面向 agent 软件开发方法比较与应用原则*)

郭 峰 姚淑珍
(北京航空航天大学计算机学院 北京100083)

摘 要 对面向 agent 软件开发方法的起源、研究现状和发展趋势进行了综述和讨论,介绍了部分典型方法,首次提出了 agent 的强、亚和弱自主性的定义。根据各种方法中对 agent 交互协议的不同描述方式,定义了显式协议和隐式协议,从四个方面对现有的典型方法做了比较,并分析了本文提出概念的应用原则。由于面向 agent 软件开发方法的研究仍然处于探索阶段,这些概念有助于研究人员理解掌握面向 agent 软件开发方法,对于该领域的发展具有积极的意义。

关键词 面向 agent 软件开发方法,自主性,交互协议

Comparing Agent Oriented Methodologies and the Application Principles

GUO Feng YAO Shu-Zhen
(School of Computer Science and Engineering, Beijing University of Aeronautics and Astronautics, Beijing 100083)

Abstract This article surveys and discusses the origination, the state of the art and the development trend of agent-oriented methodologies. It introduces the current typical agent oriented methodologies. For the first time it proposes the strong, sub and weak autonomous of agent, according to various kinds of methods used to model agent interaction protocol defines the explicit protocol and implicit protocol, and analyzes the application principles of those concepts. For the research of this area still at the stage of exploration, those concepts can help researchers understand and master agent oriented methodologies and have positive effect for the development of this area.

Keywords Agent oriented software methodologies, Autonomous, Interactive protocol

1 引言

近年来,在软件工程领域出现了一个新的研究分支——面向 agent 的软件工程(Agent Oriented Software Engineering,简称 AOSE)。面向 agent 软件工程是结合了 agent 技术和软件工程思想的一个新的研究方向,目前的主要研究内容是面向 agent 的软件开发方法 AOSE 的出现是 agent 技术发展的必然结果,agent 来源于分布式人工智能。但是在人工智能领域,agent 的研究偏重理论,对于复杂系统,用 agent 方法求解与如何设计和实现 agent 是两个不同的问题[1]。由于缺乏第二方面的研究,虽然 agent 技术具有非常吸引人的潜在优势,但在实践方面,agent 的优势并没有完全展现。agent 通常工作于复杂不确定的计算环境中(如自治、异构、动态的开放网络环境),agent 所需要完成的任务比传统软件系统更为高级、灵活和智能,多 agent 系统的组织关系也较为复杂[2],要构建正确、可靠、健壮的 agent 系统,迫切需要面向 agent 软件工程方法的支持。

软件工程是由于软件危机的出现而形成的一个研究领域,软件工程人员不断努力研究新的方法和工具以提高软件开发效率和性能,从上世纪六、七十年代的结构化编程,到八、九十年代的面向对象方法以及构件技术、设计模式等都是这些努力的结果。尽管面向对象技术发展至今,已趋于成熟并得到广泛应用,成为主流软件工程技术,但在描述动态、分布和不确定系统时,仍有许多不足之处,因此面向对象的概念也在不断发展,最显著的是面向对象和并发结合而形成的并发对象(或称主动对象)。近年来,由于 agent 研究的蓬勃发展,软件工程研究人员发现 agent 的概念进一步丰富了并发对象的内涵,因此对 agent 技术寄予厚望,将 agent 视为继面向对象之后的一种新的软件开发范型。一些学者(如 Jennings[3])认为,基于 agent 的软件开发方法未来很可能会成为象面向对象方法一样的主流软件工程方法。

本文第1节介绍了面向 agent 软件开发方法领域的研究现状,第2节介绍了八种典型方法,并提出了对 agent 强、弱和亚自主性以及显式协议和隐式协议的定义,从四个维度比较这些典型方法,第3节提出了本文研究成果的应用原则,对研究和工程人员具有一定的指导意义。最后总结全文并对未来的发展趋势作了展望。

2 相关研究现状

1996、1997是 AOSE 形成时期[4,5],之后各种具体的面向 agent 的软件开发方法大量出现。对于这些方法主要有两个分类标准,一是描述手段的精确程度上可以分为形式化方法和非形式化方法,二是从方法的来源上可以分为扩展面向对象方法和扩展知识工程方法。在目前的方法中,以扩展面向对象方法为主,非形式化方法多于形式化方法。从应用情况来看,目前虽然有很多面向 agent 的软件开发方法,但是还没有成熟实用的方法出现,总体上看,面向 agent 软件工程方法的研究与应用仍然处于探索阶段。

在目前面向 agent 软件开发方法研究中,标准化方面的工作值得关注,这一工作主要是基于 UML 开展的,当前有两个标准化组织 OMG 和 FIPA 正在从事这一方面的研究。他们都建议用 UML 描述 agent 系统的完整的生命周期,并且

*)基金项目:航空科学基金分布计算 MAS 模型及分析技术研究(01F51052)。郭 峰 博士研究生,研究方向:软件工程、Petri 网理论、多 agent 系统;姚淑珍 教授,研究方向:软件工程、Petri 网理论、多 agent 系统。

出现了 AUML[6]，AOR[7]，MESSAGE/UML[8]等扩展 UML 的方法。UML 作为准建模语言，在面向对象领域获得广泛应用，作为面向对象语言的集大成者终结了面向对象领域的方法大战。但是由于 agent 本身的特点和目前的研究现状，基于 UML 的面向 agent 的软件建模方法短期内还不能象基于 UML 的面向对象建模方法一样得到广泛的接受，面向 agent 的方法之战才刚刚开始。

3 方法介绍与比较

agent 技术已经在众多领域获得应用，近年来，面向 agent 软件开发方法的研究方兴未艾，成为一个具有广泛应用前景的新的研究领域，无论是研究人员还是工程人员，对现有方法有一个基本了解都是非常必要的。因此本文选择了八种比较典型的方法加以介绍，并进行比较。

3.1 典型方法介绍

GAIA[9]方法是一种通用的 agent 系统建模方法，既支持对 agent 社会性质的分析，又支持对单个 agent 性质的分析。该方法基于角色的思想，即每个系统都是由若干相互通信的角色组成，而每个 agent 都是承担一个或多个角色的软件实体。GAIA 方法分成两个阶段，分析阶段和设计阶段。分析阶段根据系统的需求说明建立角色模型和交互模型。该方法在较高抽象层次上描述多 agent 系统，系统分析和设计完成之后，设计模型与通常意义上的设计模型不同，还不能直向实现转换，还需要借助面向对象技术对设计模型进一步细化。GAIA 不针对具体的实现平台和 agent 结构，不考虑具体的实现技术，该方法不适用于开放系统，并且不支持继承性。

kinny 提出一种基于 OMT(object modeling technology)的多 agent 系统建模方法 AAII[10]，是特定于 BDI 结构 agent 的软件设计方法，该方法从宏观和微观两个层次来描述多 agent 系统，宏观层次包括：agent 模型和交互模型，微观层次包括信念模型，目标模型和规划模型。AAII 以面向对象的图形化方法描述 agent 的信念、目标和意图等心智概念，但系统最终在陈述式的 DMARS 语言上实现，从设计模型到实现的转化必然存在困难。该方法支持 agent 的继承性。目前对 agent 系统继承性的研究还是很初步的，Crnogorac[11]。初步控讨了 AOP(agent oriented programming)中的继承性，使 agent 的对象属性研究作为面向 agent 程序设计方法学的一个重要方面得到重视，其中的 agent 模型是 BDI 结构的，agent 内部控制流是顺序的，在程序设计的层次上研究 agent 心智概念的继承性，AAII 首次在面向 agent 软件分析与设计层次上引入了继承性的研究，但是对 agent 继承性的研究仍然限于 agent 认知特性的继承性。

G_net 是一种面向对象 petri 网，Haiping Xu 提出了基于 G_net 描述和分析多 agent 系统的方法[12]。该方法中在 G_net 模型中引入了知识、目标和规划的概念，agent 通过规划器决定采取的动作，在规划过程中利用知识、目标和规划模块选择下一步动作，并利用 petri 网工具 INA 验证模型的性质，如无死锁，交互协议的可跟踪等。该方法的 agent 概念是 BDI 与对象的混合体。在 agent 模型中借鉴了 AAII 方法的术语，但是 G_net 更加侧重研究 agent 作为一种特殊对象所具有的特点和性质，G_net 与面向对象结合紧密，强化了对象的自主控制能力，但是没有探讨如何描述 agent 的心智概念。G_Net 中首次研究了 agent 的继承异常问题，这对面向 agent 软件开发方法的研究具有重要的意义，说明对 agent 继承性的研究不再仅限于 agent 认知特性的继承。

MULAN[13]方法是基于 reference net 的方法，reference net 也是一种面向对象 petri 网模型，其主要思想是“网中有网”，也就是说 petri 网的 token 可以是 petri 网，agent 本身用 petri 网结构描述，包括了表示知识的库所和表示推理的变迁，agent 的行为又称为协议网的 petri 网描述，agent 在运行过程中根据知识库和接收到的消息确定采取的动作，对于这个选择过程在描述 agent 的 petri 网结构作为一个变迁出现，但是没有表示该变迁的语义，同时 agent 的知识退化为一个 token。所以，MULAN 对于 agent 结构的描述过于简单，只能起到示意作用。

d'Incerno 与 Luch[14]以 Z 语言作为多 agent 系统的形式化工具，定义了基本的系统类框架，该方法将多 agent 系统中的实体由下至上分成四个层次：环境实体，对象，agent 和自主 agent。每个层次是下一层次实体的增强，这些实体都用 Z 语言进行描述。它们的工作在成功描述 agent 静态结构的同时，反映出 Z 语言不宜描述 agent 之间的交互活动，它只能描述某个事件的发生对 agent 状态的影响，不能描述系统在通信活动中体现出的性质，如无死锁，活性等[15]。

MaSE[16]的显著特点是以状态图描述 agent 的行为，这些状态最终转化为实现代码中组成 agent 的类。该方法也分成两个阶段，在分析阶段，从系统最初需求获得目标层次模型，根据用例获取并发任务模型和角色模型，在设计阶段，从角色模型和并发任务模型构造 agent 类模型，从 agent 类模型和并发任务模型设计 agent 的内部模型，最后得到系统的配置模型。MaSE 的会话和并发任务都使用 UML 的状态图描述，该方法还借鉴了 UML 的用例图、顺序图和配置图等。该方法代表了一种新的趋势，多 agent 系统的构造强调 agent 之间的协作，对于 agent 内部弱化其推理能力，这种方法具有普适性。

AUML[6]可以说是一类方面的代表，它们都扩展 UML 对多 agent 系统建模，建模的重点是多 agent 的组织结构和 agent 之间的交互。AUML 起初为描述 agent 交互协议扩展了 UML 的顺序图，并且扩展了 UML 中的角色概念，允许 agent 承担多个角色。但是，对于 agent 个体结构的描述没有涉及，因此初期阶段的 AUML 对是不完整的。目前，出现了两种用于描述 agent 个体结构的方案[17,18]。由于和 agent 相关的两个标准化组织 OMG 和 FIPA 都建议用 UML 描述 agent 的完整生命周期，因此，扩展 UML 的方法值得关注。

MAS-CommonKADS[19]是一种扩展知识工程的方法，同时融合了面向对象方法和协议工程的成果。该方法包括七种模型：agent 模型、任务模型、专家模型、组织模型、协作模型、通信模型和设计模型，通过这些模型，从不同的角度比较完整地描述了多 agent 系统。agent 的概念从知识工程的角度容易解释和描述，而构造 agent 更需要软件工程领域的知识经验，这两者的结合是有益的，但是对于实践人员而言，掌握知识工程是困难的，因此，基于知识工程的方法从实用的角度上不如基于面向对象的方法。

面向 agent 的软件开发方法还有很多，限于篇幅，本文不一一介绍，读者可以参考文[20,21]等作为补充。

3.2 典型方法比较

在这些文献中，对面向 agent 软件开发方法通常按照两种标准分类介绍，一是从描述手段的精确程度上可以分为形式化方法和非形式化方法，二是从方法的来源上可以分为扩展面向对象方法和扩展知识工程方法。本文认为，这两种分类方法是必要的，但是不利于澄清 AOSE 研究中存在的争论，因此本文认为有必要增加对现有方法进行比较的难度。

Wooldrige[21]提出的 agent 弱概念和强概念的观点获得普遍认可，但是对于 agent 的自主性，只是强调 agent 运行时

对自己的行为和内部状态有一定的控制权。这种定义对于描述和实现 agent 留有很多余地，由此产生一些争论，如下列四点：

1. agent 行为的生成方式：agent 的行为包括本身的事务处理和交互行为，关于 agent 行为的生成有两种观点：自发形成或预先设定。

2. agent 的继承性：有些方法对于继承性持否定态度，如 Gaia，有些方法则持肯定态度，如 G_net。

3. agent 计算粒度：通常认为 agent 是粒度比较大的计算实体，但是有些方法实际上 agent 的粒度与对象的粒度相差无几。

4. agent 数量：在早期的研究中，通常认为系统中 agent 的数量不能太多，有些方法对 agent 的数量没有限制。

这些争论对于面向 agent 软件开发方法的设计，对于工程人员选择适当的方法都具有很大的影响，但是现有的文献对面向 agent 软件开发方法的比较，没有对这些争论进行分析。本文认为这些争论的根源在于不同领域对于 agent 的自主性具有不同的需求，因此，本文提出按照 agent 自主性强弱对面向 agent 软件开发方法进行分类的原则。

定义1(agent 的强自主性) 某些 agent 具有动态规划能力，表现出完全自发的行为，这种自主性为强自主性。

动态规划属于人工智能的研究范畴，即如何用 agent 求解，而不是如何构造 agent 的问题，重点在于 agent 决策，协作理论的研究。强自主性出现在 agent 研究的早期，但是在目前 AOSE 研究中，基本上不采用强自主性的方式描述 agent。

定义2(agent 的亚自主性) agent 的行为用离线设计的规划预先设定，agent 在运行时根据外界和内部的状态选择规划执行，这种自主性称为亚自主性。

亚自主性也是一种集中控制模式，需要 agent 具有一定的推理能力。G_net 虽然用 petri 网描述 agent 的规划器，但规划器中包含了代表推理动作的转移。AAII 方法描述了 agent 的信念、目标和规划，agent 基于实现平台提供的推理引擎进行推理。

定义3(agent 的弱自主性) agent 的行为由多个表示状态变化的并发模型组成，这些模型在运行时对应为独立运行的线程，这种自主性称为弱自主性。

具有弱自主性的 agent，更接近并发对象，但是并发对象是被动提供服务的，而 agent 内部的线程具有主动的事务处理能力。如 GAIA 用角色的责任描述 agent 的行为，MaSE 用并发任务描述 agent 内部的并发状态变化。

在三种 agent 自主性定义的基础上，我们可以对 AOSE 研究中存在的争论给出合理的解释。上述四点争论是密切相关的，自发行为的生成需要 agent 具有较强的计算能力，也就是较大的粒度，如果 agent 的粒度比较大，自然不需要太多的数量，同样继承也就难以实现。因此，强自主性 agent 通常具有行为自发生成，计算粒度大，系统中 agent 数量少，不支持继承性的特点。亚自主性 agent 的行为介于自发和预先设定之间，计算粒度中等，系统中 agent 的数量也不会很多，并开始支持继承性。弱自主性 agent 的行为是预先设定的，计算粒度小，agent 数量没有限制，支持继承性。

因为目前还没有其他研究人员提出对 agent 自主性进行分类的思想，现有的方法与本文的观点并不完全一致，比如 GAIA 方法实际上采用了弱自主性描述 agent，但在定义 agent 时仍然基于强自主性，这就造成虽然 agent 只具有弱自主性，但这种方法仍然对继承性持否定态度。

除了自主性外，agent 与对象的本质区别是其社会性，agent 的社会性基于 agent 之间的通信实现。agent 和对象都是基于消息进行通信的，但是 agent 通信消息的语义是基于言语行为理论定义的，而对象消息仅表示方法调用，agent 之间的通信也反映了 agent 的自主性。在目前的 agent 通信理论研究中，agent 交互协议的概念已经被广泛接受，因此，交互协议描述与实现的方式也是面向 agent 软件开发方法研究的一个重要问题。

交互协议本质上是多 agent 系统中的社会规范，约束 agent 的行为。社会规范的形成有两种方式：离线设计方式和自然形成方式[23]。前者是由设计者预先设计出合理的社会规范，再将其植入 agent 个体之中，后者指 agent 通过大量的交互，能够自适应、自组织地形成并接受某种有效的社会规范。前者一般被工程性系统所采用，多 agent 系统建模领域通常也采用这种方式。但是对于离线设计方式，在其应用方式上也有所不同，目前还没有文献深入讨论该问题，本文引入下面的定义进行区分。

定义4(显式协议) 交互协议的描述有与之直接对应的实现结构，这种协议称为显式协议。

如在 MASE 方法的设计阶段，交互协议用状态图描述，在生成实现代码时转换为协议类。

定义5(隐式协议) 交互协议的描述没有与之直接对应的实现结构，交互协议的实现策略隐含在 agent 的功能描述之中，这种协议称为隐式协议。

如 G_net[12]没有描述 agent 交互协议，通过规划器控制 agent 的交互行为满足交互协议的约束。

对于交互协议的描述与分析，主要考虑两个问题，一是死锁问题，二是交互协议的一致性验证问题。不同类型的协议对这两个问题有不同的解决方式。在隐式描述的情况下，这些问题一般要在运行阶段才解决，在显式描述的情况下，通常可以在设计阶段解决。

综上所述，我们认为研究人员设计一种新的或者工程人员在实践中选择面向 agent 的软件开发方法，需要考虑的最为重要的四个因素是方法的精确程度、来源、自主性的强弱以及交互协议的表示方式，因此我们根据这四个因素比较现有的典型方法，如表1所示(其中 OO 为 object oriented 的缩写，KE 为 knowledge engineering 的缩写)。

表1 面向 agent 软件开发方法比较

	形式化	来源	自主性	协议表示
G_Net	Y	OO	亚	隐式
kinny	N	OO	亚	隐式
Gaia	N	OO	亚	显式
MaSE	N	OO	弱	显式
MULAN	Y	OO	亚	隐式
Z	Y	OO	亚	隐式
AUML	N	OO	弱	隐式
MAS-CommonKADS	N	KE	亚	显式

4 应用原则

由于 agent 技术的优势，目前很多工程人员尝试采用新的技术，同时，不同领域应用系统的开发也需要针对性的方法学的支持。工程人员和研究人员在决定采用 agent 技术之后马上会遇到如何表示 agent 的问题。本文介绍面向 agent 的典型方法并定义 agent 的强、亚和弱自主性以及显式协议和隐式协议的目的就是为面向 agent 软件开发方法的研究人员在进行方法学研究中确定 agent 模型和实际工程人员在实际应

用项目的开发中回答如何表示 agent 的问题提供依据。

确定了表1的四个因素的取值，也就在一定程度上确定了如何表示 agent。但是不同的应用领域具有不同的特点和需求，因此本文提出了一些与领域特点相关的应用原则，如图1所示。

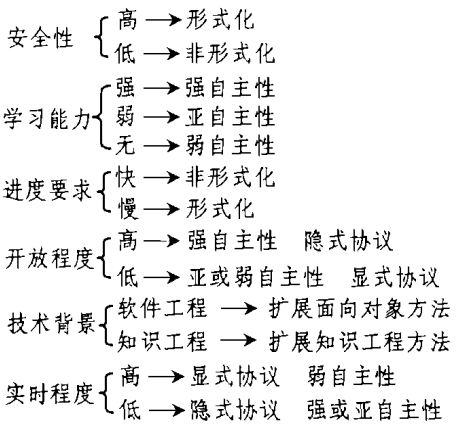

图1 应用原则

在实际系统开放或方法设计中，应综合考虑应用领域的特点，参考本文提出的应用原则，选择适当的方式描述 agent。

总结与展望 本文对 AOSE 的起源，研究现状和典型方法作了简单介绍，首次提出了 agent 的强、亚和弱自主性的定义，讨论了 agent 交互协议的不同描述方式，并根据四个主要因素比较了现有的典型方法，这些既可以帮助软件开发人员对面向 agent 软件开发方法有更深入的了解，可以指导面向 agent 软件开发方法的设计，同时也可指导具体系统的实现，对于 AOSE 和 agent 技术的发展具有积极的意义。

AOSE 的出现只有几年的时间，仍然处于探索阶段，但是发展的势头迅猛，在未来的几年里，还会有更多的面向 agent 的软件开发方法出现。agent 作为一种新的软件开发范型，已经在某些领域获得成功应用，在目前计算机科学领域中，agent 概念的出现频率非常高，这说明 agent 已经不仅仅是一种具有吸引力的概念，其应用价值逐渐获得认可，越来越多的研究人员加入这一领域，因此，我们对于 agent 技术的发展持乐观态度。

参 考 文 献

1 陆汝钤. 知识科学与计算科学. 清华大学出版社，2003
2 樊晓聪. 软件 agent 理论及其方法学的研究：[南京大学博士论文]. 1999
3 Jennings N R. On agent-based software engineering. Artificial Intelligence，2000
4 Towards a practical methodology for agent-oriented software engineering with c++and java
5 Wooldridge M. Agent-based software engineering. 1997
6 Bauer B，Muller J，Odell J. Agent UML：formalism for specifying multiagent interaction. 2000
7 Wagner G. The agent-object-relationship metamodel：Towards a unified conceptual view of state and behavior. 2002
8 Caire C，et al. Agent oriented analysis using MESSAGE/UML. 2001
9 Wooldridge M，Jennings N R，Kinny D. The Gaia Methodology for Agent-Oriented Analysis and Design. In Journal of Autonomous Agents and Multi-Agent Sytems，2000，3(3)：285～312
10 Kinny D，Georgeff M，Rao A. A methodology and modelliing technique for systems of BDI agents. 1996
11 Crnogorac L，Rao A，Ramamohanarao K. Analysis of Inheritance Mechanism in Agent Oriented programming. IJCAI97，1997. 647～652
12 Xu Haiping，Shatz S M. A Framework for Modeling Agent-Oriented software[A]. In：Proc. of the 21th Intl. Conf. on Distributed Computing Systems(ICDCS-21)[C]，2001
13 Köhler M，Rolke H. Towards a Unified Approach for Modeling and verification of Multi Agent Systems，2001
14 Luck M，d'Inverno M. A formal framework for agency and autonomy. In：proc. of the First Intl. Conf. on Multi-Agent Systems (ECMAS-95)，San Francisco，CA，June 1995. 254～260
15 刘璜. 面向 agent 的软件需求分析：[中科院博士论文]. 2000
16 Deloach S A，Wood M F. Sparkman C H. Multiagent systems Engineering. The International Journal of Software Engineering and Knowledge Engineering，2001，11(3)
17 Bauer B. UML Class Diagrams revisited in the context of agent-based systems. In：M. wooldridge，P. Ciancarini，G. Weiss，eds. Proc. of Agent-Oriented Software Engineering(AOSE 01)，number 2222 in LNCS，Montreal，Canada，Springer-Verlag，2001. 1～8
18 Hughet M-P. Agent UML Class Diagrams Revisited. In：Proc. of Agent Technology and Softwarwe Engineering(AgeS). Bernhard Bauer，Klaus Fischer，Jorg Mueller and Bernhard Rumpe(eds)，Erfurt，Germany，Oct. 2002
19 Iglesias C A，et al. Analysis and design of multiagent systems using MAS-CommonKADS. In：AAAI'97Workshop on Agent Theories，Architectures and Languages，Providence，RI，ATAL，1997
20 Iglesias C A，Garijo M，Gonzalez J C. A Survey of Agent-Oriented Methodologies. In：J. P. Muller，M. P. Singh，A. Rao. eds. Intelligent Agents V(ATAL'98)，LNAL 1555，Springer-Verlag，Berlin，Germany，1999. 317～330
21 Wooldridge M，Ciancarini P. Agent-Oriented software Engineering：The state of the Art. In：P. Ciancarini，M. Wooldridge，eds. Agent-Oriented Software Engineering. Springer-Verlag Lecture Notes in AI Volume 1957，Jan. 2001
22 Wooldridge M，Jennings N R. Intelligent agents：Theory and practice[J]. The Knowledge Engineering Review，1995，10(2)：115～152
23 王栩. Agent 通讯系统理论及组织结构研究：[中科院博士论文]. 2000

(上接第178页)

操作系统进行安全增强的概念，系统地分析了此方法的优势和不足之处，并且从设计和实现的角度出发提出了相应的策略。此方法是一个完整的安全体系，它考虑到了系统安全所需的大多方面，诸如访问控制、标识与鉴别、审计、身份认证等。在难以改造内核或对灵活性要求高的安全应用中，本文所描述的策略不失为一上选。

参 考 文 献

1 Bell D E，LaPadula L J. Secure Computer Systems：Mathematical Foundations：[Technical Report M74-244]. The MITRE Corporation，Bedford，Massachusetts，May 1973
2 Sandhu R S. Role-Based Access Control. Advances in Computers，Academic Press，1998，46：237～286
3 Portable Applications Standards Committee of the IEEE Computer Society. Draft Standard for Information Technology-POSIX-Part 1：System Application Program Interface (API)-Amendment：Protection，Audit and Control Interfaces. IEEE Draft P1003.1e，1997
4 Hofmeyr S A，Forrest S，Somayaji A. Intrusion detection using sequences of system calls. Journal of Computer Security，1998，6：151～180