角色绑定的自组织自演化多 Agent 协同模型①

A Multi-Agent Coordinated Model of Self-Organizing and Self-Guided Evolution with Role Binding

金智勇[1] 叶时平[1] 高 济[1,2] 金林樵[1] (1.浙江树人大学 信息科技学院 浙江 杭州 310015;
2.浙江大学 计算机科学与技术科技学院 浙江 杭州 310027)

摘 要: 在深入分析角色绑定的 Agent 协同模型的基础上，利用 DCL(Domain Conception Language)语言描述应用域本体、角色定义和角色协作等模型要素，以自治计算元素(ACE)作为服务代理，动态绑定角色，驱动 Agent 理性遵从协同行为规范，实现系统的自组织、自演化。该模型突破了传统 Gaia 方法的封闭性及以确定需求确定为基础的限制。

关键词: MAS 角色 Agent 二元协作 DCL 语言

多 Agent 系统(MAS)是 Agent 研究的一个重要分支，也是分布式人工智能(DAI)研究中的一个前沿领域[1]。如何解决各成员 Agent 在多变的环境下有效地协作是研究与设计 MAS 面临的重要课题。基于角色的自组织、自演化多 Agent 协同系统有助于研究理解角色概念及与 Agent 关系，简化 MAS 设计。

目前，国内外研究者对角色在 MAS 中的地位和作用，基本上还停留在理论阐述，真正在应用领域进行研究并基于 BDI 模型实现应用的较少。本文以角色、组织等概念为基础的 MAS 模型对可能世界模型的描述采用 DCL(Domain Conception Language)语言描述本体、角色等内容，以 BDI 模型刻画和描述 Agent 的概念和特征，并作为单个 Agent 的知识表示和推理的基础，解决了理论与实践分离的问题，实现了基于角色的多 Agent 协同的知识供应系统。与现有基于角色的 MAS 模型系统相比较，具有一定的完整性，并针对角色、角色交互、Agent、Agent 角色扮演以及角色协同等做了深入的分析及实例设计。该模型构造的 MAS 系统具有良好的开放性，可在运行时刻动态地实现 Agent 与角色的转换[2]。本文从人与人类社会相互关系分析入手，根据二者与 Agent 及 MAS 的同构特点，分别对本体、目标、角色、交互、Agent 与角色的绑定等关键概念进行分析，提出系统目标由角色承担，Agent 间的协作交互通过绑定在其上的角色决定，这样，Agent 角色的动态变化得以实现。

1 Agent、角色及其关系

随着智能系统被广泛应用于大型的、开放的、复杂的领域，MAS 起着越来越重要的作用。MAS 的元概念模型以 Agent 为基础，认为系统是由多个相互合作的自主 Agent 构成。考虑到应用领域的不确定性，Agent 常常被赋予由应用域所决定的任务去评估系统中各角色的真实状态，并让它们能够根据判断去执行相应的任务[3]。

1.1 角色的概念及作用

到目前为止，“角色”在社会学中没有一个公认的定义。现有基于角色的、面向 Agent 的软件开发方法对角色的定义和认识也不尽相同。Gaia 方法[4]认为角色在抽象概念中使用，从宏观和微观两个层面分析、设计 Agent 系统。角色有四项属性：职责(Responsibilites)、许可(Permissions)、活动(Activities)和协议(Protocols)，通过确认服务模型来确认在一个或者

① 基金项目:国家高技术研究发展计划(863)(2007AA01Z187);浙江省科技计划基金项目(2007C21042)
收稿时间:2009-01-13

多个 Agent 中实现角色功能。角色的职责用于定义角色应当确保做什么、不做什么；许可用于保证角色职责的实现；活动用于扮演该角色的 Agent 在无需与其他 Agent 通信的情况下即可执行的行为；协议则定义了角色的交互方式。MESSAGE/UML 方法虽然提供了完整的某种实体，确保 Agent 采取相应的行为，但该方法对角色行为研究不够深入，仅将角色定义为由 Agent 扮演的某种实体，对角色语言描述也不够详细。本文将角色定义为：角色是对实体自身行为以及实体参与交互时采取的行为模式的规约，构成要素包括：属性、目标、约束、内部活动、服务和协议。

角色具有特定的属性以刻画角色当前的状态。不同角色具有不同的应用域。角色的目标和约束等要素都建立在属性的基础上，角色目标刻画该角色被赋予的任务。角色的目标具有层次性，即一个目标可以分解为多个子目标，一个目标能否实现依赖于其子目标实现与否。角色的目标可以表示为图，图中的节点代表目标，节点间的连线代表目标之间的依赖关系。

1.2 Agent 与角色的关系

Agent 体系结构中 Agent 定义为 a=(aid, Rs, m)，其集合记为 As。As 用于描述 Agent 的名字(aid)、所绑定的角色(Rs)和方法的集合(m)。静态角色组织 Ro=(Rs, Conns)由一系列 Rs 以及角色关系 Conns 组成。每个角色由 r=(rid, m)表示，用于刻画角色名字和方法。系统给每个角色赋予一定的载体即为 Agent，将角色分配到 Agent 的行为用函数 Assign 表示：

Assign：$Rs \rightarrow 2^{As}$

一个 Agent 绑定多个角色。Agent 与角色都具有一定的心智状态。Agent 能够通过与环境的接口(WIF)自治地感知环境，实现从环境到感知的映射[5]，可以描述为：

See: $E \rightarrow Per$

其中，E 为环境的集合，Per 为非空的感知的集合。本文对 Agent 与角色的关系描述如图 1 所示。对于获得的信息，Agent 首先在知识库中寻找适当的行为模式。若没有动作被激发，则进入自身的规划知识库中寻找，一旦找到就执行规划组件(PBC)。在基于角色的 MAS 中，角色实例化得到 Agent，并在软件系统中存在且发挥作用。

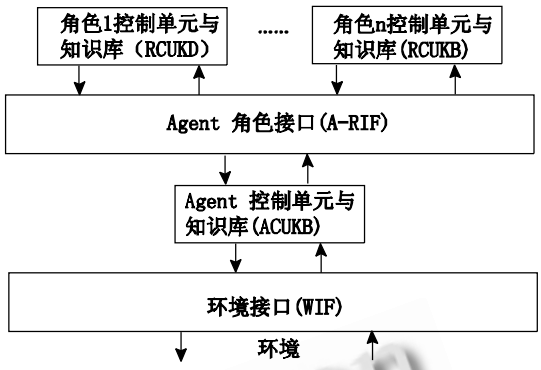

图 1　Agent 与角色的关系

通过实例化绑定角色，Agent 获得对自身行为的选择和激发机制，从而使 Agent 具有自主性和行为上的主动性。角色生活在特定的环境中，具有一定的目标，提供一定的服务，具有自身的内部行为，遵循特定的交互协议。正如人类社会中的个人具有自然属性和社会属性，Agent 也具有承担基本行为的物理实体和承担社会交互的角色实体。每个具有基本属性和行为的 Agent 绑定特定角色后构成具有自主性的软件组件。

2　角色定义及绑定

借助社会学和组织学等学科的理论，通过角色或组织等概念来理解系统行为，可将 Agent 视为系统中承担某个或某些角色的自主行为实体[6,7]。一个 Agent 可以承担一个或者多个角色，一个角色也可以为多个 Agent 所承担。角色限定了 Agent 的行为规则、交互方式，Agent 的行为能力通过其承担的角色访问。MAS 模型包括角色模型、Agent 模型，从角色模型得到 Agent 有两种选择：一是从角色得到相应的 Agent 类，并由这些类实例化得到 Agent；二是将角色视为在系统中存在的实体，不同的角色对应不同的 Agent 类，由这些类实例化得到承担不同角色的 Agent。

2.1 角色定义

基于角色的多 Agent 层次协作模型将不同角色的能力、知识、任务、要求等内容赋予相应的 Agent，使各 Agent 承担起角色所要求的任务。角色不仅定义了系统需要用户所做的工作，同时也规定了系统对用户所提供的服务。它包含两方面内容：职责和权限。参照 Gaia 方法定义系统相关的角色，本研究采用 DCL 语言来定义角色及相关概念。DCL 语言是本研究定义的一种概念表示语言，分为概念表示、概念分类体系、

概念属性的侧面定义三个部分。

定义 1. 每个业务操作型角色 bor 定义为 5 元组：

bor=(name,t-roles,rights,obligations,s-transition)

其中，bor(以 name 指示)可以承担不同服务的多个业务处置角色(t-roles)，拥有调用多种服务(业务或促进服务)的权利(rights)，并负有提供多种业务服务的职责(obligations)。状态变迁(s-transition)则指示服务协同参与状态的转变(bor 参与的协同从上一服务转变到下一服务)。作为连接 MAS 微观模型与宏观模型桥梁的角色定义如下：

BusinessPartnerRole <角色名>

Performs:{{InitiatingRole|RespondingRole}="<业务处置角色名>";}+

[Rights: {<角色协同行为规范>;}+]

[Obligations: {<角色协同行为规范>;}+]

[Transiton: {fromBusinessState < 2 方协同名>

toBusinessState < 2 方协同名>;}+]

End <角色名>

协同行为规范构成协议体系的核心，静态或动态说明参与协同的 Agent 及其承担的角色应尽的职责、禁忌和许可的权利，进而形成调控 Agent 可见行为的软约束。尽管无法确保 Agent 不会违反要遵守的规范，但可以通过设置奖惩条例以及违约检测和制裁机制，驱动 Agent 理性遵从行为规范。系统中角色协同行为规范的 BNF 定义如图 2 所示：

<角色协同行为规范>:=<逆向规则>

<逆向规则>:=(←<后果谓词公式><条件表达式>)

<后果谓公式>:=<关于服务请求的谓词公式>/<关于服务应答的谓词公式>

图 2 角色协同行为规范规则

系统定义的一个典型的权利型行为规范如下：

(← (@ServiceInitiating "KnowledgeItemBuyer" "KnowledgePurchase"));

指该角色作为"KnowledgeItemBuyer"，有权使对于服务"KnowledgePurchase"的请求(启动)成为事实，即有权请求(启动)该服务，为无条件行为规范。

另一个相应的典型的权利型行为规范如下：

(← (@TransactionResponding "KnowledgeItemSeller" "KnowledgePurchase")

(@TransactionInitiating"KnowledgeItemBuyer" "KnowledgePurchase"))

指该角色作为"KnowledgeItemSeller"，有义务让对于"KnowledgeItemBuyer"请求的应答成为事实，即有义务应答"KnowledgeItemBuyer"的请求。

业务伙伴角色划分为两类：业务操作型(Operational)角色和社交促进型(Facilitating)角色。前者定义应用域 E-机构，后者定义社交促进 E-机构。每个业务伙伴角色的描述包括角色名、承担的业务处置角色、权利 (许可)、义务(职责)、可能的协同参与状态转变。权利说明该角色有权请求(启动)的服务(以及服务下属的所有操作)；义务说明一旦被请求时有义务应答的服务(以及服务下属的所有操作)或在满足条件时有义务请求的服务[8]。权利和义务可以面向由社交促进服务 (定义于社交促进 E-机构)。协同参与状态由提供或消费的服务指示，状态转变使用 From <服务> to <服务>描述。

在知识供应例中，采用 DCL 语言对信息和知识访问者角色(Visitor)进行定义。Visitor 指系统中未注册为社区成员的角色，定义如下：

BusinessPartnerRole Visitor

Performs:InitiatingRole="CatalogBrowseRequestor";

InitiatingRole="sf:EInstitutionInformationRequestor";

Rights:(@eio: Norm NormCategory: "Role Norm" NormType:"Right" NormNo:1

Performer:"InitiatingRole"

Postcondition:(@eio:ServiceInitiating

BusinessTransactionRole:?x ServiceName:?y

TransactionName:?z RequestDate:?w));

End Visitor

2.2 角色绑定

角色生活在特定的环境中，每个角色都由特定的

Agent 扮演，Agent 通过绑定角色实例化。Agent 体系结构由 Agent 和角色所共同构成。角色刻画了系统中的交互，但角色本身的行为能力是有限的，它必须依附于某个载体，调用该载体提供的基本方法。该角色载体就是自治计算元素(ACE)，作为提供软件服务的代理。ACE 和角色绑定后具有数据、方法和状态，可以通过承担某种业务角色来参与服务协同。只要遵守相应的协同行为规范，就可以预测和控制它们的服务协同行为。Agent 为绑定在其上的角色提供服务。Agent 之间的通信受到绑定于其上角色的影响。

例如，知识访问者(只有目录浏览权利)可向社区管理 Agent(Authority)申请转换成为知识消费者，系统角色转换描述如下：

RoleBinding:Role×ACEs×{Authority}(Role-Agent

Role 应用域服务协同中 Agents 可以承担的业务操作(Operating)型角色集；

ACEs--自动计算单元的集合；

Authority--社区管理 Agent，提供业务操作型 Agents 的注册、信誉查询、角色转换、不良信誉的备案和制裁等促进服务；

Role-Agent:具有某种角色能力的 Agent。

如 Visitor 只有目录浏览权利，向社区管理 Agent 申请转换角色，成为“知识消费者”(KnowledgeConsumer) Agent，描述如下。

Visitor×ACEs×{Authority}(KnowledgeConsumer Agent

在多 Agent 团队中，每个 Agent 担负一个或多个角色，通过角色动态转换，这个团队能够提高执行任务效率，以适应不能预期的环境变化，提高系统的整体性能。角色绑定机制允许多个 Agent 在执行协作任务时相互协调。以角色为核心建立 Agent 模型，通过设定角色动态绑定机制，确定角色与 Agent 绑定的时机，使 Agent 模型的实例能够与角色动态地绑定运行，成为一个完整、可行的 Agent 协作模型。

服务作为系统协作的最小单元，是自然和直观的。角色具有一定的目标，提供一定的服务，具有自身的内部行为，遵循特定的交互协议。把计算组件、数据信息资源和商业产品等都抽象为服务(以及服务下的操作)。每个服务注册时指示一个服务类型，仅在具体调用时才建立实例。在当前 Agent 和熟人 Agent 都能提供某服务时，Agent 可以选择自己提供或派遣给熟人。只有基本服务才有其执行组件，复合服务没有对应的执行组件。每个业务服务定义为它所提供的业务操作集，业务操作 bo 定义为 5 元组：

bo = (name, prec, reqa, resas, result)

每个 bo(以 name 指示)的启动必须满足前提条件(prec)-基于 Ontology 的条件表达式。bo 启动时服务消费方的请求活动(reqa)包括指定操作的输入参数(1 个或多个 XML 文档)、文档安全需求、服务提供方确认接收到输入参数(请求消息)的截止期、操作安全需求。服务提供方执行业务操作后的应答活动(resas)可以有多种，每个 resa 指定操作的输出参数(1 个或多个 XML 文档)、文档安全需求、服务消费方确认接收到输出参数(返回消息)的截止期、操作安全需求。业务操作执行后期望达到的协同状态(result)用于检验操作是否成功。

3 系统相关定义

Agent 是构成 MAS 系统的惟一成分，MAS 系统中承担具体动作的实体是 Agent，角色决定了 Agent 基本动作的特征。每个 Agent 必须包含信念、能力、承诺、行为规则、意图等 5 个要素，通过 Agent 解释器控制 Agent 运行。根据多 Agent 协作要求，在系统中定义二元协作集、领域本体和业务处置集为系统角色服务。

3.1 应用域本体的描述

应用域本体用于对社交结构标准和协同行为规范定义时参照的应用域所作的概念化描述。系统采用 DCL 语言描述定义域本体，可以形式化表示共享本体论。概念定义、常量、变量、可嵌套列表等构成了对共享本体论的定义，而本体及其定义的概念实例联合起来则构成了信息模型。其中，概念的定义构成了共享本体论设计的核心。采用 DCL 定义概念的语法表示如下：

<Concept>:=Concept<概念名>

[Super:{超类名}+;]

{<Slot>:{<侧面名><侧面内容>,}+;}+

[Constraint:{<真值表达式>}+;]

End<概念名>

3.2 Agent 意图描述

在 MAS 系统中，必须使 Agent 的行为符合环境的特性，保持信念、愿望和意图的理性平衡。以 Agent 意图作为基于角色间协作技术 Agent 知识表示和推理的基础，理性 Agent 的行为受制于意图。理性 Agent 层给 Agent 建模，提供 BDI 心理模型本体、支持 Agent 信念(世界模型、应用域 E-机构和本地业务指令)、愿望(期望提供或获取的服务、面向复合服务提供的本地业务过程集 LBP)和意图(作为当前实现的愿望)的描述，提供策略描述本体作为 Agent 行为管理策略的描述规范(包括句法和语义)，提供 Agent 建模和运行平台支持 Agent 及其行为管理策略的建立、维护和演化，建立 Agent BDI 的意图模型 Intention。

定义 2. Agent 的意图说明复合活动如何规划执行，表示为其意欲提供的内、外部服务集。意图可定义为一个规划的集合：Intention⇒{ Service +}

Service ⇒ { NativeName , Description , Recipe }

Recipe⇒{Scheduling+}

定义 3. Scheduling 系统意欲执行的规划，定义为八元组 Scheduling=(Operator, Outputs, Fire-Pattern, SharingData, PreProcessing, Reques-tedServiceList, Planning,PostProcessing)

其中，Operator 为系统所需提供的服务操作；Outputs 代表每个操作有多种可以选择的输出；FirePattern 表示概念实例模式；SharingData 为系统共享的变量；PreProcessing 实现对调度规划需参考的信息作预处理；RequestedServiceList 列举需经协商才能调用的、外部提供的服务；Planning 表示系统的调度计划；PostProcessing 用于对活动(即调度规划)的执行结果作后续处理。

3.3 二元协作的定义

二元协作只涉及 2 个业务处置角色，分别作为协同的启动方和应答方。非二元协作的协作可转换为多个二元协作来完成。系统使用应用域的 DBPD(分布业务过程; Distributing Business Process)描述多方协同完成应用域业务的流程。

定义 4. DBPD 定义为由多个二元协作构成的半序集合，二元协作 bc 定义为 8 元组： bc = (name, t-roles, prec, result, obligations, deadline, c-activities, c-transaction)

其中：每个 bc (以 name 指示)都涉及 2 个业务处置角色(t-roles)，分别作为协同的启动方和应答方。bc 指定协同启动的前提条件(precondition)和结束后期望达到的协同状态(result)；规定双方在协同时应尽的职责(obligations)，包括提供业务服务和调用促进服务；指定协同启动后的完成期限(deadline)。非底层 bc 通过 2 个或多个下层 bc 来实现；底层 bc 描述了业务服务 bs，并通过 bs 提供的 1 个或多个业务操作 bo 来实现。下层 bc 或 bo 执行流程的编排形成对于协同活动(c-activities)的定义为(BNF 形式)：

<c-activities>:={(← {<activity>|<active-tySet>} [<condition>]) |

(or{(←{<activity>|<activitySet>}

[<condition>]) }+)}+ <activity>:=bc|bo

<activitySet>:=({sequence|concurrency}{<act ivity>}+)

非同父二元协作(服务)间的组合排序由业务操作型角色 bor 的协同参与状态转变(s-transition)来指示。业务服务需要确保收费和质量安全时，应设置签约操作(c-transaction)，使服务供、需契约能通过调用签约操作来建立。

知识供应例采用 DCL 语言对二元协作的定义如下：

BinaryCollaboration <2 方协同名>

TransactionRoles: InitiatingRole=<业务处置角色名>;

RespondingRole=<业务处置角色名>;

CollaborationActivities: <协同活动规则

[ContrctingTransaction: <业务处置名>]

[Obligations: {<角色协同行为规范>;}+]

[Constraints: <协同双方限制条件>]

End

业务处置名指示用于签约的业务处置，该处置的输入参数用于签约，也作为协作伙伴的适用性检查和协商时的约束参数。服务处置方式分为OnLineInstant、OnlineNonInstant、OffLine三种，在契约执行准则的设置中隐含体现3种服务处置方式的不同角色协同行为规范为协同双方的义务。

二元协作的描述包括协同涉及的2个业务处置角色、展开协同需满足的外部状态条件、协同活动编排、签约处置、期望的终结状态、协同双方的义务、协同完成的截止期。对于底层二元协作，其完成与否取决于编排的协作活动是否成功结束；而非底层二元协作编排的协作活动则不强加成功结束的语义。

4 结束语

基于角色的Agent模型的知识供应例，以DCL描述语言作为Agent应用域本体的描述语言，以BDI模型作为刻画和描述 Agent 的概念和特征，作为单个Agent的知识表示和推理能力的模式，实现了一种基于角色的自组织、自演化多Agent协同系统。Agent通过扮演角色，调用特定的服务实现角色要实现的目标，达到适应不同环境。角色所具有的特定行为激发规则由BDI心智模型来刻画并实现。在本系统中，角色是连接MAS微观模型与宏观模型的桥梁，基于角色的Agent服务协同模型以业务服务作为实现协同的基本单元。

通过对基于角色的多Agent协同系统进行分析研究，创建系统实现模型，有助于对角色概念及其与Agent相互关系的理解，简化Agent的设计，为进一步研究角色与Agent的关系及实现机制，促进Agent实用化等方面打下良好的基础。

参考文献

1 李滔,闫琪,齐治昌.基于多 Agent 系统的软件开发方法研究.计算机科学与工程, 2006,28(6):118－121.

2 Huhns M, Singh MP. Service-oriented computing: Key concepts and principles. IEEE Internet Comput 2005,9(1):75－81.

3 吕建,陶先平,马晓星,等.基于 Agent 的网构软件模型研究.中国科学(E 辑), 2006,36(10):1037－1080.

4 Zambonelli F, Jennings NR, WoodlDridge M. Developing Multiagent systems: The gaia methodology. CAM Trans. SoftW. Eng. Method, 2003,12(3):317－370.

5 Russell S, Subramanian D. Provably bounded-optimal agents. Journal of AI Research, 1995,(2):575－609.

6 Stal M. Using architectural patterns and blueprints for service-oriented architecture. IEEE SoftW, 2006,23 (2):54－61.

7 Wooldridge M,Van der Hoek W. On obligations and normative ability: Towards a logical analysis of the social contract. Journal of Applied Logic, 2005,(3):396－420.

8 Boella G, van der Torre L, Verhagen H. Introduction to normative multiagent system. Computational & Mathematical Organization Theory, 2006,12(2-3):71－79.