UML 与多 Agent 应用系统建模*)

孙华志
(天津师范大学计算机与信息工程学院　天津300074)

Multi-Agent Application System Model Based on UML

SUN Hua-Zhi
(College of Computer and Information Engineering, Tianjin Normal University, Tianjin, 300074)

Abstract In order to guarantee the quality and raising the reliability and maintainability of the system, we need to provide the support for designing the Agent-based software system. In view of the consistency of the Agent's concept with Object's, we analyze the thought of modeling on UML and then write this paper. This paper has made the helpful attempt to build Multi-Agent application system model based on UML, involving the descriptions such as static structure and dynamic action. It lists the major steps and method about system modeling based on expanding UML, also.

Keywords Multi-Agent, Unified Modeling Language (UML), Object-Oriented Programming (OOP), Agent-Oriented Programming (AOP)

1 引言

Agent 是一种处于不断变化、不确定的环境中，具有应激性、自治性和自我激发能力的实体，能感知环境并作用于环境。近来对多 Agent 系统的理论研究和实际应用方面已经取得了长足进展，Agent 作为一种抽象工具，为理解、设计和实现一类复杂的软件系统提供了较好的解决途径。但是，目前的多 Agent 系统开发都是基于经验的活动，没有成熟的工具为开发过程提供支持，Agent 技术在大型系统中的成功应用实例尚不多见。因此，实现基于 Agent 的软件系统仍然是目前研究的热点和难点。为了使多 Agent 系统用于大型软件开发，必须有一套完善的设计方法和建模技术[1]。

从目前的研究成果来看，多 Agent 系统的构造方法，主要借用了两方面的研究成果，一个是面向对象的方法；另一个是知识工程的方法。其中，面向对象的方法得到了更广泛的应用，这是因为一方面，面向对象技术已较成熟，有很多成熟的方法和工具；另一方面，对象和 Agent 是两种有很多相近特征的工具，从对象过渡到 Agent 直观且自然。本文所探讨的是在面向对象方法学的基础上，研究如何将现有的 OO 建模技术扩展到 Agent 系统，为分析和设计 Agent 系统探索一条有效的途径，为 Agent 系统的开发提出一套可行的方法。

UML (Unified Modeling Language)是一种以面向对象思想为基础的通用可视化建模语言，1997年11月被对象管理组织(OMG)采用，是集成和扩充了 Booch、OMT、OOSE 等面向对象建模方法的标记语言，具有表达严谨、扩展性好的特点，现已成为业界标准[2]。UML 作为软件工程领域面向对象的标准建模语言，如何运用其设计分布式应用系统仍缺乏相应的过程和方法。为有效地使用这种语言对分布式应用系统进行分析和设计，还需辅以必要的建模过程。

本文将以扩展面向对象的可视化建模工具——UML 为核心，尝试应用于多 Agent 系统分析和设计的建模方法，并通过明确各种角色作用、形成时机和发掘角色之间的依赖关系来较好地体现面向 Agent 技术的核心思想。

2 UML 中常用的建模机制

UML 是一种可用于对大型系统建模的统一建模语言，它不仅支持面向对象的分析和设计，还支持软件开发过程。利用不同的模型来描述一个应用系统在不同生命周期中的各种静态结构方面和动态行为方面的特征，为任何具有静态结构和动态行为的系统进行建模；同时提供模型扩展和管理等方面的支持。从系统分析和设计的角度出发，这些模型主要可以分为三类：用例图模型、静态结构模型和动态行为模型。

2.1 用例图模型

用例(UseCase)图是从系统外部用户的角度对系统功能进行的描述，刻画系统的用户以及外部系统与本系统的交互，用例图可直观地反映用户对系统功能的要求，便于系统分析人员与用户之间的交互，有效地提取用户需求。同时，用例图也有助于从一个抽象的层次上自然地划分系统的功能模块。

2.2 静态结构的模型

静态结构模型主要用于定义系统中各种对象和实现以及它们之间的关系，是定义系统动态行为的基础。静态建模机制包括类图、对象图、包以及实现图(包括组件图和配置图)。

类图用来表示系统中所包含的类以及各个类之间的关系，定义类的内部结构(类的属性和操作)，表示类之间的联系。类图是对系统静态结构的描述，描述的静态关系在系统整个生命周期都是有效的。

对象图是类图的实例，显示类的多个对象实例。对象图及时具体地反映了系统执行到某处时系统的工作状况。由于对象存在生命周期，因此对象图只能在系统某一时间段存在。

包是由包或类组成，表示包与包之间的关系。包图用于描述系统的分层结构，是将大系统拆分为小系统的工具，可以用于系统的理解与维护。

*)资助项目：天津市高等学校科技发展基金项目(01-20405)，天津市教育科学"十五"规划重点课题(KHS021)。孙华志　副教授，主要研究领域为人工智能，远程教育技术。

一个组件是指一些彼此之间联系较为紧密的元素构成的集合。组件中可以包含子系统、面向对象的应用程序、一个可执行的二进制文件，或者一个类。组件图描述的是组件之间的联系，以及各组件之间的交互和它们所呈现的界面。另外，在分析阶段，也可以使用组件图对联系较为紧密的类进行更大粒度的封装，并实现对用例图的更进一步功能划分。

配置图通过描述在运行过程中各处理单元的配置来反映系统运行时的分布方式。配置包括硬件以及运行在硬件上的软件(组件、进程、对象等)。

2.3 动态行为的模型

系统的静态结构描述说明了系统包含的内容以及它们的关系，但它并不解释各个对象是如何协作来实现系统功能的。动态行为模型主要用于定义对象在时间上的历史，以及对象之间为达到一定目标所进行的通信，即描述系统中的对象在执行期间不同时间点是如何动态交互的。这类模型包括行为图和交互图。

其中，行为图描述系统动态模型和组成对象间的交互关系，包括状态图和活动图；交互图描述对象间的交互关系，包括序列图和协作图。

状态图描述类的对象所有可能的状态以及事件发生时状态转移条件。一个对象一般都具有行为和状态。状态图反映的是一个对象在其生命周期中将接收到的激励，以及由此所经历的一系列状态。

活动图反映一个连续的活动流，描述满足用例要求所要进行的活动以及活动间的约束关系。活动图由各种动作状态构成，每个动作状态包含可执行动作的规范说明，当某个动作执行完毕，状态就会随着改变，控制就从一个状态流向另一个与之相连的状态。

序列图用于显示对象之间的动态合作关系，强调对象之间消息发送的顺序，同时显示对象之间的交互。一个序列图对应一个用例图。它从时间顺序角度描述一个用例所涉及到的对象之间消息传递，从而以可视的方式表达对象之间的交互过程，以及系统执行过程中在某一位置将会发生什么事件。通过序列图可以刻画出一个用例的逻辑流程，以及在这个逻辑流程中会涉及到的对象。

协作图与序列图相似，显示对象间的动态合作关系。除显示信息交换外，协作图还显示对象以及它们之间的关系。一般说来，序列图更直观地表述出消息的传递时序，而协作图则主要反映消息传递时的一种层次关系。如果强调时间和顺序，则使用序列图；如果强调上下级关系，则选择协作图。

UML 中的这三类模型反映了客观事物及对象的一些共性规律。Agent 的概念也可以看作是对象概念的自然延伸，适当地扩展这三类模型，可以使之用来进行多 Agent 系统的分析和建模。

3 多 Agent 系统的建模策略

对象和 Agent 都是一种对客观事物的抽象，它们之间共同点很多，面向对象技术也有向 Agent 技术靠拢的趋势。一般来说，对象可以看作是一种封装了属性、事件和方法的计算实体。而 Agent 则是一种更高粒度的抽象，它除了对属性、事件和方法的封装外，还封装了相关的思维能力和决策行为，从而体现了较高的自治性；较强的面向目标性；灵活的反应性以及和其它 Agent(或对象、人等)进行交互的社会性等等。因此，Agent 是一种具有主动行为能力的智能对象。从这个意义上说，人也是一种 Agent。而多 Agent 系统也就可以很自然地从人类社会中"组织"的概念对应过来。所谓组织，可以更一般地抽象为具有交互能力的多个自治的智能体在一定目标的导引下，按一定关系结成的集体。所以多 Agent 系统的建模过程也就是 Agent 组织的描述过程[3]。

从内部构成上看，一个组织可以通过构成组织的角色、角色间的静态关系和动态关系三部分来刻画。因此，从 Agent 组织分析和设计的角度，采用扩展 UML 的可视化建模工具来进行多 Agent 系统的建模是可行的。

在完成一个实际开发过程中，在分析阶段，应着重提取用例以及与应用领域相关的类和组件，应尽量避免引入应用领域以外的内容。到了设计阶段，则要根据运行平台、实现技术以及分布的需要对模型进行进一步扩充。

(1)在开发一个应用系统时，获取应用领域或用户对系统的功能需求，从而明确系统的功能。用例驱动的方法强调通过用例提取和反映用户对系统的需求，以 UseCase 图作为与用户交互的依据及形成其它所有文档的起点。

(2)在建模的开始阶段，主要考虑与应用领域相关的类，用来描述实际应用领域中复杂的逻辑抽象，以使设计具有较好的可重用性。这些类的设计与领域相关而与实现无关。随着设计的细化和深入，当设计逐渐接近实现时，必将要涉及到应用领域以外的因素，诸如具体的实现技术、运行平台以及对分布的需要。应该根据应用系统所涉及的操作系统、开发工具、数据存储以及硬件的分布来补充应用领域以外的类。

(3)为满足实际应用中对系统的要求，需要进一步扩充和细化类图，并充分利用实现图(包括组件图和配置图)来满足系统对分布的需要，其关键的步骤就是使用组件图对各个类进行更大粒度的封装和分布。实现功能的自然划分、功能模块的高内聚低耦合是软件工程的重要原则，同样也是在设计组件图和配置图所应遵循的原则[4]。

对组件的提取或者封装，首先应提取出领域组件(Domain Component)，即与应用领域有关的组件。组件之间的联系主要应考虑组件之间消息交互、网络传输、响应时间等。因为组件可能分布到不同的硬件上，所以各个组件之间的交互则意味着潜在的网络传输，为减少组件之间的交互，应将交互复杂的类尽量自然地封装到同一个组件中。当完成上述工作后，则需要通过配置图将各个组件映射到实际的硬件配置上。设计原则是：满足实际应用中对分布的要求；尽量减少网络的传输量，必要时应根据需要建立专门的高速连接。

4 多 Agent 系统的建模实现

本文根据开发分布式远程智能教学平台的应用实践，从以下三方面，给出基于扩展 UML 的多 Agent 系统建模的主要步骤和方法。

·采用基于用例图的思想来提炼 Agent 组织的相关角色；

·采用基于扩展类图和对象图的角色关系图来分析 Agent 组织的静态组织结构；

·采用扩展的顺序图和状态图来描述角色间的交互行为和角色内部的推理行为。

4.1 Agent 组织的角色定义

角色是组织中一定责任、功能和行为的结合体，它对组织中一个或多个特定的目标负责。一个 Agent 可以对应多个角色，一个角色也可以由多个 Agent 来扮演。在组织建模过程中，角色是基本单位。这是因为和每个具体的 Agent 个体相比，角色抽象出了一类 Agent 的共性，而忽略了其具体细节，从而可以更稳定、更有效地描述组织特性。另一方面，以角色为基本建模单位反映了系统建模过程中目标驱动的思想，而

采用用例图进行系统功能描述的重点也是对系统目标的满足，因此两者是一致的。

这样，结合用例图的思想，Agent 组织的角色定义过程实际上就是组织目标的不断细化和明确的过程。这个过程可以简单地总结如下：

(1)组织目标的确定：确定系统总的目标和功能；

(2)组织目标的细化：列出为达到组织总目标需要完成的各个子目标；

(3)目标间的结构化：分析各子目标间的关系，对相关子目标合并、细化，得到目标间的层次关系图；

(4)确定组织角色：以各级子目标为基础，根据建模需要选定角色粒度，得到组织角色列表。

在得到组织的目标层次图的基础上，可以根据分析的需要，确定角色定义的粒度。如在一个分布式教学系统中，为了反映整个系统的概况，我们根据组织目标来确定二级组织的角色，这样就可以得到教师、学生、管理、监控等四个粗粒度的角色，分别扮演的是教学资源提供方、需求方、管理方、协调方。

4.2 Agent 组织的静态结构建模

Agent 组织的静态结构建模主要用于描述组织中由角色关系反映出来的组织结构。这种结构具有相对静止和稳定的特点。比较典型的有控制关系的等级结构、平等关系的扁平结构等等。组织结构的核心是角色间的关系，这种关系影响了角色在彼此的交互过程中可能采用的行为模式。

在多 Agent 系统中，扮演一定角色的 Agent 必须承诺根据角色的要求与其它 Agent 进行交互，这实际上避免了 Agent 由于高度自治可能产生的与组织目的不一致的内耗行为，从而有利于保证组织整体行为的协调一致。

UML 中的类(对象)图是一种通过描述构成系统的类(对象)及它们之间的关系来描述系统静态特性的建模工具。通过将类的概念扩展到 Agent 组织中的角色概念，尤其是将类之间的关系扩展为角色之间的关系，我们可以得到描述组织静态结构的角色关系图。

在 UML 的类图关系中，类与类之间的关系通常有四种：关联、通用化(继承)、依赖和精化。而 Agent 组织中的角色关系可以根据组织建模的需要进行扩展。比如，为了反映角色间的权力关系，可以引入控制关系、平等关系。另外还可以在系统引入角色之间的依赖关系、友善关系、拥有关系等。

4.3 Agent 组织的动态行为建模

Agent 组织的动态行为建模主要用于描述组织中角色间的动态行为，角色间的动态行为可以从两方面来刻画：一方面着重于角色之间交互行为描述，可以称之为角色间的交互行为建模，这是一种从角色的外部表现上对角色之间动态关系的刻画；另一方面着重于每个角色内部推理行为描述，可以称之为角色个体的推理行为建模，这是从角色的内部结构上对角色内部思维状态的刻画。这两个方面的建模在本质上是统一的，它们反映了观察组织中角色行为的不同视角。

(1)*角色间的交互行为建模* 交互协作是多 Agent 系统的一个重要特性，组织中角色的交互行为是 Agent 交互行为的集中反映。通过建立一些相对稳定的交互行为模型，可以对交互行为进行合理的抽象，形成相应的交互协议和交互模式，这有利于简化交互行为的描述和推导，也便于软件重用[5]。

Agent 组织中各个角色之间的动态关系是建立在相互之间进行消息通信基础上的，交互行为建模的重点在于对消息序列的刻画，即描述消息是如何在交互对象之间发送和接收的。

由于 Agent 与对象这两个概念之间的差异，因此在将 UML 中的顺序图应用于 Agent 之间的交互行为建模时，也应该对其进行适当的扩展。一个最基本的扩展是将图中对象的概念扩展到 Agent、角色或是某个 Agent 群体；另一种扩展是将消息的格式进行扩展；第三方面比较重要的扩展是对并发机制的支持。

可给 Agent 设计一些专门的接口，使 Agent 间能建立一个动态的、松散的协作关系。Agent 通过接口与外部、与其它 Agent 进行联系。Agent 间的联系可以是同步的，也可以是异步的。每个 Agent 使用相同的 KQML 消息原语，使用相同名称的接口进行处理。而且，在系统分析设计时，要把 Agent 的知识和能力，以本体论的形式进行描述，在实现时可以用数据库的形式表示。

(2)*角色个体的推理行为建模* 角色个体的推理行为模型主要着眼于角色个体的内部行为机制，这是一种范围相对较窄但更精细的描述，也更直接地对应于具体的角色个体和 Agent 个体的编程实现。

Agent 角色的推理行为模型主要借助 UML 中的状态图来表达。它是一种从有限状态自动机的思想中引申出来的建模工具。适合于表达一个对象或系统在其整个生命周期中的状态变化，可以比较容易地用来对 Agent 角色建模。

具体来说，角色的状态模型由角色在生命周期中所处的各个状态和状态之间的变迁组成。其中，状态是对角色在生命周期中一段时间的描述，在这段时间中或者是角色的属性值相对稳定，或者是正等待某些事件的发生，或者是正在执行某些内部操作。

在进行具体角色推理行为建模的过程中，角色在每个状态下的内部行为可以进行更详细的定义。同时，通过为多 Agent 系统中的每种角色建立相应的推理行为模型，结合它们之间的交互协议模型，可以有效地检验交互过程的协调性，防止死锁、冲突的产生。

结束语 基于“对象”概念和 Agent 概念的一致性，本文提出了采用扩展的可视化建模工具 UML 进行多 Agent 系统建模的思路。另外，由于 Agent 概念中蕴涵有一些不同于对象的特性，因此，在扩展 UML 时，如何处理好表达的准确性和规范性的问题是应该注意的重点。Agent 技术作为新一代的计算技术与面向对象技术的发展趋势是一致的，因此，利用 UML 进行多 Agent 系统建模是一个重要的方向，面向多 Agent 系统的体系理论和相关软件技术的发展，将会对计算机应用领域产生深刻的影响。

UML 虽然是一种可视化建模语言，但是它与大多数面向对象语言存在紧密的映射关系。在 UML 语言的代码生成机制方面，可以直接利用 UML 类图生成计算机框架程序。

参 考 文 献

1 Yim H, Cho K, Jongwoo K, et al. Architecture-Centric Object-Oriented Design Method for Multi-Agent Systems [EB/OL]. http://www.auml.org/auml/working/yim-w001.pdf, 2000-02/2002-03

2 Fowler M, Scott K. UML Distilled, Applying the Standard Object Modeling Language [M]. Massachusetts: Addison-Wesley Pub. 1997

3 冯珊，唐超，闵君. 创建智能体系统的软件工程方法研究. 系统工程与电子技术，2002，24(12)：96～99

4 郭亮，黄席樾. 面向 Agent 的软件工程. 重庆大学学报，2002，25(10)：132～135

5 Odellj, Parunakhvd, Bauerb. Representing Agent Interaction Protocols in UML [EB/OL]. http://www.jamesodell.com/Rep-Agent-Protocols.pdf, 1999-04/2002-03