

# 网络故障管理中基于模型驱动的移动代理应用

智 勇<sup>1</sup>, 黄海平<sup>2</sup>, 王汝传<sup>2</sup>, 孙正林<sup>2</sup>

(1. 南京大学 管理工程学院, 江苏 南京 210093;

2. 南京邮电大学 计算机学院, 江苏 南京 210003)

**摘 要:**目前传统的网络管理都是采用集中式方式,存在很多缺陷,例如执行效率低下。文中针对传统网络管理系统的不足,提出了一种新的网络故障管理模式,并在此基础上介绍了一种基于心理学“刺激-反应”原理的模型驱动的移动代理开发方法。通过对该心理学原理的模拟,来指导移动代理的开发,并将“刺激-反应”模型贯穿移动代理开发中的分析、设计及实现过程。最后将移动代理应用到网络故障管理的实例中,并详细介绍了基于模型驱动的移动代理在网络故障管理中应用的开发过程。

**关键词:**模型驱动;移动代理;网络管理

**中图分类号:**TP393.07

**文献标识码:**A

**文章编号:**1673-629X(2009)10-0222-04

## Application of Mobile Agent Based on Model-Driven in Network Fault Management

ZHI Yong<sup>1</sup>, HUANG Hai-ping<sup>2</sup>, WANG Ru-chuan<sup>2</sup>, SUN Zheng-lin<sup>2</sup>

(1. College of Management Engineering, Nanjing University, Nanjing 210093, China;

2. College of Computer, Nanjing University of Posts and Telecommunications, Nanjing 210003, China)

**Abstract:** At present, traditional network management adopts centralized management, which has many disadvantages (e.g. low execution efficiency). This manuscript proposes a new mode of network failure management, and a methodology aiming at developing mobile agents based on “Stimulation-Reaction” model of psychology is also described. The manuscript presents the development of mobile agent by simulating the psychological theory. “Stimulation-Reaction” model runs through analysis, design and implementation of mobile agent development. The manuscript, by an example of network failure management, introduces the process of mobile agent development under the model in detail.

**Key words:** model driven; mobile agent; network management

### 0 引言

移动代理(Mobile Agent)是一系列代码与状态的集合,是具有智能的主体,它能够自行决策在网络各个节点之间移动,代表其它实体(人或其他 Agent)进行工作的一种软件实体<sup>[1]</sup>。它能自行选择运行地点和时机,根据具体情况,中断当前自身的执行,移动至另一设备上恢复运行,并及时将有关的结果返回<sup>[2]</sup>。移动的目的是使程序的执行尽可能靠近数据源,降低网络通信开销,节省带宽,平衡负载,从而提高分布式系统的处理效率<sup>[3]</sup>。

传统的基于 SNMP 的集中式网络管理,几乎完全依赖管理站对各个网络节点的主动查询来发现故障<sup>[4]</sup>。整个过程几乎都是管理站来承担,但随着网络规模的不断扩大,必然导致执行效率降低。采用移动代理机制可以弥补传统网络管理系统的不足,有效地平衡网络负载,优化网络性能,增强管理的灵活性<sup>[5]</sup>。

### 1 刺激决策反应模型

在心理学上,有一种理论,认为人或动物的行为是一种「刺激-反应」的模式。也就是说,外界出现的一种刺激,会引发人的某种反应。因此,人的行为可以使用许许多多的不同刺激导致不同反应的方式来描述。

可以考虑以一条条的刺激->反应的式子,来描述 Agent 的行为,是一种分解描述的方式。但可以预期的是,所有的反应式子加总起来,理论上应该构成该

收稿日期:2009-01-31;修回日期:2009-05-13

作者简介:智 勇(1976-),男,江苏泰州人,博士研究生,研究方向是通信经济理论和通信业务体系架构等;王汝传,教授,博士生导师,研究方向是计算机软件、计算机网络和网络、对等计算、信息安全、无线传感器网络、移动代理和虚拟现实技术等。

Agent 完整的行为。想要一次描述完整 Agent 的行为透过低阶的程序性的方式可能在某种情境下会显得困难,但如果该 Agent 的行为,适合利用这种分解成一条条刺激反应式来看待时,将会为行为建模带来很大的便利。

1.1 在刺激决策反应模型下进行 Agent 的分析

进行系统分析,可以依循下述的步骤进行:首先,初步找出需要的 Agent 类型;其次,决策每一种 Agent 类型的责任;第三,以粗略的观点来看各种 Agent 之间的互动,检查应用的目标是否被解决;第四,倘若目标尚未解决,便考虑增加新类型的 Agent 或是为已存的 Agent 类型增加责任;最后,倘若以粗略的观点来看,各种 Agent 之间的互动已能满足目标,则结束。

在识别出 Agent 的类型,以及大致的责任分工后,通过「Agent 属性表」来进一步描述 Agent 的性质,各字段的名称与用途于表 1 中描述。

表 1 Agent 属性表

Agent Name	填写 Agent 的名称
Responsibility	明确 Agent 的责任
Quantity	此种 Agent 在整个 Multi-Agent System(多移动代理系统)中的数量
Mother Agent	此 Agent 是由哪种 Agent 所派生的
Children Agent	此 Agent 会衍生哪些类型的 Agent
Collaboration	此 Agent 会和哪些类型的 Agent 合作
When to be born	此 Agent 被产生的时机为何
What to do after the birth	此 Agent 初生之际,会进行什么动作
When to be killed	此 Agent 何时会被结束生命
What to do before the death	此 Agent 在结束生命之前,会进行什么动作
When to move	此 Agent 何时移动
.....	.....

1.2 在刺激决策反应模型下进行 Agent 的分析

在刺激决策反应模型下进行 Agent 的设计,利用表 1 中 Agent 的属性,再转化成为所谓的 Agent 刺激反应式。每一条 Agent 刺激反应式都是由三个部分组成:Stimulation(刺激)、Info & Logic(信息与逻辑)和 Reaction(反应)<sup>[6]</sup>。

Agent 反应式的写法须依循下述的步骤:首先,列出原始的刺激;其次,写出每个刺激所引发的反应,并且记录做出此反应需要的辅助信息(记忆与刺激的内容)及判断逻辑;第三,列出反应所引发的新刺激;第四,检查是否所有列出的刺激皆有相对应的反应,若否则回到步骤 2,反之则接续下一步骤;最后写作结束。

1.3 在刺激决策反应模型下进行 Agent 的编程

依据 1.2 节可以找出每个 Agent 所需的刺激与反应类型。移动代理平台一般应该提供一些预先实现的刺激类,它们都应该继承自 Stimulation 这个抽象类<sup>[7~9]</sup>;例如 ClockStimulation:用来表示时间驱动的刺激;EarStimulation:用来表示收到其它 Agent 讯息的刺激;等等。MAP 提供下述的反应类,它们都继承自 com.vantech.map.model.Reaction 这个抽象类别;例如 Echo:将信息打印至 Console;MoveLeg:移动至另一个主机;等等。

根据 1.2 节中的反应式可以编写 Agent 的程序。主要方法如下:首先对每一反应式,皆以下述步骤处理;其次,找出刺激部分所对应的刺激类别;第三,找出反应部分所对应的反应类别;第四,判断反应部分是否提供以 .Net Attribute 方式来实现;第五,倘若反应部分能以 .Net Attribute 方式来实现,则在 Agent 模型的类别文件里以 .Net Attribute 的方式写下刺激与反应部分;最后,倘若反应部分无法以 .Net Attribute 方式来实现,则以 Attribute 来撰写刺激,并将它关联至某个处理的方法。

通过上述步骤,就可以用刺激决策反应模型来开发出各个 Agent。

2 模型驱动的移动代理开发方法在网络管理中的应用

本节以一个网络故障管理系统为例,利用 Multi-Agent System 来实现该应用,并说明刺激决策反应模型下的分析、设计和开发。

在第一阶段,分别识别出了 HomeAgent、Agent-Producer、FaultWatchAgent 以及 FaultFixAgent 等四类 Agent。其中 HomeAgent 为管理中心节点专有的 Agent,并负责与使用者互动的 GUI 沟通,接受来自使用者的指令。同时它也接受其它 Agent 的回报,包括 FaultWatchAgent 和 FaultFixAgent,并且有能力显示一些信息给使用者。而 FaultWatchAgent 负责定期监视自己所在的被管理节点是否有故障发生,倘若有的话,则回报给 HomeAgent。此外,当 HomeAgent 接收到 FaultWatchAgent 回报的故障信息,则根据故障管理策略(故障管理算法、故障历史信息比较)来决策故障处理方法,在必要的情况下可以通知 AgentProducer(移动代理产生器)产生 FaultFixAgent,移动到被管理节点上执行故障修复策略所指派的任务。

在识别出所需的 Agent 类型与分别的主要责任后,便可填写如下的 Agent 属性表(以 HomeAgent 为例),如表 2 所示。

表 2 HomeAgent 属性表

Agent Name	HomeAgent
Responsibility	Interact with a GUI and accept the commands from the user Receive and record the report from the FaultWatchAgents Receive and record the report from the FaultFixAgents Warn the user while the user is using the GUI
Quantity	1
Children Agent	FaultWatchAgent
Collaboration	FaultWatchAgent : watch the managed site for fault information AgentProducer: produce the FaultFixAgents by the request from HomeAgent FaultFixAgents: migrate to specified site to fix the existent fault
When to be born	When the user executes the GUI
What to do after the birth	Read the managed sites list Spawn the FaultWatchAgents
When to be killed	When the user decides to terminate the watch mission
When to move	Do not move
How/Where to move	
Periodical behavior	Refresh the managed sites list
When to spawn the children	[FaultWatchAgent]When the user requests to start watch mission
Is persistent?	Yes

由于整个应用系统中的 HomeAgent 只会有一份, 因此它的 Quantity 是 1。由于网络故障管理系统是对一批被管理节点进行管理, 因此必须在 HomeAgent 初始化的时候, 便加载这一份被管节点列表。而初始化的行为最适合用「What to do after the birth」字段来描述。由于被管节点列表可能会有所变化, 因此 Home-

Agent 必须定期的更新这一份列表, 而定期性的行为则描述于「Periodical behavior」中。可以先利用 Agent 属性表, 找出 Home-Agent 的原始刺激。由于 Home-Agent 在 After the birth 后会有行为, 因此它会对此种刺激有所反应。另外, 它具备了定期性的行为, 因此会对某个固定间隔的 Clock 有所反应。当使用者在 GUI 上下了一个指令要求开始或结束监督时, 也会自 GUI 传来刺激。当 FaultWatchAgent 回传信息, 回报发现故障信息时, 也会导致 HomeAgent 产生刺激。而当 FaultFixAgent 回报修复故障结果时, 也会导致 HomeAgent 产生刺激。所谓「原始的刺激」即为根据 Agent 属性表直接可以找出来的刺激, 例如 After the birth, On clock1 tick, User requests to start a watch mission 等等。

可以先依照原始的刺激构造表 3, 其中, Reaction 部分代表该刺激所引发的反应。例如 After the birth 会对应到 Read the managed sites list 以及 Spawn the FaultWatchAgents 两个动作。

表 3 HomeAgent 刺激构造表

Stimulation	Info & Logic	Reaction
After the birth		Read the managed sites list Spawn the FaultWatchAgents
On clock tick		Refresh the managed sites list
User requests to start a watch mission		Discover all alive site for watching
User requests to terminate watch mission	[Mem]the ID and location of the FaultWatchAgents	1. Request all FaultWatchAgent to terminate 2. Request the FaultFixAgent to terminate 3. Commit a suicide
A FaultWatchAgent warns the fault information in the managed site	[Stimulation]the fault info If the user is using the GUI	Report to the user Record the fault info
	[Stimulation]the fault info If the user is not using the GUI	Record the fault info
The FaultFixAgent reports the result of the fault - fixing	[Stimulation] the result of the fault - fixing If the user is using the GUI	Report to the user Record the result of the fault - fixing
	[Stimulation] the result of the fault - fixing If the user is not using the GUI	Record the result of the fault - fixing

表 3 中 Info & Logic 这个字段,以 User requests to terminate a watch mission 为例,它所对应的反应发生在使用者要求结束监控时,同时要求所有已送出去的 FaultWatchAgent 自行结束生命。因此,HomeAgent 会在自己的存储区中记录所有送出去的 FaultWatchAgent。由于反应的执行必须参考存储区的内容,因此 Info & Logic 这个字段中便填写了「[Mem]the ID and location of the FaultWatchAgents」。而像「A FaultWatchAgent warns the fault information in the managed site」这个刺激中则包含了示警的内容。因此在这个字段便注明了「[Stimulation]the fault info」。由于 HomeAgent 会依据使用者是否正在主控台旁边操作来决策是否要对使用者提出示警,而这是关系到 Logic 的部分,因此产生了一个分支(branch),对同一个刺激,会依据使用者是否正在主控台旁边操作,而有不同的反应。类似此类的 Logic,亦记载于 Info & Logic 字段中。

可以进一步整理每个 Agent 相关的刺激、信息逻辑与反应,把属于同类的合并在一起<sup>[10]</sup>:

HomeAgent

Simulation

Lifecycle Handler(生命周期控制器)

After the birth(产生后刺激)

Clock(时钟)

Periodical(周期)

Memory Change(记忆改变)

The managed sites list(管理节点列表)

All alive site for watching(所有监视的活动节点)

Face(交互)

A request from the user to start watch mission(用户请求开始监视任务)

A request from the user to terminate watch mission(用户请求中止监视任务)

Ear(外部)

A FaultWatchAgent warns the fault information(FaultWatchAgent 通知故障信息)

The FaultFixAgent reports the result of the fault-fixing(FaultFixAgent 报告故障修复结果)

Info & Logic

Mem the ID and location of the FaultWatchAgents(FaultWatchAgent 的识别号和位置)

the managed sites list(管理节点列表)

Simulation

The warning from a FaultWatchAgent(来自 FaultWatchAgent 的告警)

The result of the fault-fixing(故障修复结果)

Reaction

Tool

ManagedSitesListReader(管理节点列表读取器)

FaultInfoDiscoverTool(故障信息发现工具)

FaultInfoRecorder(故障信息记录器)

Reproduction(再生)

FaultWatchAgents

Remember(记忆)

The ID and location of the FaultWatchAgents(FaultWatchAgent 的识别号和位置)

Face(交互)

FaultInfoReport(故障信息报告)

其中 Tool 是进行某些反应时额外使用的工具,例如 Agent 会需要利用 ManagedSitesListReader 来读取被管理节点列表。而 Face 表示 GUI 系统,GUI 系统可能会传来刺激,也可以用来显示某些想要告诉用户的结果。而 Ear 代表 Agent 可能会接收到的信息。

### 3 结束语

文中提出了模型驱动的移动代理开发方法,影响的不单只是编程时的活动,还包括从分析、设计,一直到实现整体的思维方式。通过该方法可以实现 Agent 的高效开发,未来的工作是将 Agent 的设计进一步完善。

#### 参考文献:

- [1] 史忠植. 智能主体及其应用[M]. 北京:科学出版社,2000.
- [2] 张云勇. 移动 agent 及其应用[M]. 北京:清华大学出版社,2002.
- [3] 张云勇,刘锦德. 移动 Agent 技术[M]. 北京:清华大学出版社,2003.
- [4] 汤 博,杨 妮,刘人有. 面向网络管理的移动主体安全设施[J]. 软件学报,2003,14(10):1762-1767.
- [5] 王汝传,徐小龙,黄海平. 智能 Agent 技术及其在现代信息网络技术中的应用[M]. 北京:北京邮电大学出版社,2006.
- [6] Shosam Y. Agent-oriented programming[J]. Artificial Intelligence,1993(6):51-92.
- [7] 徐晓钟,谢康林. 基于 MDA 方法学软件开发方式的原理与实现[J]. 微机发展(现更名:计算机技术与发展),2004,14(4):15-19.
- [8] JACK. Intelligent Agents[EB/OL]. 2008-02. <http://www.agent-software.com/>.
- [9] Koehler J,Hauser R,Kapoor S,et al. A Model-Driven Transformation Method[C]//Proceedings of the Seventh IEEE International Enterprise Distributed Object Computing Conference (EDOC'03) 2003. [s. l.]: IEEE Computer Society Press,2003.
- [10] Wu Qishi,Nageswara S. V. R,Barhen J,et al. On computing mobile agent routes for data fusion in distributed sensor networks[J]. IEEE Trans. on Knowledge and Data Engineering,2004,16(6):740-752.