Agent 互操作性研究*)

贾志勇 景广军 谢 立
(南京大学软件新技术国家重点实验室 计算机科学与技术系 南京210093)

Research on Agent Interoperability

JIA Zhi-Yong JING Guang-Jun XIE Li
(State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210093)
(Department of Computer Science and Technology, Nanjing University, Nanjing 210093)

Abstract Agent technology provides a flexible and uniform framework for Web and distributed applications, and interoperability is one of the most critical problems for agent technology to become mature and practical. This article analyzes the interoperability problem of Agent deeply. It first discusses the main research work about agent interoperability, namely OMG's MAF specification and the set of specifications of FIPA, and points out their drawbacks. Then a research model covering the main aspects of interoperability in Agent environment is brought up, and a reference model for agent system is also provided, which fully corresponds with the research model for agent interoperability. The two models have nice universality and adaptability, and provide a practical reference for building an Agent system with good interoperability.

Keywords Agent, Interoperability, Agent system model

1 引言

Agent 是一种具有自主性、交互性、移动性和智能性的软件主体[1,2]。Agent 的想法和研究起源于人工智能、人机界面设计和面向对象编程，它提供了一种新的分析、设计和实现复杂软件系统的方法和一个通用、灵活的分布式计算模式[3~5]。

从应用的角度来看，Agent 特别适合网络环境下的 Internet 应用、复杂企业计算和移动计算，而分布、异构和动态则是这些应用最鲜明的特征和需要首先面对的问题。此外，由不同开发者完成的各类 Agent 系统也迫切需要能够彼此相互操作。因此，作为用来解决这些问题的主要手段的互操作也就自然成了决定 Agent 技术能否走向成熟和实用的关键环节之一。简单地讲，Agent 互操作就是用来解决整个 Agent 计算环境下各个组成部分(不同的 Agent 平台、不同的自主 Agent)之间的相互联系和协作问题的。

本文对 Agent 互操作性进行了深入的探讨，在对当前该领域主要的一些研究工作深入分析的基础上，给出了关于 Agent 互操作的研究和实现模型，这些模型较全面地涵盖了 Agent 环境下互操作的各个方面。

2 当前关于 Agent 互操作性的主要研究工作分析

与 Agent 其它方面的研究相比，有关其互操作性的研究相对来说比较集中，主要集中于 OMG 的 MAF 和 FIPA 的一组规范。

2.1 MAF

MAF(Mobile Agent Facility)[6]是 OMG 为了解决 Agent 系统之间的互操作而提出的一个规范。它提供了一个概念性的框架，通过定义一组 Agent 标准化的接口来规范不同 Agent 系统之间对 Agent 的定位、查找和管理。

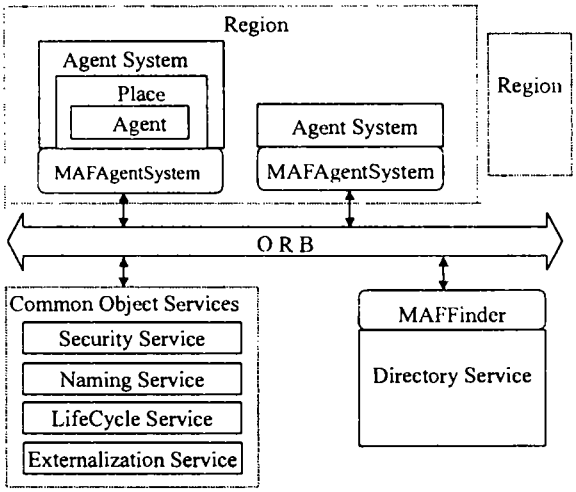

图1 MAF 概念框架

如图1所示，在 MAF 中，整个环境由多个经 ORB 这一通信总线连接起来的 Agent 系统(Agent System)组成，一个 Agent 系统内可以包含多个称作 Place 的 Agent 运行环境，多个隶属于同一属主的 Agent 系统组成一个 Region。此外，各个 Agent 系统还能够利用 CORBA 的一些公共对象服务来对 Agent 的安全、命名、活动控制和序列化等操作提供支持。Agent 系统间的互操作主要是通过对 MAFAgentSystem 和 MAFFinder 这两个接口的定义来实现的。MAFAgentSystem 接口必须由每一个 Agent 系统实现，它用来对 Agent 的管理作出规范，给出了常用的操作定义，如 Agent 的创建、终止、挂起、重激活、接收等，通过这些操作的标准化，能够方便地实现

*)本课题得到江苏省高技术产业化项目基金资助。贾志勇 博士研究生，主要研究领域为分布式计算、软件 Agent 技术。景广军 博士后，主要研究领域为人工智能、软件 Agent 技术。谢 立 教授，博士生导师，主要研究领域为分布计算与并行处理、先进操作系统等。

跨 Agent 系统的管理。MAFFinder 可以看作是一个目录服务，可以由多个 Agent 系统或多个 Region 共享，提供对 Agent、Place 和 Agent 系统的注册及动态查找和定位服务，借助 MAFFinder，Agent 或 Agent 系统能够方便地同其它远程实体建立联系。

MAF 虽然在一定程度上提供了对 Agent 互操作的支持，它也有如下一些较为明显的不足：(1)MAFAgentSystem 和 MAFFinder 是 Agent 系统级的接口，因此 MAF 所涉及的实际上仅限于 Agent 平台层次的互操作，未对其他层次的互操作(如 Agent 之间的互操作)给出明确说明；(2)MAF 是基于 CORBA 架构的，直接利用了 CORBA 的通信机制，因此较难对 Agent 之间所传递消息的语义进行限制；(3)MAF 所利用的 CORBA 公共对象服务无法充分满足 Agent 应用的需要，如 Security Service 只能对 Agent 系统提供较低程度的安全保障，缺乏 Object Trader Service 支持的 Naming Service 也无法提供针对 Agent 能力和服务类型进行查询的黄页服务。

2.2 FIPA 规范族

FIPA(Foundation for Intelligent Physical Agents)[7]是一个旨在促进智能 Agent 技术发展和应用的非盈利性国际组织，它定义了一组规范(即 FIPA 规范族)，用来支持 Agent 之间以及基于 Agent 的应用之间的互操作。

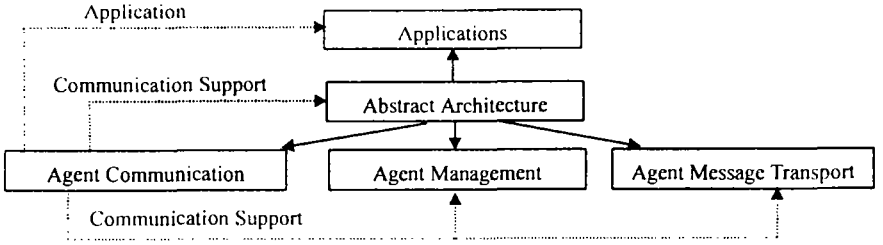

图2 FIPA 规范族

如图2所示，组成 FIPA 规范族的所有规范分成五个大类。抽象体系结构(Abstract Architecture)规范是理解整个规范族的出发点，它通过对不同 Agent 系统实现中的共性进行抽取，在抽象设计这一层次定义了一个体系结构，它包含三个必需元素：消息传递(Messaging)、目录服务(Directory Service)和 Agent 通信语言(Agent Communication Language)；Agent 管理(Agent Management)类规范在 Abstract Architecture 的基础上提出了一个用来作为 Agent 管理和运行平台的参考模型，这个模型由 Agent 管理系统(AMS)、目录设施(DF)和消息传输系统(MTS)三部分组成；Agent 通信(Agent Communication)类规范对 Agent 之间的消息格式、消息类别、消息内容表示以及会话级的 Agent 交互协议(Agent Interaction Protocols)做了说明；Agent 消息传输(Agent Message Transport)类规范是用来解决如何在各类网络应用协议上完成消息传输的，它给出了基于 HTTP、IIOP 和 WAP 等协议的消息传输模式和多种消息编码格式；Agent 应用(Applications)类规范则从 FIPA Agent 模型的角度对移动计算、网络管理、个人助理等领域的 Agent 应用进行了分析。此外，在整个规范族中，Agent 通信类规范是核心，所有其它规范都是围绕着 Agent 通信和 Agent 通信语言(FIPA ACL)建立起来的。因此，如图2中的虚线所示，可以把除应用外的其它三类协议都看成是通信支持(Communication Support)类规范。

与 MAF 相比，FIPA 对互操作性的覆盖面较宽，而且在一定程度上支持基于 Agent 通信语言的语义级互操作。但它也有着如下一些不足：(1)在信息传输方面，仅对 Agent 之间的消息传递提供了支持，没有对 Agent 自身的移动所需的传输服务支持做出说明；(2)将应用级的互操作仅限制为基于 Agent 的应用之间的互操作，并且只支持共享相同的语义信息库这样一种单一的应用互操作模式；(3)实际的多 Agent 系统应用中，Agent 之间经常会有相互依赖关系，FIPA 并没有提供与之相适应的动态分组、部署和配置支持；(4)尽管 FIPA 中多次提到了概念及关系库(Ontology)和安全(Security)机制的重要性，但却没有明确给出这两部分的相关实现规范。

3 Agent 互操作性研究框架

3.1 传统分布式应用下的互操作形式

在计算机系统中，应用之间的互操作性问题是由两方面原因造成的[8]：(1)不同应用所处的计算环境之间存在着多方面的差异；(2)应用之间有共享数据和使用对方所提供服务的需要。在传统的分布式环境下，应用之间能够有多种互操作形式，概括来讲，它们可分为两类：(1)应用互连(Application Connection)，它包括两方面内容，一方面是用来在分布式环境下寻找和定位所需的服务和信息，另一方面是处理服务的请求者和提供者之间的操作差异，如对它们之间接口、通信协议、同步、异常处置和资源管理等方面操作的规范；(2)信息兼容，用来处理应用之间的不同数据组织模式、不同的视图形式以及不同的数据表示形式之间所造成的信息共享障碍。

3.2 Agent 环境下的互操作研究框架

由于应用环境的相似性，在 Agent 环境下同样要首先对传统分布式应用下的互操作形式加以考虑。此外，由于 Agent 具有自主性、智能性等新特点，它对互操作提出了如下一些新要求：(1)支持携带明确语义信息的消息的传递和理解。带有语义信息的消息可以增进 Agent 间对彼此意图的了解，从而有助于 Agent 更好地完成任务，而且，它也能更好地体现 Agent 的智能性；(2)支持 Agent 在各个 Agent 平台之间的自主移动。移动是 Agent 的一个重要特性，是构建 Agent 计算模式的一个重要基础。对移动的支持通常包括实现 Agent 状态捕捉、代码及状态迁移和 Agent 运行环境重构等；(3)支持传统应用和 Agent 应用之间的交互与协作。Agent 应用不是孤立存在的，常常会与一些非 Agent 形式的应用发生相互作用。

为了能够清楚地说明 Agent 互操作问题，我们提出了一个如图3所示的 Agent 环境下的互操作研究框架，用来对 Agent 环境下的互操作内容作一个较全面的总结。在这个框架中，我们把 Agent 环境下的互操作内容分成了四个层次，由

低到高分别为基础设施层互操作、Agent 系统层互操作、Agent 层互操作和应用层互操作。

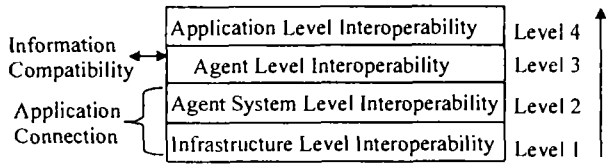

图3 Agent 环境下的互操作研究框架

基础设施层互操作的实现是其它各层实现的基础，它包括以下四个部分：(1)建立 Agent 之间以及 Agent 系统之间的传输通道，使得 Agent 之间的消息以及 Agent 自身都能够顺畅地在不同的 Agent 平台之间进行传输；(2)提供对 Agent 和 Agent 系统的命名、查找和定位服务，使得各个服务的提供者能够容易地被找到；(3)提供多 Agent 系统环境下基本的安全服务；(4)提供对 Agent 状态的捕捉机制和对 Agent 代码和状态的保存机制，来为 Agent 的迁移和意外情况下的系统恢复提供支持。

Agent 系统层互操作的实现与 MAF 中对 MAFAgentSystem 接口的定义功用类似，主要用来对 Agent 系统中一些基本操作的执行语义和执行步骤加以规范，从而能够方便地实现跨 Agent 平台的管理和操作。

Agent 层互操作的实现包括以下三方面呈逐次递进关系的内容：(1)Agent 间具有明确语义的单个消息的传递和理解；(2)会话语义的理解。在单个消息语义理解的基础上，可以通过构建一组具有特定关联的消息集合来实现会话级的语义理解，这样一组消息通常具有共同的语义背景和执行目标，能够表达单个消息难以表达的复杂语义；(3)在会话语义理解基础上的 Agent 动态分组、配置、合作和协商。通过构造适当的会话协议，能够使得遵循会话协议进行交互和动作的 Agent 之间自然地形成某种特定的依赖关系，并实现会话语义所蕴涵的合作和协商。与将分组和协作关系内含于 Agent 当中的实现手段相比，通过会话语义来实现 Agent 的分组和协作能够显著地提高系统的扩展性和简化 Agent 的开发工作。

应用层互操作包括两方面：(1)不同 Agent 应用之间(其中每个 Agent 应用可以包含多个 Agent 或 Agent 群组)的相互关联和互操作。解决的基本思想是把它们看作是需要进行简单合作的两个智能群体，并通过通信语义上的共享来实现这种合作。(2)Agent 应用与非 Agent 应用(如数据库系统、Web 浏览器及各种行业应用等)之间的相互作用。它应该既包括 Agent 应用对传统应用的调用也包括传统应用对 Agent 应用的调用，通常可以通过包裹(Wrapping)机制来对传统应用的调用接口进行包装来实现。

在以上四个层次中，如果暂不考虑 Agent 系统中特有的运行状态捕捉、消息和消息集合的语义理解等情况，前两个层次可以看作是用来实现应用互连形式的互操作，第三个层次用来实现信息相容形式的互操作。各层之间存在着相互依赖关系，较低层次互操作的实现是较高层次互操作实现的基础，由于具有语义信息的 Agent 通信是 Agent 智能性的一个重要体现，也是实现最高形式的互操作-应用级互操作的基础，同时它也是与传统的分布式互操作的最大区别，因此以 Agent 通信语言实现为基础的 Agent 层互操作是整个 Agent 互操作架构的核心。从这个模型可以看出，MAF 实际上只涉及到了一些应用互连级的互操作，即图中最低两个层次的一部分实现。FIPA 虽然对各个层次均有涉及，但如2.2节中所述，在每一个层次的实现都存在较多的不足，尤其是对一些关键的支持服务，如 Agent 的状态捕捉和传输、多 Agent 的分组和部署、多个概念及关系库的建立和相互协调等没有给出相应的规范。

4 基于互操作研究框架的 Agent 系统模型

为了能清楚地说明如何在实际的 Agent 系统构造中提供对互操作的支持，我们在前面 Agent 互操作研究框架的基础上给出了如图4所示的一个 Agent 系统参考模型。

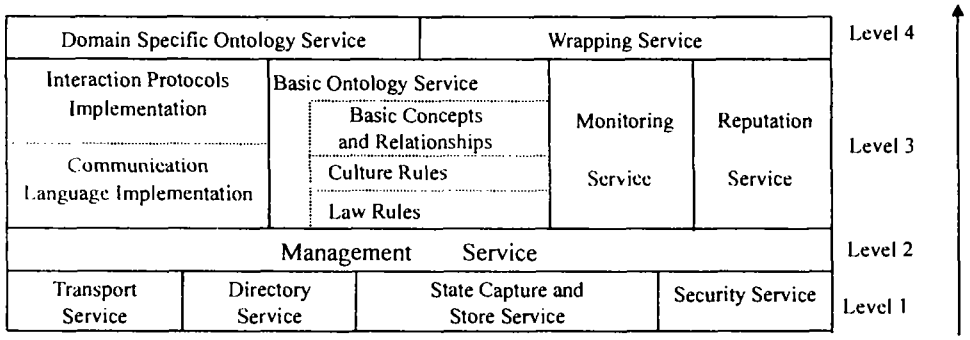

图4 Agent 系统参考模型

如图4所示，参考模型也分为由低到高的四个层次，各个层次分别与图3中的互操作研究框架各层次一一对应，每个层次都由一些支持服务构成，用来支持相应层次的互操作实现。

在最底层，传输服务(Transport Service)用来建立传输通道，实际实施时，由于不同的 Agent 平台可能会基于不同的传输机制实现(如 IIOP、JMS、HTTP、WAP、RMI 以及由不同开发者建立的专属协议等)，需要考虑提供不同传输机制之间的转换服务；目录服务(Directory Service)用来查找操作对象或服务的提供者，同样，如果存在着多种目录实现(如 LDAP、NIS、CosNaming、Jini LookupService、MAFFinder 以及各种由不同开发人员实现的特定目录服务)，也要考虑这些目录服务之间的转换；状态捕捉和存储服务(State Capture and Store Service)能够捕捉 Agent 的状态，并将 Agent 的代码和状态转换成方便保存和传输的形式，这样 Agent 就能够迁移到新的环境继续执行，在发生系统崩溃时也能够方便地恢复各个 Agent 对象；安全服务(Security Service)用来对各个计算节点、网络资源和 Agent 实施有效的保护，提供一个安全的计算环境。加密、授权和信任关系建立是实现安全机制的主要手段。

第二个层次的管理服务(Management Service)用来在各

个 Agent 平台间规范各种基本操作，如 Agent 的移动、接收、创建、中止等等，这些操作可以大致上分为系统操作和 Agent 操作两类，在实现上，它通常表现为一组面向基础设施层的接口 API。

在 Agent 层，具有语义特征的通信语言实现(Communication Language Implementation)是首先要提供的，从实现的角度来看，可以通过规范消息并对各个消息赋以相应的语义来实现对语义消息的支持，最常见的是基于言语行为理论(Speech Act Theory)的消息构造[9]，KQML 和 FIPA ACL 就是这方面的典型例子；此外，出于对会话语义支持的需要，要有交互协议实现(Interaction Protocols Implementation)，这就需要确定一组交互协议，每个协议都有一个特定的会话语义(如建议、查询、拍卖等)，并明确地规定了一组消息的类型和传递顺序；基本概念及关系库(Basic Ontology Service)是对前面通信语言实现和交互协议实现的辅助支持，它实际上是一个通用的概念、关系和规则集合，在这个集合中可能有以下三种元素：(1)基本概念和关系。用来定义一些通用的事物概念以及这些事物之间可能存在的关系；(2)文化规则[10]。定义一组非强制性的规则，作为 Agent 在执行任务时的参考；(3)法律规则。是一组强制性的规则，是相关 Agent 在执行任务时必须遵守的。与通常的概念库不同，我们在这里特别包含了文化规则和法律规则。它是基于这样一种认识，即作为自主执行的软件主体，Agent 的行为与其所属的应用领域、Agent 群组及执行环境密切相关并受其约束，这种约束有两种基本的表现形式：强制性的和非强制性的，前者表现为一种强制规则，能够即刻在 Agent 的行为中得到体现，而后者多表现为 Agent 在动作执行时的偏好，不一定立即在其行为中得到反映，更多的是表现为一种潜在的长期影响、一种趋势，从而使得具有相似文化背景(文化规则)的 Agent 群组的行为具有某些较为明显的公共特征；监督服务(Monitoring Service)和名誉服务(Reputation Service)一起，用来控制和监督各个 Agent 按照特定的交互协议、文化规则和法律规则执行动作和相互联系，从而保证各个协议和规则所蕴涵的各种分组、合作等语义得到体现。监督机制用来对 Agent 的执行进行审查和强制约束，名誉服务对一些 Agent 未能按照协议和规则执行的情况加以记录，从而能够方便管理者进行管理和调整，也为 Agent 在执行时选择合适的合作对象提供一定的依据。

在应用层，特定领域概念及关系库服务(Domain Specific Ontology Service)由专属于某一领域的概念和关系集合组成，使得不同应用领域的 Agent 应用能够对相互之间所传递的消息内容具有一致的理解。包装服务(Wrapping Service)提供多个针对不同应用系统的包装 Agent(Wrapping Agent)，这些 Agent 负责对各种非 Agent 应用进行包装，其它的 Agent 应用只需同这些包装 Agent 交互即可。同样，我们也可以提供适合传统应用调用的包装 Agent，它提供了接口用来供传统应用调用，并同时负责与其它 Agent 交互。

从这个模型的设计指导思想来说，我们希望达到如下三个目的：(1)提供一个具有良好可重用性和伸缩性的 Agent 平台架构。根据实际情况，可以对模型中各个层次的服务作适当的取舍。例如，如果运行环境是一个各个节点相互信任的小规模局域网，底层的安全服务就可以不必考虑。如果不存在非 Agent 应用，则不必考虑包装服务。在一些简单的情况下，文化规则也不是必需的。如果要添加对 Agent 的可视化的监控和管理，只要对管理服务中的一些有关 Agent 状态信息获取和行为控制的 API 进行调用即可。(2)提供对 Agent 各个层次互操作的全面支持。它是图3中 Agent 互操作研究框架的一个完整实现，全面地覆盖了 Agent 环境下互操作的各个方面。(3)支持尽可能简单的应用 Agent 开发。在模型中，我们尽可能地把各种支持服务和基本操作通过 Agent 平台来实现，这样开发者只需专注于具体应用部分的代码实现即可。例如 Agent 的移动，Agent 只需通过管理服务调用一条迁移指令，其它的支持，如状态捕捉、格式转换、传输、恢复执行等都有相应的平台服务来实现。

为了从实现的角度对这个参考模型加以验证，我们构造了一个基于 Java 技术的 Agent 系统，它将以上模型中的各部分集成到了基本服务模块、核心管理模块、执行引擎、通信翻译模块、概念关系库模块和包装 Agent 容器等几个子模块当中。为了更集中于较高层次的互操作实现，在系统的初期构造中对于安全服务只保留了它的一些基本接口而没有具体实现，同时也简化了目录服务实现，暂时没有考虑与其它类型 Agent 系统目录服务的互操作。系统的通信语言采用了 FIPA ACL，并且支持一组简单的交互和动态协作协议，支持旅行计划和即刻决策两种形式的 Agent 自主弱移动(即只有代码和数据状态的移动，而不牵扯到执行状态的搬迁)，系统包含一组针对数据库操作的包装 Agent。在此基础上，我们构造了一个模拟大宗农产品交易的 Agent 应用例子和几个信息采集和分析的例子，并取得了较好的效果，也很好地验证了图4中 Agent 系统参考模型的合理性和适应性。

结束语 本文深入讨论了 Agent 系统的互操作问题，在对现有研究工作充分分析的基础上，提出了一个 Agent 互操作研究框架，它涵盖了 Agent 互操作问题的各个方面。此外，还给出了与这个框架相对应的一个 Agent 系统参考模型，并从实证的角度构造了一个基于 Java 的 Agent 系统。

参 考 文 献

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

2 Object Management Group. Agent Technology Green Paper. 2000, 8. http://www.objs.com/agent/

3 Jennings N R, Sycara K, Wooldridge M. A roadmap of agent research and development. Autonomous Agents and Multiagent Systems, 1998, 1: 275～306

4 Kotz D, Gray R S. Mobile agents and the future of the internet. ACM Operating Systems Review, 1999, 8: 7～13

5 Luo YingWei, Wang Xiaolin, Cong Shengyi, et al. Research on the geo-agents. Journal of Computer Research and Development, 2000, 37(12): 1504～1512

6 Object Management Group. Mobile Agent Facility Specification. 2000, 1. http://www.omg.org/technology/documents/formal/mobileagentfacility.htm

7 The Foundation for Intelligent Physical Agents. FIPA Specifications. 2001, 8. http://www.fipa.org/specifications/index.html

8 Kent W. Object orientation and interoperability. Database Technology Department, Hewlett-Packard Laboratory: [Tech Rep: HPL-93-58]. 1993

9 Finin T, Weber J, Wiederhold G, et al. Draft specification of the KQML agent-communication language. 1993, 8. http://www.cs.umbc.edu/kqml/

10 Blanzieri E, Giorgini P, Massa P, et al. Implicit culture for multi-agent interaction support. In: Proc. of the Intl. Conf. on Cooperative Information Systems, 2001. 12～16