面向 agent 软件工程

Agent-Oriented Software Engineering

吴元斌　石纯一

（重庆三峡学院计算机系　重庆万州404000）（清华大学计算机系　北京100084）

Abstract　Agent-oriented techniques represent a new means of analyzing, designing and building complex software systems. They have the potential to significantly improve current practice in software engineering. In this paper some important techniques and methods for analyzing, designing, specification, implementation, and verification are analyzed. The comparison of characteristics between agent-oriented engineering and object-oriented engineering are given. And the issues and research methods of agent-oriented engineering are discussed.

Keywords　Agent, Agent-oriented technique, BDI model

1　引言

面对复杂分布软件系统的开发，现有软件工程技术常遇到困难。面向 agent 技术为这类系统的理解、建模、开发提供了一种很自然的方法，较现有其它软件技术更适合于解决这类复杂的现实问题[1]，是面向对象、构件、软件体系结构等软件工程思想的一种自然进步，被认为是网络时代计算机系统的基础。

Oren Etzioni 认为："对于智能 agent，99%属于计算机科学，而1%属于人工智能"[7]。面向 agent 软件工程方法是面向 agent 系统开发引入软件产业的关键因素，现有软件开发技术不能直接用于面向 agent 的软件开发，除了基本概念不同外，更重要的是现有方法不足以描述 agent 灵活性、自主问题求解能力、交互和 agent 组织结构的复杂性等特性[7,9]。近年来 Wooldridge、Jennings 等人进行了面向 agent 软件工程的研究，本文首先描述了面向 agent 技术的基本概念和观点，然后分析了在面向 agent 的分析与设计、规范、实现、验证中现有的一些重要技术和方法，并与面向对象软件工程特性作了比较，最后讨论了面向 agent 软件工程存在的问题和研究方法。

2　面向 agent 技术的基本概念和基本观点

面向 agent 软件工程以 agent 作为关键抽象机制，研究复杂分布式软件系统的工程实现问题。agent 是指处于某一环境中封装的计算机系统，为满足其设计目标，具有在环境中灵活、自主行动的能力[1]。它具有如下一些特性[1,8]：(1)自主性：agent 能控制自己的内部状态和行为，不需要外界（人或其它 agent）的直接干预。(2)反应性：agent 位于特定的环境中，通过感知器接收有关环境的状态，通过效应器作用于环境，并能适时地对环境的改变作出响应。(3)主动性：agent 不仅能对环境作出反应，而且能为未来目标预先行动。(4)社会性：agent 能与其它 agent（或人）进行交互，为完成目标能参与社会活动（如协作、协调、协商等）。

面向 agent 技术的基本观点是：将现实世界看成一系列自主 agent 按一定方式组成的社会，agent 彼此间进行各种交互与通信，完成各种复杂任务。用计算机世界中的 agent 来模拟现实世界中的实体（将非自主实体也看成特殊 agent，称为被动 agent[10]）。这种抽象机制是对分布、动态、开放、复杂的现实问题在计算机世界中自然、直观的模拟，与现有其它软件抽象技术（如面向对象技术）相比，抽象层次更高，更易为一般人所理解和接受。

3　面向 agent 软件开发方法与技术

3.1　面向 agent 分析与设计

面向 agent 分析的目的是理解系统的需求，建立系统模型。面向 agent 设计的目的是将分析阶段的抽象模型变换成抽象级足够低、易于实现的模型。近年来，许多学者对面向 agent 建模技术和方法学进行了广泛的研究。各种方法可大致分成如下几类[9]：(1)扩充 OO 建模技术或方法学的适用性来设计 agent 系统。(2)扩展知识工程的方法学和建模技术。(3)利用形式化方法和语言（如 Z 语言），提供定义框架以支持 agent 或 agent 系统规范。(4)支持特定的 agent 系统开发。如：CASSIOPEIA，支持基于合同网的系统设计，已用于 Robot Soccer。

一种典型的面向 agent 分析与设计方法——Gaia[9]，它将软件系统看成一个组织，组织内包含各种角色，角色与 agent 间不必是一一对应的。Gaia 方法的优点是分析与设计模型独立于 agent 体系结构，但在支持 agent 的自私性、组织结构、协作协议、语义等方面还存在不足。

Gaia 分析是理解系统及其结构，将组织视为彼此间具有一定关系、按一定模式进行交互的角色的集合。系统的最高抽象级是组织，分析阶段产生组织模型。组织模型由角色模型和交互模型组成。角色模型指定系统的关键角色。角色是实体功能的抽象描述。它由两类属性描述：(1)许可（permission）：角色可利用资源的类型和数量。(2)责任（responsibility）：角色具有的功能，责任分为活动（liveness）责任和安全（safety）责任。交互模型表示角色之间的联系，由一组协议组成。协议是角色间的交互类型，协议定义包括目的、创始者、回应者、输入、输出、过程等5个属性。

Gaia 设计是降低分析阶段的组织模型的抽象级，使得传统设计技术可用来实现 agent。Gaia 设计关心的是社会 agent 如何协作实现系统级目标及为此目标每个 agent 需要的功能。设计过程将产生三个模型：(1)agent 模型：指定组成系统

吴元斌　讲师，研究方向为 agent 技术及应用。石纯一　教授，博士生导师，研究方向为人工智能应用基础。

的 agent 类型，从分析阶段的角色变换而来。(2)服务模型：指定实现 agent 角色的主要服务，描述服务有输入、输出、前置条件、后置条件四个属性。服务从角色的协议、行为、责任、活动特性中衍生。(3)熟人模型：描述不同 agent 之间通信关系，用有向图表示，其中结点表示 agent 类型，弧表示通信路径。熟人模型可从角色、协议和 agent 模型中衍生。

3.2 规范

规范是建立系统的基础，这里讨论的规范，不是 Gaia 规范，而是基于 agent 的意图模型。将 agent 看成有意图的系统，如 agent 的 BDI 模型，用信念(belief)、愿望(desire)、意图(intention)来刻画 agent 的结构，用模态逻辑和可能世界语义作为表示 agent 的形式化工具。按 BDI 模型，agent 系统的规范框架包括[6]：(1)agent 具有的信念。(2)agent 将完成的目标。(3)agent 完成的行为及其影响。(4)agent 之间及与环境之间的交互。

最成功的规范框架是用模态逻辑表示规范，典型的时态模态逻辑 agent 规范框架包括：用正规模态逻辑联接词表示 agent 的信念；时态逻辑联接词表示系统的动态性；用正规模态逻辑联接词表示意动(如愿望、意图、义务等)；一些工具表示 agent 完成的行为。但时态模态逻辑的可能世界语义存在以下问题：(1)它隐含 agent 是逻辑全知的，具有用于推理的无限资源，没有实际 agent 满足这种特性。(2)可能世界语义通常没有基础，描述 agent 状态的抽象表示与具体计算模型没有准确的关系。

3.3 实现

实现阶段关心的是如何正确实现规范，即把抽象的规范转换为具体的计算系统，完成这种转换的方法有人工求精、直接执行、编译等。

人工求精是通过按原则而非形式化的求精过程，求精规范到可执行的形式。如"自顶向下，逐步求精"技术。但应保证每一步是前一抽象规范的真正求精。典型的工作是 KGR 方法(由 Kinny、Georgeff、Rao 完成)，是用四步设计 BDI agent 系统，它紧密相关于 DBI 模型的特殊实现：PRS 体系结构。这四步方法总结为：(1)确定应用领域的相关角色，并据此建立 agent 类层次。(2)确定每个角色相应的责任、需要提供的服务，然后决定服务的目标。(3)确定要完成每一目标的计划，以及计划适合的上下文条件。(4)确定系统的信念结构，即每一计划和目标所需的信息。重复以上四步分析过程，直到形成的模型接近于 PRS agent 体系结构，然后系统模型用 PRS 实现。此方法用于 OASIS 空中交通管理系统的开发。

直接执行规范是指给定用逻辑语言 L 描述的系统规范 φ，将其作为执行规范，直接解释规范以产生 agent 行为。解释 agent 规范可看成是可满足性的构造证明，通过为 φ 建立模型表明 φ 是可满足的。若规范语言 L 的规范可给定一个计算解释器，则建立模型可看成是执行规范。如对 Concurrent MetateM 语言，赋予 agent 行为的模态逻辑规范，此规范直接执行产生 agent 的行为。在 agent 中，时态逻辑模型被指定为线性离散的状态序列，执行 agent 规范是构造状态序列的过程。这些状态序列可看成用程序执行时作历史踪迹，Concurrent MetateM 的模态逻辑是基于计算解释器。

编译 agent 规范是通过自动合成过程将抽象的规范转换成具体的计算模型，其目标是减少抽象的符号规范，变成简单的不用符号表示的计算模型。这种推理是在编译时(离线)完成，编译后的系统执行时很少或不需要做运行时的符号推理。编译方法通常依赖于时态/模态逻辑模型与有限状态机之间的关系，用于 agent 开发的典型例子是"Situated automata"范型，它用知识逻辑指定 agent 系统的感知部分，然后用基于构造证明技术从规范直接合成自动机。

人工求精的缺点是可能引入错误，若没有证实每一步是真正的求精，则实现的正确性很大程度上依赖于开发人员的直觉。直接执行 agent 规范，在运行时要进行操纵规范的符号表示，这种操纵通常类似于某种形式的推理，计算代价高。自动合成的通用方法在理论上有吸引力，但也受限于：(1)随着 agent 规范语言表达力的增强，离线推理代价太高以至无法实施；(2)所产生的系统无学习能力；(3)agent 规范框架没有具体的计算解释器，难于合成。

3.4 验证

验证是确定系统是否正确地实现了规范，若系统用非形式化方法开发，则验证尤为重要。验证方法可分为两类：公理化方法和模型检查方法。

公理验证是用具体的程序，由此衍生出表示程序行为的逻辑理论(程序理论)，若程序理论与规范用同一逻辑语言表达，则验证变成证明；规范是程序理论的定理。程序理论可利用公理化系统实现的程序设计语言来完成，一旦完成公理化，程序理论可用系统的方法从程序文本中衍生。一个典型的例子是用时态信念逻辑公理化两个多 agent 语言(Agent0和 Concurrent MetateM)。给出公理，可用上述方法系统地衍生出表示系统特性的程序理论。这种方法依赖于 agent 的操作足够简单，其特征可用逻辑公理化。

另一有效的验证方法是基于规范语言语义的模型检查。模型检查方法是给定语言 L 的公式 φ 及 L 的模型 M，决定 φ 在 M 中是否成立，即是否有 $M\models_L\varphi$。基于模型检查的验证的研究与模态逻辑相关，也依赖于时态逻辑的模型和有限状态机的紧密关系。假定 φ 是某一系统规范，π 称为实现 φ 的程序，为确定它是否真正实现了 φ，由 π 可给出与 π 相应的模型 M_π，M_π 决定了 π 的所有可能计算，确定是否 $M_\pi\models\varphi$，即规范 φ 在 M_π 中是否成立。若程序 π 满足规范 φ 则验证成立。Rao 和 Geotgeff 提出了用 BDI agent 规范语言的逻辑模型、语言公式，确定公式在模型中是否成立的算法。它基于正规模态逻辑，将 DBI 模态词加入 CTL 分支时间框架，算法在多项式时间完成。

公理化方法将验证转化为时态逻辑的证明，证明的困难使得此方法受到限制。用分支时间时态逻辑 CTL 的模型检查可在多项式时间完成，然而 agent 规范语言的语义没有基础，对任一程序来说，无法确定其信念、愿望和意图，因此验证变得很困难。

4 与面向对象软件工程特性的比较

面向对象软件工程已比较成熟，Booch，Coad/Yourdon，OMT 等方法得到了广泛认可，最近又出现了 UML，而目前面向 agent 软件工程还处在发展的早期。两种工程方法一个明显的相似之处是都以问题领域中的实体标识来建立系统模型，不同传统的基于功能或数据分析方法。它们之间又有明显的不同，如 OMT 建立的是对象模型、动态模型、功能模型，而 Gaia 分析阶段建立的是角色模型和交互模型，而设计阶段建立的是 agent 模型、服务模型、熟人模型。面向 agent 方法较面向对象方法至少有以下特点：

(1)许多问题固有的分布性(数据、能力、控制)，本身就是

由多个交互自主实体组成的松散耦合网。面向 agent 方法采用粗粒度的自主计算实体(agent)作为抽象机制,以社会学的观点和人们熟悉的概念(组织,角色等)进行分析、建模、设计复杂分布式系统。面向对象方法则采用细粒度的非自主计算实体(对象)为抽象机制,对复杂问题分析与建模,面向 agent 方法较面向对象方法自然、直观、简单,容易理解。

(2)面向 agent 分析与设计处理实体间知识级交互,有专用的交互语言(如 KQML 等),实体能自主确定交互的方式、范围、时间、内容;同时,存在多种组织关系(如等级制、专家共同体、市场机制等),适合复杂系统的建模。面向对象方法中对象的交互是用处于语法级对象间的消息传递来实现,对象间的组织关系少(如包含、聚合关系),缺乏足够的机制来模型化复杂系统。

(3)面向 agent 方法主要强调角色、责任、服务、目标这些抽象机制来处理复杂性,以何时完成何种目标来分析应用领域,核心是完成的目标。面向对象方法主要强调完成目标的行为类型。因为在任何应用领域中目标比行为或计划更稳定,强调目标与强调行为这看似很小的改变,却导致实质上的不同,面向 agent 的目标分析使得系统设计更稳定、健壮、模块化,能进行渐增式地开发和测试,具有可扩展性。上下文敏感的计划提供了模块性和构成性,不必改变已有的计划,为实现同一目标的新的上下文计划可加入系统中,这使得系统能处理易变性和特例。

(4) AOP(Agent Oriented Programming)是 Shoham 在 OOP 的基础上提出的[11],并将 AOP 视为一种特殊的 OOP,但存在明显不同,如:OOP 的基本部件是对象,而 AOP 的基本部件是 agent;OOP 对定义基本部件状态没有明确规定,而 AOP 中,按 agent 的具体模型(如 DBI 模型),基本部件状态可包含信念、承诺、能力、选择等;OOP 用方法引用处理消息,而 AOP 的消息类型来自言语行为理论,可包含通知、请求、提供、承诺、拒绝等,agent 通信有专用语言,如 KQML;通常 agent 在面向 agent 方法中被实现为有意图的系统,而 OOP 将对象作为类的实例。

尽管面向对象方法与面向 agent 方法有明显的不同,但对象与 agent 之间有一种天然的进化关系,有人称 agent 是活化的对象,或对象加上思维状态等。现有一些重要的面向 agent 方法研究中都利用或扩展面向对象方法来实现面向 agent 方法(如 Gaia 和 KGR 方法等)[2,9]。

结束语 目前面向 agent 软件工程还处在发展的早期,现有的技术和方法还很不成熟。缺乏系统性和标准化,没有贯穿软件生存周期(如可行性研究、分析、设计、实现、验证、维护等)的系统化技术与方法,缺乏对 agent 重要功能需求的支持,如:行为的反应性和主动性、形式语义、行为规范既是陈述式的又可执行、与平台无关、开放性、异质性、协作能力、模块化等[3]。目前面向 agent 软件工程面临许多问题和挑战,其中最重要的问题是[1,6,7]:(1)理解 agent 求解适合的情形;(2)有原则而非形式化的开发 agent 系统。对这些问题,许多学者进行了研究,典型的工作如 Gaia 和 KGR 方法,这表明研究面向 agent 软件工程的有效方法是利用 agent 技术自身的特性及现有软件工程的成功技术与方法。

参 考 文 献

1 Jennings N R. On agent-based software engineering. Artificial Intelligence, 2000,117(2): 277～296

2 Kinny D, Georgeff M. Modelling and design of multi-agent systems. In: M. Wooldridge and N. R. Jennings, eds. Intelligent Agent Ⅱ (LNAI Volume 1193), Springer-Verlag, Berlin, Germany, 1997. 1～20

3 Fisher M, Müller J, Schroeder M. Methodological foundations for agent-based system. The Knowledge Engineering Review, 1997,12(3): 323～329

4 Nikolaos, Skarmeas. Agents as objects with knowledge base state. London: Imperial College Press, 1999

5 Jennings N R, Wooldridge M. Agent Technology: Foundations, Applications and Markets, Berlin: Springer, 1998

6 Jennings N R, Wooldridge M. Agent-oriented software engineering. In: J. Bradshaw, ed. Handbook of Agent Technology, AAAI/MIT Press, 2001

7 Wooldridge M. Agent-based software engineering. IEE Proc. Software Engineering, 1997,144 (1): 26～37

8 Wooldridge M, Jennings N R. Intelligent agents: Theory and practice. Knowledge Engineering Review, 1995,10 (2): 115～152

9 Wooldridge M, Jennings N R, Kinny D. The Gaia methodology for agent-oriented analysis and designs. Internat. J. Autonomous Agents and Multi-Agent Systems, 2000,3(3): 285～312

10 陆汝钤,石纯一等. 面向 agent 的常识知识库. 中国科学, E 辑, 2000, (5): 453～463

11 Shoham Y. Agent-oriented programming. Artificial Intelligence, 1993, 60(1): 51～92

(上接第104页)

阵部分(归约矩阵可由前面所提到的方法求得)。这两部分在算法中都用一句话代替,因为这样更有利于把算法的核心思想体现出来。对算法的其他说明可在算法的注释中得到。

在算法中,先行后列和先列后行的归约顺序产生的归约矩阵,一般说来是不同,如果最优 H 回路是唯一的,答案不会变,否则不同归约顺序可能得出不同的最优 H 回路。

该算法的效率主要是估计出状态空间树中不受限制的结点个数,我们可以用估算法。该方法是在状态空间树中生成几条 LC 路径(按 LC 值选择路经上的结点)。设 X 是某 LC 路径上的一个结点,而且 X 在状态空间树的第 i 级。在结点 X 处用限界函数确定没受限界的儿子结点数 m_i,在队列中选择下一个 LC 值最小的结点作为树中某路径上的下一个结点。这些路径的生成在一个叶结点或某一结点的所有儿子结点都已被限界终止。用这些 m_i 总和即可以估算出这棵状态空间中不受限结点的总数。

结论 本文对货郎担问题提出了精确的、合符实际的问题描述和定义,并用图论的方法研究解决此问题的算法,并对算法时间复杂度进行了分析。更重要的是该问题的提出和解决为具有相同性质的其他问题提供了有价值的解决方法。

参 考 文 献

1 肖位枢. 图论及其算法. 北京: 航空工业出版社. 1993

2 Knuth D E. The art of computer programming. Addison-Wesley Publishing Company, Inc. 1973,3

3 Boudy J A, Murty U S R. Graph theory with application. The Macmillan pross LTD, 1976

4 邹海明,余祥宣. 计算机算法基础. 华中理工大学出版社, 1984

5 徐卓群,张乃孝,杨东青,等. 数据结构. 高等教育出版社,1985

6 Howitz E. Fundamentals of computer algorithms. Computer Science Press, Pitman, Inc. 1978

7 戴一奇. 图论与代数结构. 北京: 清华大学出版社,1995