提出了判断一个程序是否为软件 Agent 的最小准则集[8],包括下面 5 个准则:

1. 能够互相对话,其通信模式可能是非常复杂的;

2. 是自主的和智能的,能够对复杂的刺激发生反应且伴以复杂的和适当的行为;

3. 必须是强壮的,能够对那些未曾预料的变化作出适当的反应;

4. 通常不是固定不变的——它们具有记忆力并且随着时间的推移改变自己的行为。Agent 可以采用形式化的机器学习技术,或者在其运行中更为理性地搜集数据,且能够对计算环境中的某些特殊事件作出自发反应;

5. 在典型情况下是分布于网络之中,因而它们的行为可能同时具有局部和全局效应。

另外,也有人认为(私人观点)Agent 之所以成为 Agent 完全取决于一个观察者的个人意愿,即若一个观察者认为某程序是 Agent 则它就是 Agent。这种观点虽然从哲学角度来看是完全站得住脚的,但它似乎没有给人们带来任何益处。更进一步说,由于 Agent 的研究日益受到重视,可能限制一下这个词的范围会更有好处。

### 2.2 软件 Agent 的构造及其困难

对于各种软件 Agent 的定义而言,都存在着如何构造软件 Agent 的问题,这个构造过程本身就提出了一些困难的技术问题。

定义一个 Agent 的抽象行为通常是直接的,但是传统程序设计语言没有为 Agent 所进行的典型的高层"在线行为"提供原语层次上的支持,例如从用户处以图形方式接受结构化的信息或与其他 Agent 进行可靠的通信行为。

构造 Agent 通常是一项多层次的开发活动(multi-layered approach)[8]。首先,它要求大量的特定的系统知识如网络协议、窗口系统知识等等,这通常是非常困难的。其次,Agent 的构造经常包括中层计算问题,例如,让 Agent 处理并发事件或自动处理错误等。最后,Agent 要服务于它们的计算基础,例如 AI 研究者可能想要构造一个知识表示系统,每一层次都需要独立实现和调试,显然简化 Agent 的构造过程势在必行。

另外,如何将新的 Agent 传播于世并让其他人使用也是困难的,因为 Agent 通常具有复杂的地点特殊性(site-specific)。

最后,对于那些处理敏感信息甚至代表用户与其他人交往的 Agent 而言,其拥有者需要对于该 Agent 是否能够正确运行具有足够的信心。

总之,目前在构造软件 Agent 方面存在三个难题:

1. 用传统程序设计语言和操作系统难于构造软件 Agent。2. Agent 的使用推广非常困难。3. 人们难以对一个未知的(也许有错的)Agent 产生足够的信心。有关 Agent 研究的问题并不止这些,我们在下面将有论述。

## 3. 相关研究成果

在前面我们讨论 Agent 定义时已经介绍了许多在特定软件 Agent 开发方面所取得的研究成果,本节再从其它角度介绍相关的研究成果。

• 面向 Agent 的程序设计(Agent-Oriented Programming,简记为 AOP) 最早由 Shoham 在 1990 年提出[1],他认为 AOP 是一种以计算的社会观为基础的新型程序设计范例,并将 AOP 看作 OOP 的特例,Agent 就是包含了信念、承诺、能力和决定等精神状态的实体,他定义了一种形式化语言来描述 Agent 的精神状态,同时又提出了一种对应的 Agent 程序设计语言 AGENT0,该语言提供了一些用于 Agent 之间通信的高层原语如 REQUEST(请求)、INFORM(通知)等,这些原语来自语言行为理论(Speech Act Theory),为 Agent 通信提供了更为丰富的机制。其他学者如 Freeman 和 Beynon 也从不同角度讨论了 AOP。

• 基于 Agent 的软件工程(Agent-Based Software Engineering)[9] 是为了解决同类或异类 Agent 之间的交互作用(interoperate)——Agent 交换信息和服务从而共同解决问题。Genesereth 等人提

出了一种"联邦式"(federation)Agent 体系结构,并设计了一种用于 Agent 之间相互对话的通用 Agent 通信语言(ACL),通过 ACL 可以完成各种 Agent 之间的通信行为。目前,ACL 正在实现当中,若 ACL 真能按照预期的目标得以实现,则它将对软件 Agent 的研究带来极大的影响。

· Agent 协商模型 在一个包含多个 Agent 的环境里,Agent 之间产生各种各样的冲突是不可避免的,而协商是解决冲突的有效途径。有关协商的研究已经成为 AI 的一个重要领域,这方面的研究成果已有许多,基本上是沿着协议与实现两个方向进行的。在协议方面有早期的合同网协议和近年来出现的新协议如基于言语行为的协商协议(Speech-Act-Based Negotiation Protocol)等等,具体协商系统的实现在 2.1.4 节已有介绍。

· Agent 开发环境 为了方便某些特定类型 Agent 的开发工作,一些相应的 Agent 开发环境已经研制成功。M. H. Coen 开发的 SodaBot[8]是一个通用目的的软件 Agent 用户环境和构造系统。它的基本元素是所谓"基本软件 Agent"——用于建造 Agent 的计算框架或者说是 Agent 操作系统。SodaBotL 是为该系统配备的一种新的程序设计语言,用户使用该语言可以很容易地实现一些典型类型的软件 Agent 应用,如个人在线助手和会议日程安排系统,"基本软件 Agent"与 SodaBotL 之间的关系类似于 UNIX 和 C++的关系。

D. C. Smith 等人开发的 KidSim (Kid Simulation)[10]系统则是另一种 Agent 开发环境,它是一个工具包系统,提供了一种新的程序设计方法,完全消除了传统语言中的语法问题,综合了图形重写规则和演示性程序设计两种新思想,使得儿童可以很容易编程自己的行为来构造和修改仿真程序。

## 4. 结 论

在前面我们对软件 Agent 的有关问题进行了论述。显然,作为一个崭新的研究领域,需要解决或澄清的问题还有许多,比如 Agent 如何学习、不同的 Agent 如何协同工作,以及它们如何通信等等,甚至于 Agent 究竟是不是一个可处理的研究领域目前还不十分清楚。这些问题的解决有待于进一步的研究工作,Agent 的未来产生于构造真实 Agent 的科学和对于最终用户使用这些 Agent 的经验性研究。为了达到这一目的,我们必须从许多不同的例子和学科中学习新东西。

## 参考文献


[1]Y. Shoham, Agent Oriented Programming, AI, Vol. 60, 1993

[2]O. Etzioni, D. Weld, A Softbot-Based Interface to the Internet, CACM, 37(7), 1994

[3]P. Maes, Agents that Reduce Work and Information Overload, 同[2]

[4]C. Knoblock, Y. Aren, An Architecture for Information Retrieval Agents, In Working Notes of the AAAI Spring Symposium on Software Agents, Stanford, 1994

[5]B. Moulin 等, A Multi-Agent System Supporting Cooperative Work Done by Persons and Machines, Conf. Proc. 1991 IEEE Intl. Conf. on System, Man, and Cybernetics, Charlottesville, USA, 1989

[6]A. Satbi, M. S. Fox, Constraint-Directed Negotiation of Resource Reallocation, TR CMU-RI-TR-89-12, Carnegie Mellon University, 1989

[7]R. Goodwin, Formalizing Properties of Agents, TR CMU-CS-93-159, Carnegie Mellon University, Stanford, 1992

[8]M. H. Coen, SodaBot: A Software Agent Environment and Construction System, TR 1493, MIT, Cambridge, 1994

[9]M. Genesereth, S. Ketchpel, Software Agent, 同[2]

[10]D. C. Smith 等, KidSim: Programming Agents without a Programming Language, 同[2]


更正:我刊 1995 年 Vol. 22 No. 5 所发表论文"实用化的软件重用方法及其环境"之第一作者应时,现力时应,特向作者表示歉意。