多 Agent 系统仿真平台①

李慧琴[1]，薛 霄[1,2]

[1](河南理工大学 计算机科学与技术学院，焦作 454000)
[2](清华大学 国家 CIMS 工程中心，北京 100084)

摘 要：Agent 和多 Agent（MAS）系统目前已经逐步走向成熟化，且已经被广泛应用于系统建模、敏捷制造、网络监测等众多领域。为了能够便于 Agent 系统的开发，人们构建了各种 Agent 基础设施平台，例如已经广泛应用的 JADE 等。但是基于这些平台的应用大多具有平台依赖性，难于扩展和定制，无法支撑我们大规模的扩展使用。因此，将通过多 Agent 系统仿真平台 Maze 的设计与实现，着力构建出 Agent 系统各种基础设施的通用开发模式，包括 Agent 自治机制、通信机制、协作机制、协调机制等，从而便于用户在此基础上进行定制开发。另外，该平台的形象演示可以帮助初学者理解 Agent 的各种特性（如自治性、社会性、预动性等），比较各种协作算法的优劣等，是一个很好的入门学习工具。

关键词：Agent；通信机制；自治性；协调与协作

Multi-Agent System Simulation Platform

LI Hui-Qin[1], XUE Xiao[1,2]

[1](College of Computer Science and Engineering, Henan Polytechnic University, Jiazuo 454000, China)
[2](National CIMS Engineering Center, Tsinghua University, Beijing 100084, China)

Abstract: Agent and Multi-Agent (MAS) system has gradually become mature, and has been widely used in system modeling, agile manufacturing, network monitoring and many other fields. In order to facilitate the development of MAS, a variety of Agent oriented infrastructure platform have been built, e.g. JADE which have been used widely. However, it is difficult to extend and customize the customer-centric application based on these platforms, which have been an obstacle for us to apply agent technology in practice. Therefore, this paper will summarize some common patterns (including autonomous mechanism, communication mechanism, collaboration mechanism, coordination mechanism)to support the development of the MAS system through, the design and implementation of Maze, which is a multi-Agent system simulation platform. Based on these patterns, users can develop the customized application based on agent technology conveniently. In addition, the vivid presentation of the platform can help beginners understand, various characteristics of Agent (such as autonomy, social, and pre-mobility, etc.) compare the performance of different coordination algorithms. Therefore, it is also a good entry-learning tool for studying the agent technology.

Key words: Agent; communication mechanism; autonomy; coordination and collaboration

1 引言

自 20 世纪 70 年代末 Agent 理论和技术被作为人工智能中一个研究领域以来，Agent 技术的发展变化很快，并引起了国内外众多研究人员的普遍关注，究其原因是 MAS（Multi-Agent Systems）在计算机科学及其领域扮演了重要的角色[1]。随着现代计算机的发展，计算机并不再是独立运行的系统，已经转变为大型分布式的系统，计算机之间、计算机和用户之间的密切联系使计算机和信息处理系统也变得日益复杂。传统的集中式模型不能适应现在大型分布式信息处理

① 基金项目:河南理工大学研究生学位论文创新基金(CX2010-31);矿山空间信息技术国家测绘局重点实验室开放基金(KLM201110)
收稿时间:2011-08-21;收到修改稿时间:2011-09-16

的要求，但基于 Agent 的计算和以 Agent 为主体的高层交互可以满足现代计算和分布式信息处理系统的要求。目前国内外已经开发了一些 Agent 仿真系统，比如由 Parma 大学所开发的 JADE(Java Agent Development framework)和英国电信开发的 ZEUS 等[2-4]。其中，JADE 是最流行的一种分布式的 Agent 平台[2,5]。虽然 JADE 得到了广泛的应用，但是基于这个平台的应用大都受制于这个平台，而且在我们大规模扩展使用时，难以对其扩展和定制。ZEUS 的模块化比较强，但是只有其中的一种 Agent 模块得到了支持，因此限制了对多 Agent 系统多样化的设计，而且，ZEUS 像其他复杂的工具一样难以把握。

通过对相关工作的分析，面向 Agent 的系统开发方法目前多停留在抽象层次，对于实际开发缺乏细节支持，我们所开发的 Maze 仿真系统是根据层次开发框架 HDA 中的设计模式来设计和实现的[3]，因此，Maze 系统的仿真和实现是对 AO（Agent Oriented）软件开发方法[6]难以在实际软件开发中运用的一个挑战。

2 Maze系统

2.1 系统概述

Maze 系统以官兵捉强盗为背景，来充分演示 Agent 和多 Agent 的各种特性，比如：Agent 的自治性、能动性、通信能力、协调与协作能力等。在 Maze 系统中，每个 Agent 对应一个官兵或者强盗，所有的 Agent 就组成了一个多 Agent 系统，它们遵循以下的运行规则：

(1) 每个 Agent 都像游戏中的各个角色一样都具有自己的体力临界值、速度临界值、信心临界值、位置、个人成就感等，并且这些值之间是相互关联的，比如说体力值越高，Agent 的速度则越快，而且这些值的大小还决定了 Agent 的自治性和它们之间的协作程度（呈反比关系）；

(2) 每个 Agent 都属于两种不同的组织，就如游戏中的官兵和强盗，在一定的范围内，它们可以相互发现对方，并且攻击对方，当它们的体力值降低到零的时候，这个 Agent 的生命就结束了；

(3) 每个 Agent 都有一定的自治性，它们可以根据自身的情况，向队友提出请求，也可以对队友提出的要求进行援助或者拒绝；

(4) Agent 的任务是在迷宫中搜索敌人并消灭敌人，当发现敌人的时候，Agent 根据自身的情况可以攻击敌人，也可以向队友求助或者逃跑；

(5) 在迷宫中有安全区域（图中的灰色地带），Agent 可以逃到安全区域进行休息，恢复自己的体力和生命值；

(6) Agent 之间可以互发消息，它们可以进行协作并共同消灭同一个敌人。

Maze 系统的主界面如图 1 所示，分为四个部分：①最上方的部分是工具栏，是用户绘制迷宫、控制 Agent 运行时所要用到的工具；②左侧是参数设置界面，用户可以通过对话框设置 Agent 的组织结构形式、Agent 的属性值、以及 Agent 进行交互时的消息格式等；③右方的大视图是程序运行的演示界面，用户绘制好迷宫，并把 Agent 放置好以后，用户可以观察到 Agent 的运行过程和运行结果；④右下侧为 Agent 在迷宫中进行对抗时，它们交互时的消息显示界面，用户可以观察到它们在对抗过程中，每个组织的 Agent 交互的次数、交互的双方、以及交互的内容。用户绘制好 Agent 的运行环境后，并确定了 Agent 的初始运行条件后，就可以通过工具栏上的“开始”、“暂停”和“停止”按钮来控制 Agent 的运行了。

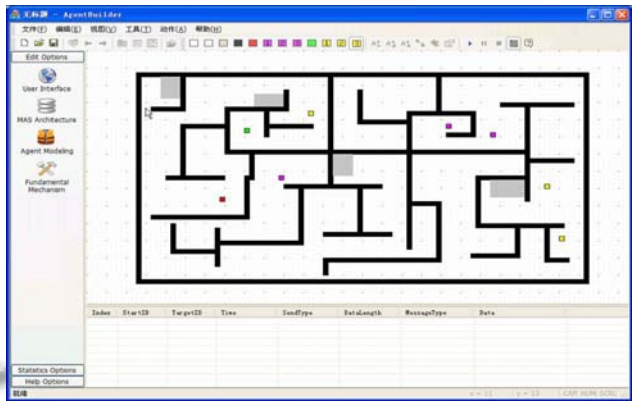


图 1 Maze 程序的主界面示意图

2.2 Agent 的设计

在官兵捉强盗这个场景中，所有的 Agent 被分为两个相互对抗的组织：Police(官兵)和 Enemy(强盗)。其中 Police(官兵)根据其职权不同又分为：Commander (指挥官)和 Soldier(士兵)；与此类似，Enemy(强盗)也分为：Robber(强盗头目)和 Subrob(小喽啰)。每个 Agent 被赋予了场景中的一个角色，根据所扮演的角色来履行它自己的责任和义务。在官兵和强盗相互追逐、相互对抗的过程中，在同一个组织内部会出现队友之间的协调和协作，但是每个 Agent 又都有自己的意愿，面对队友提出的请求，可以拒绝也可以接受。

因此，在设计和实现 Agent 的时候，一方面需要保证 Agent 的自主(治)性、反应性、预动性等个体特征；另一方面还需要兼顾 Agent 的社会性，要保证 Agent 之间可以交互，协作和协商。根据 HAD 中的 Agent 架构设计模式[4]，我们给出了 Maze 中的 Agent 类图，如图 2 所示。

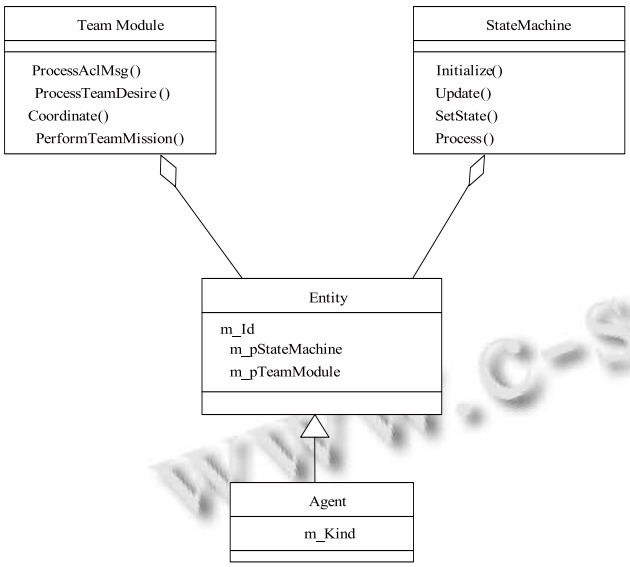


图 2　Maze 中的 Agent 模型图

3　Agent的通信机制

Agent 的通信机制是 Agent 进行交互和协作的基础，也是 MAS 得以形成和正常运营的关键所在。在 Maze 中，Agent 之间的通信是通过个体之间的消息传递来实现的。Agent 在通信时采用的是端到端的通信模式，在这种模式中，MAS 中消息的接收者和发送者是平等的，并允许一个自主 Agent 根据自己的意愿忽视收到的消息[7]，所传递的消息都被放入到一个缓存中，直至接收者决定读取它们。

在 Maze 系统中，通过两个队列来管理消息和所要接收到消息的 Agent：消息队列和熟人对列。其中，消息队列是用来存储消息的，每个消息接收者和发送者都有这样一个消息队列；熟人队列是由 Agent 的队友所组成，用来管理能接收所发消息的 Agent。熟人队列的创建需要借助于目录服务，目录服务包括服务注册、服务查找、Agent 注册、Agent 查找等等。而对 Agent 的注册又包括注册 Agent 的数目和这个 Agent 所能提供的服务和所能接收的服务，因此在知道了每个 Agent 所能提供的服务的情况下，就可以对能够提供这种服务的 Agent 建立一个熟人队列。Agent 之间具体通信过程如下：

（1）发送方 sender 将自己的需求意愿编制成二进制消息；

（2）把消息发送到消息邮箱；

（3）通过目录服务查找能接收该消息的 Agent 的 ID；

（4）消息邮箱把消息传递到能接收该消息的 Agent 的消息队列中去；

（5）接收者 receiver 把消息解码成思想状态，从而了解到 sender 的意愿。

在通信过程中涉及到一个重要的部分——消息邮箱，在消息传递的过程当中，无论是 sender 发送的消息还是 receiver 发送的消息都要先发送到消息邮箱，然后通过消息邮箱的功能找到 receiver，把消息发送出去。消息邮箱的功能如下：

（1）从消息发送方那里接收消息：对消息进行记录并保存；

（2）把发到消息邮箱中的消息发到能够接收该消息的 Agent 的消息队列中去，如图 3，具体步骤如下：

① 查看消息邮箱是否有消息；

② 从 Agent 目录中获取 Agent 指针；

③ 找到接收者；

④ 把消息加到接收者的消息队列中去。

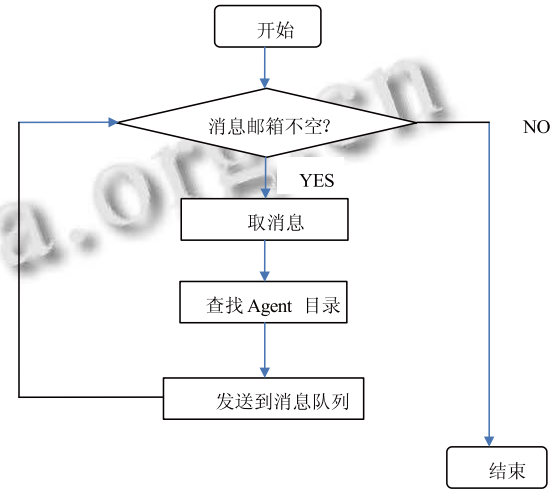
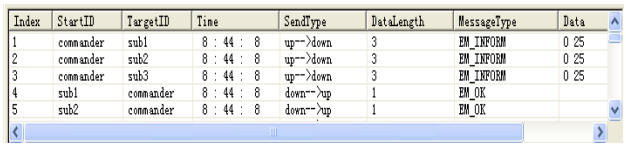


图 3　消息邮箱工作流程图

通过系统演示界面中的消息视图，用户可以清楚地看到消息的序列号、消息的发送者、消息的接收者、发送消息时的时间（精确到时、分、秒）、消息的发送类型（也就是上级向下级发还是下级向上级发亦或是同等级别的发送）、数据内容的实际长度、消息类型、数据内容等。消息视图如图 4。



Index	StartID	TargetID	Time	SendType	DataLength	MessageType	Data
1	commander	sub1	8：44： 8	up-->down	3	EM_INFORM	0 25
2	commander	sub2	8：44： 8	up-->down	3	EM_INFORM	0 25
3	commander	sub3	8：44： 8	up-->down	3	EM_INFORM	0 25
4	sub1	commander	8：44： 8	down-->up	1	EM_OK	
5	sub2	commander	8：44： 8	down-->up	1	EM_OK	

图 4 消息视图

4 Agent之间的协调与协作

Agent 是有一定的社会性和自主（治）性的，其社会性在 Agent 之间进行通信时体现得比较明显，而 Agent 的自治性主要体现在它们进行协调和协作的过程中。Maze 系统主要实现的是两组 Agent 之间的对抗行为，那么在它们相互进行攻击时，同一种类的 Agent 之间必然需要协调和协作。在 Maze 系统中，Agent 之间的协作性主要由 Agent 的体力临界值、信心临界值和个人成就感来决定，它们中的协作程度与临界值呈反比关系，临界值越小，合作性越强。而 Agent 的自治性主要由 Agent 的等级，以及其协作性（相反）的程度来决定。

针对 Maze 系统中 Agent 的种类不同，以及同一种类 Agent 的等级不同，在分析它们之间的协调行为时需要根据它们的等级来分析：对于不同级别的 Agent，上级 Agent 与下级 Agent 进行协调时，采用的是命令（Order），下级 Agent 必须强制执行上级的命令，然后把执行结果返回给上级 Agent；同一级别的 Agent 之间进行协调时，采用的是建议（Advise），对方可以评估一下这个建议，然后根据自己的情况可以接受这个建议也可以拒绝这个建议。以通信为载体，Agent 之间进行协调与协作时消息邮箱的伪代码如下：

```
BEGIN(程序开始)
while（IsEmptyMsgBox==False） //消息邮箱不空
{
  //取消息，查看发送者和接收者的级别
GetMessageId();
//上级给下级发
if(msg.SendType=0)  {

//传递到接收者的消息队列中
  SendMessage();
}
//对方能接收这个消息
else if （IsEnableMsg==True）{
//传递到接收者的消息队列中
    SendMessage();
}
    else {
          return false;    //报错
        }
//检测是否有下一条消息
AgentMessage msg = MsgBox.RemoveHead();
}
END(程序结束)
```

5 结语

本文基于 Maze 系统对 Agent 技术进行了实际的研究，在 Maze 系统中将 Agent 的属性进行了实例化，并根据这些属性对 Agent 之间的通信机制、协调以及协作经行了深入的研究。目前，Maze 系统已经能够顺利的实现 Agent 之间的对抗，在下一步的工作中，将结合新的 Agent 技术对 Maze 系统进行进一步的完善，将该仿真系统开发成插件式的系统，用户可以利用这个系统测试不同的协同算法和自治算法等，并将该仿真平台与煤矿安全评估进行结合，通过构建基于 Agent 的仿真环境，为专家进行安全评估提供量化的依据。

参考文献

1 Glaser N. The CoMoMSA Methodology and Environment for Multi-Agent System Development. In: Zhang C, Lukose D, eds. Multi-Agent Systems and Applications III. 1996.

2 Bellifemine F, Poggi A, Rimassa G.. Developing Multi-Agent Systems with a FIPA-Compliant Agent Framework. Software Practice and Experience, 2001,31:103−128.

3 薛霄.面向 Agent 的软件设计开发方法.北京:电子工业出版社,2009.

4 Nwana HS, Ndumu DT, Lee LC, Collis JC. ZEUS: A Toolkit for Building Distributed Multi-Agent Systems. Applied Artificial Intelligence Journal, 1999,1(13):129−185.

5 Poslad S, Buckle P, Hadingham R. The FIPA-OS Agent Platform: Open Source for Open Standards. 2000. http://fipa-os.sourceforget.net.

6 Kearney P, Stark J, Caire G, Garijo FJ, Jorge J, Sanz G, Pavon J, Leal F, Chainho P, Massonet P. Message: Methodogy for engineering systems of software Agents. Technical Report EDIN 0223-0907, Eurescom, 2001.

7 Object Management Group. CORBA 2.4.2 Specification. 2000a. http://www.omg.org