面向 Agent 软件工程：现状、挑战与展望

毛新军

（国防科学技术大学计算机科学与技术系 长沙 410073）

摘 要 面向 Agent 软件工程是近年出现的一种新颖软件开发范型，它借助于 Agent 技术来支持软件系统的工程化开发，被视为是支持复杂软件系统开发的一种重要方法和手段，受到了学术界和工业界的高度关注和重视。尽管在过去十年多的时间里面向 Agent 软件工程的研究取得了诸多进展，但其技术潜力尚有待进一步发挥，走向大规模工业化应用仍面临诸多问题和挑战。在分析面向 Agent 软件工程产生背景和技术特点的基础上，从方法、过程和工具三个不同的角度，综述了面向 Agent 软件工程的研究现状，识别和分析了它当前所面临的一组关键挑战，展望和讨论了其未来的研究方向。

关键词 Agent，多 Agent 系统，面向 Agent 软件工程

State-of-the-Art, Challenges and Perspectives of Agent-oriented Software Engineering

MAO Xin-jun

(Department of Computer Science and Technology, National University of Defense Technology, Changsha 410073, China)

Abstract Agent-oriented software engineering is a novel software paradigm that is considered as an important approach to supporting the development of complex software systems based on agent technology. Many attentions had been paid by researchers in academic literature and practitioners in industry literature. In the past ten years, great progresses of agent-oriented software engineering has been made. However, there are still a great number of problems that should be solved before it moves to industry application and its potentials should be exploited extensively. After investigating the background and characteristics of agent-oriented software engineering, the paper overviewed the state-of-the-art of the researches on agent-oriented software engineering, identified and analyzed a number of key challenges of such technology, outlooked and discussed the future directions from technique, process and tool viewpoints.

Keywords Agent, Multi-agent system, Agent-oriented software engineering

1 引言

回顾软件工程的发展历程，可以发现应用需求是推动技术发展的主要驱动力。一方面，软件系统特征和复杂性的变化将对已有的软件工程技术提出挑战，并促使新的软件工程范型的产生和发展。另一方面，新颖、有效的软件理论和技术往往推动计算机技术向新的应用领域发展和渗透，从而拓宽计算机的应用范围，导致软件形态和特征的进一步变化[1]。

随着计算机技术和网络技术的不断发展，当前以网络为核心的软件密集型系统（如部署在 Internet 上的软件）的形态和特征正发生着深刻的变化，具体表现为：

环境：软件系统驻留在动态、开放的环境中，如 Internet/Intranet、战场环境、物理设施环境等，环境变化不受系统的完全控制且无法事先确定，但同时又会对系统产生重要影响。系统与环境之间的边界模糊、不明确。

系统：软件系统规模超大[2]，系统动态、开放、异构、发散、跨域和跨组织；由于环境的不确定性、不可预知性以及软件需求的事先未知性和变化性，大量原先由设计人员在设计阶段做出的决策将改由软件系统在运行阶段来完成，即软件系统需具备自主性；由于环境的复杂性，软件系统需具备应对环境变化的能力，如自适应、自组织、自管理等。软件系统需要通过持续演化来满足系统的不间断运行、变更需求以及新技术的融入。

交互：系统与环境之间、系统与系统之间、系统内部的各个成分之间将进行多样化、持续、复杂的交互和协同，如 CPS 和物联网。由于系统的跨组织和自主性等特征，交互的过程和结果具有不确定、不可控等特点。环境和系统的复杂性将导致大量的交互需要由软件系统自身在运行时自主完成，即由软件系统来决定与谁、采用何种方式进行什么样的交互。

显然，软件系统的上述复杂性特征远远超出现有软件工程（如面向对象软件工程、基于构件软件工程）的抽象能力，系统特征的变化将作为一种重要的驱动力，促使软件工程甚至计算机科学范型的根本性变革[3]。为了应对上述挑战，近年来人们提出了一系列新的软件工程概念和技术，如网构软件（Internetware）[4]、主动对象和构件（Active Object and Component）、软件 Agent[5,6] 等等。其中，面向 Agent 软件工程引起了人们的极大关注和重视。

Agent 概念最初来自于 AI 领域，它是指驻留在特定环境

本文受国家自然科学基金（60773018），教育部博士点基金（20094307110007）资助。

毛新军 博士，教授，博士生导师，主要研究方向为软件工程、智能 Agent 理论和技术、自适应和自组织系统等，E-mail：mao.xinjun@gmail.com。

下能自主、灵活地执行动作以满足设计目标的行为实体。驻留性、自主性和灵活性是 Agent 的基本特征。驻留性是指 Agent 处于特定环境中，能感知环境的变化，并能通过自身行为影响环境；自主性是指 Agent 能根据其内部状态和外部环境决定自身状态，无需外界的干涉来决定和控制自身行为；灵活性是指 Agent 能与其它 Agent 实施复杂的交互和协同，行为具有自发性。早在 20 世纪 80 年代末，AI 领域的诸多学者对智能、理性 Agent 产生了浓厚兴趣，开展了智能 Agent 理论和技术的研究，取得了一系列的研究成果，如智能 Agent 的 BDI 理论框架和体系结构、以 AGENT-0 为代表的面向 Agent 程序设计语言、以 KQML 和 FIPA ACL 为代表的 Agent 通讯语言及其语义和语用等等，并将这一技术应用于诸如工业、电讯、航空等领域，取得了一些成功的案例。进入 20 世纪 90 年代中后期，随着 Agent 理论和技术研究与应用的不断深入，人们希望能够从一些特定的 Agent 技术和具体的应用案例中抽取出一般性的、具有普遍意义的思想、原理、原则、方法、过程和模型，提供一种系统、普适的方法以指导软件系统的工程化开发。在此背景下，以 N. R. Jennings 为代表的一批学者将 Agent 概念、理论和技术引入到软件工程领域，并与软件工程的思想、原理和原则相结合，提出了面向 Agent 软件工程概念和思想[5]。

面向 Agent 软件工程以 AI 领域的智能 Agent 理论和技术为基础，以自主 Agent 作为基本的概念、抽象和技术手段来开发软件系统，代表了一种新颖的软件开发范型，见表 1。

表 1　面向 Agent 软件工程与传统软件工程之间的对比

	面向 Agent 软件工程	传统软件工程
概念模型	Agent，多 Agent 系统等	过程、进程、对象等
计算模型	自主计算	过程调用、对象计算
交互模型	基于语义和语用的交互	基于语法的函数调用、消息传递、RMI，RPC
理论基础	言语行为理论、认知理论等	抽象数据类型理论、并发理论

由于 Agent 的驻留性、自主性和灵活性可有效应对环境、系统和交互的复杂性，因此面向 Agent 软件工程被视为支持复杂软件密集型系统开发的一种重要、富有潜力的技术手段[6]。一些应用实践展示了 Agent 技术在支持一类复杂系统开发方面的优势和潜力，如生产过程、面向服务计算、分布式网络管理等[7]，人们期望这一新颖软件工程技术能够应用于诸如过程控制、电讯电力、交通管理、信息系统、游戏娱乐、医疗护理、军事国防等复杂应用的开发。由于认识到这一技术的潜力，并对它寄予很高的期望，过去十年来学术界和工业界的专家、学者和工程实践人员针对面向 Agent 软件工程进行了深入的研究和实践，在方法学、程序设计、软件体系结构和模式、语言设施、支撑工具等方面取得了一系列的研究成果。但是，时至今日面向 Agent 软件工程并没有像人们所期望的那样，在应对复杂软件密集型系统的开发、缓解和解决“软件危机”中发挥出应有的作用，其思想和方法未能得到软件工程领域学者和工程实践人员的普遍接受，技术尚不成熟，未能在工业界得到广泛应用，至今基于面向 Agent 软件工程技术开发出来的软件系统并不多。导致这种状况的原因是多方面的，这也意味着面向 Agent 软件工程的研究面临着许多问题。

本文在系统综述面向 Agent 软件工程研究现状的基础上，深层次地识别和分析面向 Agent 软件工程研究存在的问题和面临的挑战，并展望和讨论进一步研究。本文第 2 节从方法、过程和工具三个视点概述面向 Agent 软件工程的研究现状；第 3 节分析当前面向 Agent 软件工程存在的问题和挑战；第 4 节讨论和展望进一步研究，最后总结全文。

2　研究现状综述

方法、过程和工具是构成软件工程的三个基本要素，任何软件工程范型均包含这方面的内容。图 1 从方法、过程和工具三个视点概括了当前面向 Agent 软件工程的主要研究内容。

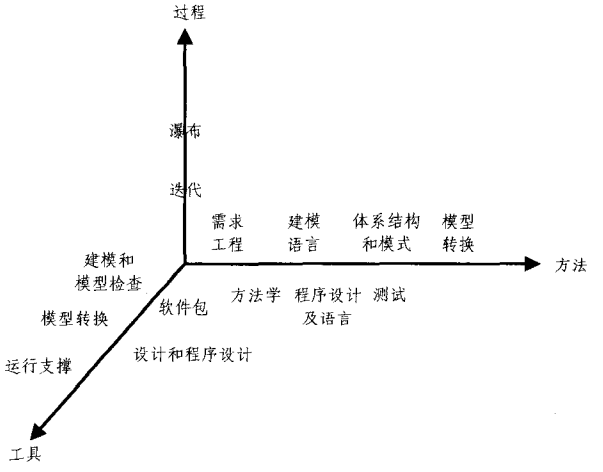

图 1　面向 Agent 软件工程的研究内容

(1)方法

方法是构成软件工程的技术手段。自 Wooldridge 提出面向 Agent 软件开发方法学 Gaia 以来，方法研究就成为面向 Agent 软件工程研究的主要内容[8]，具体包括：

1)需求工程

面向 Agent 的需求工程旨在研究如何利用 Agent 抽象来提取、导出、分析和描述系统的需求，包括功能性需求和非功能性需求。针对面向 Agent 需求工程特点[9]，该方面研究通常借助于认知科学、社会组织学等学科的知识和思想，并与面向目标的需求工程[10]紧密地结合在一起。代表性工作包括 i^*[11]，Tropos[12]，KAOS 等。i^* 引入了目标(Goal)、角色(Role)、Actor、Agent 等概念来表示系统需求，它区分软目标(Hard Goal)和硬目标(Soft Goal)，强化了对系统非功能需求进行建模。Tropos 区分了早期需求分析和晚期需求分析，通过早期需求分析可以很好地理解为什么需要软件需求，从而更好地导出软件系统的需求。由于当前越来越多的软件密集型系统实际上是社会组织系统在计算机世界中的映射，除了传统的功能性需求之外，非功能性的需求变得越来越重要，因此基于高层的 Agent 抽象有助于实现对这类复杂系统的需求进行自然建模和分析。

2)方法学

面向 Agent 的软件开发方法学旨在提供结构化的步骤和良定义的描述语言，以对系统进行分析和设计。21 世纪初期该方面研究非常活跃，至今人们提出了几十种面向 Agent 软件开发方法学。根据这些方法所依赖的技术背景及其差异性，可以将现有面向 Agent 软件分析和设计方法分为以下几个类别[1]：(1)基于对象技术，该类别的方法借助于面向对象软件开发方法，将 Agent 视为一种特殊的对象，通过对现有面向对象软件开发方法以及建模语言(尤其是 UML)的扩充来

支持对基于 Agent 系统进行建模、分析和设计，代表性工作包括：MaSE，ODAC，Massive，AOR，Prometheus 等。（2）基于知识工程，该类别的方法借助于知识工程（如人工智能、认知科学等）领域的概念、思想和技术对基于 Agent 系统进行建模、分析和设计。一种常用的方法是将 Agent 视为由各种认知部件所组成的意向系统，如信念、目标、意图、规划等，代表性工作包括：Tropos，DESIRE，MAS-CommonKADS，Styx，AAII 等。（3）基于组织思想，该类别的方法借助于组织学和社会学的思想，将基于 Agent 系统视为是一个组织或者社会，利用社会学和组织学中的抽象概念（如角色、组织、组织规则、职责、权利）和思想来对基于 Agent 系统进行描述、分析和建模，代表性工作包括：Gaia，AALAADIN，ODAM 等。文献[13,20]分析了面向 Agent 软件方法学的发展历程，评估和分析了现有工作的优势和不足，文献[14]介绍和分析了基于组织思想的面向 Agent 软件开发方法学。

3）建模语言

面向 Agent 建模语言借助于 Agent 抽象，提供了特定的语言符号来对系统的结构和行为等进行建模，目前代表性的面向 Agent 建模语言主要有三种：AUML[15]，MAS-ML[17]和 AML[16]。AUML 通过对面向对象建模语言 UML 的扩展来支持多 Agent 系统的建模。AML（Agent Modeling Language）是一个半形式化的可视建模语言，它引入了认知科学、社会组织学的诸多概念和思想，并通过对 UML 2.0 进行扩展来支持面向 Agent 的建模。MAS-ML 是对 UML 2.1 的扩展，它引入了诸多 Agent 概念和抽象（如角色、组织、规划等）来支持多 Agent 系统的建模。为了解决面向 Agent 建模语言的多样化问题，人们开展了语言和符号的统一化工作[18]。

4）程序设计及其语言

面向 Agent 程序设计及其语言旨在为多 Agent 系统的构造和实现提供程序设计的思想和语言设施。该方面的最初工作来自于 Shoham 的研究[19]，至今人们已经提出了十多个面向 Agent 的程序设计语言。根据这些语言实现技术的差异，可以将它们大致分为以下几类[1]：（1）基于 LISP 技术，这类面向 Agent 的程序设计语言采用 LISP 的语法形式，用类似于 LISP 的语句来表示 Agent 的内部组成成分以及 Agent 之间的交互通信。它们具有良好的理论基础和严格的语义定义并采用符号处理的方式来解释和执行所编写的程序，代表性工作包括：由 Shoham 提出的 AGENT-0、由 Thomas 提出的 PLACA、由 Winton H E Davies 和 Peter Edwards 提出的 A-GENT-K。（2）基于逻辑技术，这类面向 Agent 的程序设计语言采用各种形式的逻辑系统（如一阶谓词逻辑、时序逻辑）来表示和定义 Agent 的结构和行为。它们具有良好的理论基础和严格的语义定义。一般地，它们将 Agent 视为是一个定理证明器，将 Agent 的执行过程视为是一个定理证明过程。代表性的工作包括：由 Michael Fisher 提出的并发 METATEM、由 De Giacomo 和 Yves Lesperance 等人提出的 CONGOLOG、由 Koen V. Hindriks 等人提出的 3APL 等等。（3）基于对象技术，这类面向 Agent 程序设计语言利用面向对象的软件开发技术和软部件技术，通过对现有的面向对象程序设计语言（如 C++和 Java）进行扩充（包括语法扩充和语义模型扩充），为基于 Agent 系统的开发提供程序设计语言级的支持，代表性的工作包括：由 Agent Software 公司提供的 JAL（Jack Agent Language）、G. McCabe 和 Keith L. Clark 提出的 April、SLABSp 等等。

5）体系结构和模式

体系结构设计是软件设计的一个重要环节，设计模式和体系结构模式的识别、提取和分析有助于实现软件重用，这一思想在面向对象软件工程中得到成功应用。面向 Agent 软件体系结构和模式的研究大多借助于社会学和组织学的思想，将多 Agent 系统的体系结构视为是组织结构[22]。文献[23]分析了至今人们提出的各种多 Agent 系统体系结构范型和模式，包括层次式、联盟式、社会式、联邦式、小组式等等，每一种结构形式都有其优势和不足，适合不同的应用需求。文献[24]从社会组织学的角度提出和分析了多种形式的软件体系结构和模式，包括 structure-in-5、平坦、金字塔等等。

6）模型转换

基于 Agent 的模型转换旨在研究如何将模型驱动体系结构技术（MDA，Model-Driven Architecture）与 Agent 技术相结合，以提高面向 Agent 软件开发的效率和质量，是近年来面向 Agent 软件工程的研究热点。目前该方面的研究非常活跃，大部分的研究针对特定的方法学研究相应的模型转换技术。文献[25]提出了将基于 MAS-ML（Multi-Agent System Modeling Language）描述的平台独立模型（PIM，Platform-Independent Model）转换为基于 UML 描述的平台相关模型（PSM，Platform-Specific Model）的方法。文献[26]提出针对面向 Agent 软件开发方法 INGENIAS 的模型转换方法及其支撑工具。文献[27]基于 SMART 概念框架，提出了一种基于构件的面向 Agent 模型转换方法，以将高层的分析和设计模型转换为针对 JACK 的实现模型。针对现有面向 Agent 软件开发方法学只关注软件的分析和设计而忽视从设计到平台代码的生成这一问题，文献[28]提出了将基于 Tropos 方法学的软件设计模型转换为一类基于 Malaca 元模型的平台代码，包括 JADE 平台、FIPA-OS 平台等。

7）软件测试

测试是发现软件故障，确保软件质量的一种主要手段。由于软件 Agent 的行为自主性，面向 Agent 的软件测试有其特殊性和难度。尽管认识到其重要性，但是有关该方面的研究并不多。文献[29]针对多 Agent 系统提出了一种面向目标的软件测试技术，文献[30]针对 PASSI 方法学提出了一种包含多个层次的多 Agent 系统测试框架，并开发了相应的测试支撑工具。

8）其它

近年来，面向 Agent 软件工程日益与一些新颖技术相融合，从而为这些技术提供解决途径，包括：面向服务计算、语义 Web、对等计算、网格计算、Self-* 系统、自治计算、数据挖掘。围绕这些技术出现了一些新的研究方向，如基于 Agent 的数据挖掘、基于 Agent 的服务计算等等[7]。

（2）过程

过程定义了结构化的步骤和有序的软件活动以指导软件系统的工程化开发。当前面向 Agent 软件过程的研究与应用主要借助于传统的软件工程过程。绝大部分的软件开发方法学（如 Gaia，Tropos，MaSE 等）仍然基于瀑布模型来支持软件

开发，一些软件开发方法学（如 ODAM）则采用了迭代的软件过程模型[31]。从总体上看，当前面向 Agent 软件开发过程研究未能充分展示 Agent 技术的特点及其在解决复杂系统开发方面的优势。

（3）工具

面向 Agent 的 CASE 工具旨在为面向 Agent 的软件开发提供自动或半自动的支持。至今人们已经开发出了几十种面向 Agent 的 CASE 工具。尽管它们数量众多，功能、类型和特征各异，但是归纳起来，现有的面向 Agent 的 CASE 工具大致可分以下几类：（1）软件开发工具集，如 AgentTool，Agent-Builder，Zeus 等，它们通常包含一组工具以支持软件开发人员对目标软件系统进行建模、分析、设计、实现、部署和测试等等。许多工具通常集成在一个环境中，因而可以为软件开发人员提供一个较为系统和完整的开发支持。（2）基础设施和平台，它们提供了面向 Agent 软件系统运行所需的基础设施和基本服务，比如通信服务、安全服务、目录服务、移动服务等，如 Voyager、ADE、Grasshopper 等。（3）开发包和可重用库，如 JADE、Microsoft Agent 等。软件开发人员可以重用它们来创建多 Agent 系统[1]。

3 面临的问题和挑战

尽管面向 Agent 软件工程的研究引起了人们的极大关注，研究活跃并取得了许多成果，但是这一新颖软件开发范型的研究与实践仍然存在诸多的问题和挑战[40]。这些问题和挑战带来的直接结果是这一技术未能得到主流软件工程领域的学者和工程实践人员的普遍认可和接受；尚未能在工业范围内得到广泛应用；一些研究机构和企业推出了许多基于 Agent 的软件产品，但是它们鲜有获得高度评价。导致这种状况的原因是多方面的，如教育和宣传、新技术的认同等等。图 2 从方法、过程和工具三个不同视点识别了当前面向 Agent 软件工程研究面临的一组问题和挑战。

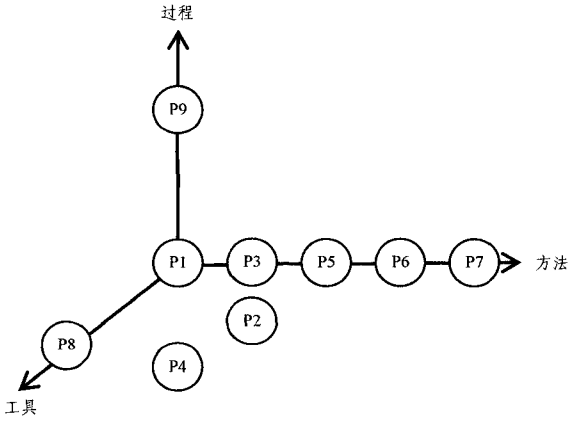

图 2 面向 Agent 软件工程面临的问题和挑战

P1：忽视软件工程关注点

软件工程的研究和实践非常关注软件开发的效率、成本和质量，这是成功开发一个软件系统的关键。例如，面向对象软件工程在支持软件系统分析、设计和实现的同时，借助于继承、封装等机制和技术来支持重用，以提高软件开发的质量和效率。当前面向 Agent 软件工程领域的许多研究人员来自于 AI 领域，他们往往从 AI 视点来认识面向 Agent 软件工程并开展该领域的研究工作，所取得的研究成果仅仅侧重于某些特定的技术环节，而忽视了多 Agent 系统软件工程化开发应关注的诸多要素。这必然导致技术的单一性，缺乏实用价值，影响工程实践人员对技术的接受和认可程度。例如，许多方法学仅仅关注分析和设计问题，没有考虑在分析和设计时所采用的技术手段是否有利于最终软件系统的构造，整个方法学技术框架是否有助于建立高质量的软件模型等。

P2：缺乏统一的软件 Agent 模型

面向对象软件工程采用单一的对象模型，即对象是属性和方法的封装。面向 Agent 软件工程中的软件 Agent 模型多种多样，包括反应型、认知型、知识型以及混合型等[1]。典型的软件 Agent 模型包括反应式和 BDI。显然，这种多样化的软件模型将给软件开发人员带来三方面的问题：第一，难以清晰地理解 Agent 软件模型；第二，当面临特定问题需要设计软件模型时，往往不知该选用何种模型来进行抽象和设计；第三，不同实现模型的软件 Agent 之间交互和互操作存在困难。显然，缺乏简洁、统一、有效的软件 Agent 模型是目前阻碍面向 Agent 软件工程走向大规模工业应用的主要因素之一。

P3：高层方法与底层实现脱节

一个完整的软件开发过程通常需经历分析、设计、实现和测试等阶段。一般而言，采用统一的抽象来贯穿这些阶段有助于减少变换，简化开发。例如，面向对象软件工程在分析、设计、实现和测试阶段采用统一的概念和抽象，确保不同阶段之间的自然衔接和过度。当前，面向 Agent 软件工程在分析阶段与设计阶段、设计阶段与实现和构造阶段往往基于不同的抽象和思想，出现高层方法与底层实现相脱节的现象，导致高层的分析或设计模型与底层的实现模型有较大的抽象差距，而这种差距的转换需要靠软件开发人员来完成。例如，许多方法学（如 Gaia，ROADMAP 等）在分析和设计阶段均基于组织学和社会学思想来分析和设计系统，建立系统的高层模型。而对于实现阶段可能采用的是基于认知思想的 Agent BDI 模型。对于如何将高层的组织模型映射为底层的 BDI 模型，这些方法既不关心也没有提供相关支持。这使得采用这些方法来进行面向 Agent 软件开发变得极为复杂和困难。

P4：互操作问题

面向 Agent 软件工程的方法、语言、工具和平台的多样化必然导致互操作问题。例如，由不同工具和平台（如 JADE，JACK）所构造的多 Agent 系统之间如何交换信息和数据，如何理解由不同面向 Agent 建模语言（如 AUML，AML）所描述的软件模型等等。这些问题在面向对象软件工程的发展早期同样存在，由于及时开展了技术标准化工作并取得进展，面向对象软件工程较好地解决了这一问题。无疑，一旦面向 Agent 软件工程在工程实践中得到推广和应用，互操作问题就会日益突出。

P5：缺乏程序设计方法学

任何软件工程范型都必须面对程序设计问题。面向对象软件工程的成功源自于其先期开展的是面向对象程序设计及其语言的研究与实践，如 Smalltalk，C++和 Java 等，并提出了相应的程序设计思想，告诉程序设计人员如何采用面向对象思想来构造软件系统。我们注意到，虽然早在 20 世纪 90 年代初 Shoham 就提出了面向 Agent 程序设计思想，并在随

后引发了面向 Agent 程序设计语言研究小高潮，但是随后由于遇到诸多的问题，面向 Agent 程序设计语言的研究未能取得实质性的进展。到了 20 世纪 90 年代中后期，该方面的研究几乎处于停滞状态。尽管至今人们已经提供了十多种面向 Agent 的程序设计语言，而且该方面的研究在 2005 年左右再次引起人们的关注和重视，但是有关面向 Agent 程序设计方法学的研究甚少，如何运用面向 Agent 程序设计范型来构造高质量的软件系统是当前面向 Agent 软件工程面临的一项重要挑战。

P6：程序设计语言的瓶颈

面向对象软件工程的成功首先源自于面向对象程序设计及语言，因为通过面向对象的程序设计，软件开发人员对面向对象思想有直观的理解和认识，这为他们接受和认可这一软件工程范型奠定了基础。缺乏简单、有效、可用和实用的面向 Agent 程序设计语言是面向 Agent 软件工程面临的一项重要挑战。现有面向 Agent 程序设计语言或者是一种实验性的语言（如 Agent-0），或者实际是一种不具有明显 Agent 技术特征的逻辑程序设计语言（如 CONGOLOG，Concurrent METATEM）及其变种，或者是对面向对象程序设计语言的扩充（如 JAL），难以充分展示 Agent 技术的特色和优势等。显然，要设计一种真正意义上的面向 Agent 程序设计语言需要从软件工程的视点权衡多方面的因素。

P7：缺乏验证和确认技术

质量是任何软件工程范型都必须关注的问题，提供有效的质量保证手段是实现技术走向大规模工业化应用的前提。面向对象软件工程不仅提供了分析、设计和实现的技术手段，还提供了与对象技术相适应的质量保证手段，如面向对象的软件测试。当前，面向 Agent 软件工程缺乏有效的验证和确认技术，尤其是面向 Agent 的软件测试技术。导致这种状况的一个主要因素是由于软件 Agent 自主性以及交互和协同不确定性给软件测试带来的挑战。

P8：缺乏有效的工具和平台

有效的工具和平台可以极大地提高软件开发的效率和质量，面向对象软件工程提供了一系列成功的软件工具和环境，如 Rational Rose，Eclipse 和 Visual Studio 等。至今人们开发了许多面向 Agent 的支撑软件工具和环境，但是在工具的可靠性、友好性、简单性和集成性方面有待进一步加强，还缺乏支持一些关键软件开发活动（如程序设计、软件测试、模型转换）的支撑软件工具。

P9：缺乏特定的软件过程

技术和过程的有效融合往往可以更好地发掘技术的潜力，面向对象软件工程引入了针对对象技术的软件开发过程 RUP。当前，面向 Agent 软件工程主要采用传统的软件过程模型，如瀑布模型、迭代模型等等。显然，这些模型用于解决规模和复杂性不高的软件系统是有效性的。但在运用 Agent 技术来应对大规模的复杂系统方面，面向 Agent 软件工程需要寻求新颖、有效的软件过程模型。

4 未来研究展望

上述挑战表明面向 Agent 软件工程的研究尚有许多问题有待解决，要真正让面向 Agent 软件工程在支持复杂系统开发方面发挥作用并得到大规模的应用必须在以下一些关键环节方面开展研究。

F1：程序设计的理论、方法、语言和工具

面向 Agent 程序设计已成为阻碍当前面向 Agent 软件工程发展与应用的一个主要瓶颈，其成功与否将直接检验面向 Agent 软件工程的技术潜力，也将对这一新颖技术能否为人们所接受并在工业范围内得到大规模应用产生重大影响。未来面向 Agent 程序设计需关注以下四个方面的内容。

面向 Agent 程序设计理论，需要研究支撑面向 Agent 程序设计的基础理论，从而为方法学和语言设施的设计奠定基础，包括类型理论、交互理论、语义基础等等。

面向 Agent 程序设计方法，需要研究特定于 Agent 技术的程序设计方法，从而为开展面向 Agent 的程序设计、构造高质量的软件系统提供方法学指导。例如，如何组织模块单元，如何有效实现重用，如何提高软件系统的可维护性等等。

面向 Agent 程序设计语言，需要研究支持面向 Agent 程序设计的语言设施。除了考虑面向 Agent 程序设计语言设施的语法、语义之外，还需要考虑语用、与分析和设计方法学之间的衔接、语言的表达能力和简洁性之间的权衡等一系列问题。面向组织的程序设计[42]和面向目标的程序设计[35]值得关注。

程序设计的支撑工具和环境，需要开发和提供类似于 Eclipse 这样的集成、开放、可靠、有效的面向 Agent 程序设计支撑工具和环境。

F2：验证与确认

质量保证是当前面向 Agent 软件工程的薄弱环节。面向 Agent 软件工程需加强面向 Agent 验证和确认技术的研究，以保证所开发软件系统的质量。

面向 Agent 的软件测试，需要研究针对面向 Agent 规约、设计和实现的软件测试技术，解决由于环境开放性以及 Agent 自主性、适应性等给软件测试带来的挑战，并以此为基础提供测试工具，为面向 Agent 的工程化开发提供支持。

面向 Agent 的模型检验，需要根据 Agent 技术特点研究相应的模型检验技术。

F3：面向复杂系统

一项技术的生命力在于它能解决其它技术不能或者不好解决的问题。多年来，面向 Agent 软件工程被视为在应对复杂软件系统的开发方面富有潜力。但是，至今这种潜力无论在技术层面还是在应用层面都未能得到很好的发挥。例如，许多面向 Agent 的软件方法学或者建模语言都无法支持开放的环境和动态的系统。因此，未来面向 Agent 软件工程需要加强以下几个方面的研究。

环境，已经成为理解、分析和构造系统的一个不可或缺的要素[39]。对于一些复杂的系统而言尤其如此。因此，需要针对开放环境的特点，开展环境建模、表示、分析、感知、系统与环境交互等方面的研究，并将它们与 Agent 技术紧密地结合在一起[31]。

适应性，由于驻留环境的动态、开放、不可控和不确定等特点，越来越多的复杂系统呈现出自适应的特征。自主的软件 Agent 在支持这类复杂系统开发方面有其独有的优势，因为自主性是自适应性的基础。因此，有必要针对自适应系统

的复杂性等特点，研究基于 Agent 的软件工程技术以支持这类软件系统的开发。

F4：技术标准化

技术标准化是推动技术走向成熟和应用的重要方式和手段。在过去十多年时间里，人们在 Agent 技术标准化方面做了很多的努力，有多个组织如 FIPA、OMG 和 ISO 等参与了该方面的工作，在软件 Agent 体系结构、Agent 通信语言、技术框架等取得了一些成果。但是总体上进展不大，近几年甚至停滞不前。这种状况也反映了 Agent 技术的研究仍然存在许多关键性的问题，这一技术距真正成熟尚有较大的差距。尽管如此，未来面向 Agent 软件工程的研究仍然需要在技术标准化方面做大量的工作，标准化的对象包括面向 Agent 的体系结构、建模语言、程序设计语言等等。

F5：软件过程模型

我们注意到，未来大规模、复杂软件系统将不再是一次性整体设计出来的，而是通过长期、持续演化而产生的。由于需求的不确定性、未知性和动态性以及系统的跨组织性，当前和未来许多复杂系统的开发无法采用传统的自顶向下、逐步求精的方式来完成，迭代开发的过程模型也无法有效地解决问题。在充分发挥 Agent 技术来解决复杂系统开发的同时，需要寻求相应的软件过程模型以更好地挖掘 Agent 技术的潜力，如研究自底向上和动态演化等多种方式相结合的面向 Agent 软件过程模型以及结合敏捷软件开发思想的过程模型等。

F6：形式化技术

形式化技术有助于提供准确和严谨的手段来支持系统的规约、分析、设计和实现。尽管目前面向 Agent 软件工程领域有大量的方法、语言等来支持多 Agent 系统的开发，但是很少有形式化技术的支持[41]。例如，已有的许多面向 Agent 建模语言和开发方法学借助于组织学和社会学的抽象和思想，并为此提供多样化的语言设施，但是没有一种方法准确地给出诸如组织、角色、规则等严格定义，以及严格的语言设施来支持系统的建模和分析。未来面向 Agent 软件工程的研究需要结合各种数学工具（如自动机、逻辑、Petri 网等），围绕规约、分析、设计、转换等软件开发问题，研究基于 Agent 的形式化技术，从而为面向 Agent 软件开发提供严格的技术和工具支持。

F7：集成与借鉴

软件工程已经经过了四十多年的发展历程，根据大量的应用和实践，总结出了许多普适性的原则、策略和经验，如模块化、高内聚、低耦合、重用、模式等等。面向 Agent 软件工程要真正在实践中发挥作用，在工业范围内得到大规模的应用，必须集成和借鉴软件工程中成功的经验、方法和技术。因此，未来面向 Agent 软件工程的研究需要关注面向 Agent 软件重用、面向 Agent 的软件设计模式、多 Agent 系统的软件体系结构风格和模式、基于 Agent 的模型转换等等。

结束语 面向 Agent 软件工程已经走过十多年的发展历程，尽管其研究在许多方面（如方法学、建模语言、支撑工具等）取得了不少进展，但在支持复杂软件系统的工程化开发方面仍未能像人们所预期的那样发挥作用，当前的研究和应用未能充分挖掘 Agent 技术的潜力。

近年来，越来越多的学者反思面向 Agent 软件工程的研究与实践，一些学者认为 Agent 技术的作用和功效被过分夸大了，另外一些学者则依然坚信 Agent 技术将在未来五到十年时间内在诸如面向 Web 服务、P2P 计算、语义 Web 和云计算等方面发挥关键性的作用和得到广泛的应用。文献[33，35]在分析多 Agent 系统开发现状的基础上，提出三个有待进一步研究的关键方向，即集成设计和实现的技术、对面向 Agent 程序设计语言进行扩展引入新的语言要素（如社会概念、环境等）、调试和验证技术，并展望了这三个方向的未来研究。文献[34]从三个不同的观察尺度即微观（micro）、宏观（macro）和 meso，分析了多 Agent 系统工程化开发面临的关键问题以及未来相应的研究方向。文献[38]强调了 Agent 技术与现有软件工程之间的集成和融合，以更好地促进多 Agent 系统技术在工业领域中的应用。

本文系统、简要地综述了面向 Agent 软件工程的研究现状，在此基础上从方法、过程和工具三个不同的角度识别和分析了面向 Agent 软件工程研究面临的问题和挑战。这些问题极大地影响了面向 Agent 软件工程的发展和应用，也是面向 Agent 软件工程走向成熟的主要障碍。最后，论文从程序设计、验证与确认、软件过程模型、形式化技术等多个方面讨论和展望了面向 Agent 软件工程的未来研究，以指导该领域的进一步研究工作。

参 考 文 献

[1] 毛新军. 面向主体软件开发[M]. 北京：清华大学出版社，2005

[2] Northrop L. Ultra-Large-Scale Systems: The Software Challenge of the Future[Z]. Software Engineering Institute, Carnegie Mellon University, 2006

[3] Zambonelli F, Van Dyke Parunak H. Towards a Paradigm Change in Computer Science and Software Engineering: a Synthesis [J]. The Knowledge Engineering Review, 2003, 18(4): 329-342

[4] 杨芙清. 软件工程技术发展思索[J]. 软件学报，2005，16(1)：1-7

[5] Jennings N R. On Agent-based Software Engineering[J]. Artificial Intelligence, 2000, 17(2): 277-296

[6] Jennings N R. An agent-based approach for building complex software systems[J]. Communication of ACM, 2001, 44(4): 35-41

[7] Luck M, Mcburney P, Preist C. A Manifesto for Agent Technology: Towards Next Generation Computing[J]. Autonomous Agents and Multi-Agent Systems, 2004, 9: 203-252

[8] Wooldridge, Jennings N R, Kinny D. The Gaia Methodology for Agent-Oriented Analysis and Design[J]. International Journal of Autonomous Agents and Multi-agent System, 2000(3)

[9] Cysneiros L M, Werneck V, Yu E. Agent/goal Orientation versus Object Orientation for Requirements Engineering: A Practical Evaluation Using an Exemplar[C]// Proc. of VIII Workshop in Requirements Engineering, 2005. Porto, Portugal, 2005: 123-134

[10] Van Lamsweerde. Goal-Oriented Requirements Engineering: A Guided Tour[C]// Proc. of 5th IEEE Int. Symp. on Requirements Engineering. 2001

[11] Yu E. Towards Modeling and Reasoning Support for Early-

Phase Requirements Engineering[C]// Proceedings of the 3rd IEEE Int. Symp. on Requirements Engineering. 1997

[12] Giunchiglia F, Mylopoulos J, Perini A. The Tropos software development methodology: Processes, Models and Diagrams[C]// Third International Workshop on Agent-Oriented Software Engineering. July 2002

[13] Zohreh Akbari O. A survey of agent-oriented software engineering paradigm: Towards its industrial acceptance[J]. Journal of Computer Engineering Research, 2010, 1(2): 14-28

[14] Mao Xinjun, Yu E, Organizational and Social Concepts in Agent oriented Software Engineering[C]// Proceedings of Agent oriented Software Engineering 2004, LNCS3382. Springer Verlag, 2004: 1-15

[15] Bauer B, Müller J P, Odell J. Agent UML: A Formalism for Specifying Multiagent Software Systems[J]. The Int. J. of Software Engineering and Knowledge Engineering, 2001, 11(3): 207-230

[16] Cervenka R, Trencansky I, Calisti M, et al. AML: Agent Modeling Language Toward Industry-Grade Agent-Based Modeling [C]// Proc. of 5th International Workshop AOSE, LNCS 3382. Springer Verlag, 2005: 31-46

[17] da Silva V T, Choren R, de Lucena C J P. MAS-ML: a multiagent system modelling language[J]. IJAOSE, 2008, 2(4): 382-421

[18] Padgham L, Winikoff M, Scott D, et al. A unified graphical notation for AOSE[C]// Proc. of 9th Agent-Oriented Software Engineering, LNCS. 2008: 116-130

[19] Shoham Y. Agent-Oriented Programming[J]. Artificial Intelligence, 1993, 60(1): 51-92

[20] Arazy O, Woo C C. Analysis and design of agent-oriented information systems[J]. The knowledge engineering review, 2002, 17(3): 215-260

[21] Ferber J, Gutknecht O, Michel F. From agents to organizations: An organizational view of multi-agent systems[C]// Proc. of IV AOSE, LNCS 2935. Springer-Verlag, 2003

[22] Lind J. Patterns in Agent-Oriented Software Engineering[C]// Proc. of 3rd International Workshop on Agent-Oriented Software Engineering, LNCS 2585. Springer, 2003: 47-58

[23] Horling B, Lesser V. A survey of multi-agent organizational paradigms[J]. Knowl. Eng. Rev., 2004, 19(4): 281-316

[24] Kolp M, Castro J, Mylopoulos J. A Social Organization Perspective on Software Architectures [C] // Proceedings of the STRAW'01. 2001: 5-15

[25] Alves B, Maria D, Torres V, et al. Developing Multi-Agent Systems Based on MDA[C]// Proceedings of the 17th Conference on Advanced Information Systems Engineering (CAiSE'05). 2005

[26] Pavón J, Gómez-Sanz J J, Fuentes R. Model Driven Development of Multi-Agent Systems [C] // Proc. of ECMDA-FA, LNCS 4066. 2006: 284-298

[27] Jayatilleke G B, Padgham L, Winikoff M. A model driven component-based development framework for agents[J]. International Journal of Computer Systems Science & Engineering, 2005, 4: 273-282

[28] Amor M, Fuentes L, Vallecillo A. Bridging the Gap between Agent-Oriented Design and Implementation Using MDA[C]// Proc. of Agent-Oriented Software Engineering, LNCS3382. Springer, 2005: 93-108

[29] Nguyen C D, Perini A, Tonella P. Goal-oriented testing for Multi-Agent Systems[J]. International Journal of Agent-Oriented Software Engineering, 2010, 4(1): 79-109

[30] Caire G, Cossentino M, Negri A, et al. Multi-agent systems implementation and testing[C]// Proceedings of Fourth International Symposium: From Agent Theory to Agent Implementation(AT2AI-4). Vienna, Austria(EU), April 2004: 14-16

[31] Mao Xinjun, Wang Ji. Engineering Adaptive Multi-agent Systems with ODAM Methodology [C] // Proceedings of Prima 2007, LNAI. Berlin Heidelberg: Springer-Verlag, 2009: 380-385

[32] Cossentino M, Gaud N, Hilaire V, et al. ASPECS: an Agent-oriented Software Process for Engineering Complex Systems[C]// Proc. of the 5th Agent Oriented Software Engineering Technical Forum(AOSE-TF5). 2007

[33] Bordini R H, Dastani M, Winikoff M. Current Issues in Multi-Agent Systems Development[C]// Proc. of ESAW 2006, LNAI 4457. Springer, 2007: 38-61

[34] Zambonell F, Omicini A. Challenges and Research Directions in Agent-Oriented Software Engineering[J]. Autonomous Agents and Multi-Agent Systems, 2004, 9: 253-283

[35] Winikoff M. Future Directions for Agent-based Software Engineering[J]. International Journal of Agent-Oriented Software Engineering, 2009, 3(4): 402-410

[36] Georgeff M. The Gap Between Software Engineering and Multi-Agent Systems: Bridging the Divide[J]. International Journal of Agent-Oriented Software Engineering, 2009, 3(4): 391-396

[37] Calisti M, Rimassa G. Opportunities to Support the Widespread Adoption of Software Agent Technologies [J]. International Journal of Agent-Oriented Software Engineering, 2009, 3(4): 411-415

[38] Weyns D, Helleboogh A, Holvoet T. How to get multi-agent systems accepted in industry[J]. International Journal of Agent-Oriented Software Engineering, 2009, 3(4): 383-390

[39] Weyns D, Parunak H V D, Michel F, et al. Environments for Multiagent Systems State-of-the-Art and Research Challenges [C]// Proc. of E4MAS 2004, LNAI 3374. 2005: 1-47

[40] DeLoach S A. Moving Multi-Agent Systems From Research to Practice[J]. International Journal of Agent-Oriented Software Engineering, 2009, 3(4): 378-382

[41] Ahmad R, Rahimi S. Motivation for a new formal framework for agent-oriented software engineering[J]. International Journal of Agent-Oriented Software Engineering, 2009, 3(2/3): 252-276

[42] Wester-Ebbinghaus M, Moldt D, Reese C, et al. Towards Organization-Oriented Software Engineering[C]// Software Engineering Konferenz 2007 in Hamburg: SE'07 Proceedings. Volume 105 of LNI., GI(2007). 2007: 205-217