面向 Agent 的软件分析和设计方法*)

向郑涛[1] 缪育平[2] 鲁东明[1]

(浙江大学计算机科学与技术学院 杭州310027)[1] (杭州展望科技有限公司 杭州310003)[2]

摘 要 在复杂系统的分析、设计、开发过程中,面向 Agent 的方法在映射现实世界,模拟人认识问题、解决问题的过程等方面具有优势。从描述 Agent 的概念、分类和体系结构出发,对面向对象和面向 Agent 的分析设计方法进行了比较,讨论了面向 Agent 的分析模型及各模型的组成部分,以及面向 Agent 的分析模型到设计模型的转换,并指出面向 Agent 的分析设计方法提供了在建模、设计和实现不同级别上复用的基础。以政府公共行政服务系统为例对上述方法进行实例分析。最后得出面向 Agent 的软件分析和设计将为复杂系统提供新的解决思路的结论,同时指出本文提出的面向 Agent 的分析和设计方法的优点以及所面临的问题。

关键词 Agent,面向 Agent 的分析,面向 Agent 的设计,软件复用

Agent-Oriented Software Analysis and Design Method

XIANG Zheng-Tao[1] MIAO Yu-Ping[2] LU Dong-Ming[1]

(College of Computer Science, Zhejiang University, Hangzhou 310027)[1]

(Hangzhou Prospect Science and Technology Co., LTD., Hangzhou 310003)[2]

Abstract In the process of analysis, design, and development of complex systems, the agent-oriented method is good at mapping the real world, simulating the process of human beings' perceiving and resolving problems. The concepts, types, and architecture of agents are described. After comparing of methods of object-oriented and agent-oriented analyzing and designing, the agent-oriented analysis model and the design model are interpreted. Based on the agent-oriented analyzing and designing method, we can achieve the multi-level software reusing, which means the reusing of model, analysis and design. We examine the approach represented above in the context of an example of public administrative service system. Finally, we conclude that the agent-oriented software analysis and design is a new solution of complex system. Meanwhile, the advantages and problems of the agent-oriented analysis and design method are proposed.

Keywords Agent, Agent-oriented analysis, Agent-oriented design, Software reuse

1 引言

在软件工程的发展历程中,如何开发满足用户需求的、可靠的、可维护的、可复用的软件,一直是软件专业人员的目标。在这种思想的驱动下,软件工程从无到有;从传统的线形模型到原型模型,到演化模型;从传统的面向过程的方法到面向对象的方法,到现在正是研究热点的面向 Agent 的软件工程方法;软件工程方法学经历了复杂而富有争论的过程。

目前,软件开发者面临的一个选择是面向 Agent 的方法,尽管它在很多方面尚处于研究阶段,但它目前所表现的特性已经令人承认,从对现实世界的映射的角度来看,面向 Agent 的方法的确比以前的方法来得自然,将是分析、设计、开发复杂系统的有效工具[2]。本文第2节对 Agent 及其特点进行概述,第3节将从面向 Agent 的分析 AOA(Agent-Oriented Analysis)和面向 Agent 的设计 AOD(Agent-Oriented Design)的角度来说明面向 Agent 方法的这一自然开发的特点,并对其可复用性进行讨论。第4节通过一个实例来对本文提出的面向 Agent 的分析和设计方法进行说明。最后作出面向 Agent 的分析设计方法将为复杂系统提供新的解决方案的结论,指出面向 Agent 方法的优势所在,以及其所面临的问题。

2 Agent 概述

2.1 概念

Agent 是一类在动态环境下能感知环境,并能自治地连续运行代表其设计者或使用者实现一系列目标的计算实体或程序[1]。这里需要指出的是 Agent 处于环境中,其自治能力使得 Agent 能响应外界环境的变化,而不依赖人工干预作出灵活和智能的反应。

根据上面的定义,一个 Agent 的最基本的特性应当包括:自治性、反应性、目标性和针对环境性[1]。每个 Agent 首先应具备这4条最基本的特性,然后再根据其应用情况拥有其他特性,如移动性、自适应性、知识级的通信能力、理性、持续性或时间连续性、自利等特性[1,3]。这些特性使得 Agent 在软件设计中具有独特的优势,特别是其自治性、移动性等特性使得基于 Agent 的软件体系具有更为松散的耦合度。

2.2 分类

从 Agent 思考性的角度出发,单个 Agent 的结构通常分为思考型 Agent、反应型 Agent 和混合型 Agent[4]。从 Agent 的移动性[4]来看,可以分为移动 Agent 和静止 Agent。但是从对现实世界的映射来看,可以将 Agent 分为主动 Agent 和被动 Agent。现实世界中既存在着施动的实体,如个体的人,也存在着受动的实体,如建筑等。而在分析建模中,可以分别对应主动 Agent 和被动 Agent。主动 Agent 是可以对环境施加影响的实体;而被动 Agent 只能接受环境的影响,改变自身的状态。

2.3 体系结构

*)基金项目:国家"八六三"高技术研究发展计划(2002AA4Z3360)。向郑涛 硕士生,讲师,从事分布式计算、人工智能研究。缪育平 从事计算机网络、软件工程方法学研究。鲁东明 博士,教授,从事计算机网络、人工智能研究等方面。

目前大多数研究者倾向于把 Agent 定位为有意识系统，所谓有意识，指 Agent 的行为能通过归因于信念、期望和理性的方法去预言。因此，Agent 的体系结构在很大程度上决定于其认知结构，即如何形式化描述 Agent 的各种思维属性及其之间的关系，以及 Agent 规划、行为、协调、合作等活动的关系，使不同的 Agent 具有不同的意识，从而使 Agent 显示出智能性。目前比较成熟的 Agent 认知结构是由 Rao 和 Georgeff 提出的 BDI(Beliefs，Desires，Intentions：信念，期望，意图)模型，用信念、愿望和意图这三类意识态度来作为 Agent 行为规划的依据[5]。当然，一个完整的 Agent 还应该具有感知、推理、学习、通信、执行、知识库等模块，而上面的认知结构，也就是心智及其管理模块是 Agent 的核心。在这种讨论之下，给出 Agent 的体系结构模型，如图1所示。

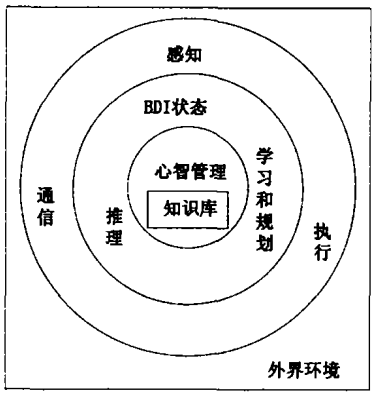

图1 Agent 体系结构模型

3 面向 Agent 的分析和设计

3.1 概述

从某种角度讲，可以认为 Agent 概念是 Object 概念的自然延伸，因此，面向对象分析 OOA (Object-Oriented Analysis)、面向对象设计 OOD(Object-Oriented Design)中的某些分析方法和设计技术可以运用到 AOA 及 AOD 中来，作为相关讨论的起点。与 OO 相比，AO 有以下几个主要优点[6~9]：

(1)*基本思想* OO 强调直接以问题域中的事物为中心来思考、认识问题，通过对事物本质特征的提取和抽象来定义类，并在继承的基础上实现类层次的划分。通过对类内部属性和操作的定义，实现了对数据和操作的封装。操作是对象的一部分，但不是自治的，换言之，OO 的操作是被动的。

AO 是从现实世界中事物的相互作用出发，认为事物不是完全孤立的，而是与所处的环境有十分密切的关系，而这种关系应该体现在分析和设计中[10]。因此强调不仅将数据和操作进行封装，还应该将心智和推理封装，而且 Agent 不仅对自身的状态进行控制，还对自身的行为进行主动控制。在这种思想下，将事物抽象为 Agent，作为系统的构成单位，通过 Agent 间的协作和妥协实现系统的整体目标。

(2)*交互方式* OO 系统中对象的交互是以函数调用为基础的，多个对象在设计时就已经被"绑定"在一起，实际上是被动行为，这不仅使 OO 系统的耦合性较高，也使系统的适应性和灵活性受到很大制约。

在 AO 系统中，由于引入了信念、期望、意图等心智状态和推理、学习等心智活动，将主观意识和客观世界紧密结合，使得 Agent 的交互模式是主动的，从而可以基于实际环境和其他 Agent 进行交互，这表明多个 Agent 是相对独立的实体。另外，Agent 还可以具有移动性。因此，相对 OO 系统而言，AO 系统是松散耦合的；而且，由于 Agent 的自治性，AO 系统在运行中具有很强的适应性和灵活性。现在已经有 Agent 交互的标准协议 ACL (Agent Communication Language)，目前最著名的 ACL 是 KQML(Knowledge Query and Manipulation Language)[11,12]。在 Agent 的交互中，KQML 不仅传递消息本身，而且能通过定义丰富的消息类型及它们的语义去提出接收者应如何处理消息内容和如何应答的期望，以增强可理解性，促进通信的协调，能够实现知识级的通信。

(3)*分析与设计* 目前从对象技术衍生出了许多 OO 方法，但都没有提出一个划分系统粒度的标准，因此导致在实际系统分析中，不同的人的结果会有很大差异。

AO 则将系统作为问题求解的整体来看待，强调的是问题的分解、分配以及求解过程的协同，这与客观世界的真实情况和人类认识、解决问题的习惯和方法更为接近。这种对现实世界更为自然的映射使我们能够采用知识级系统分析与设计的思想，有助于我们对复杂系统行为的预计和理解。

3.2 面向 Agent 的分析

AOA 的作用是描绘客观现实的概貌，但 AOA 并不能一次到位，因此这样的分析是演进式的。

(1)*需求收集* 在这一阶段，客户和开发者在一起定义基本的系统和软件需求。在描述系统的总体目标后，进行环境分析，为 Agent 定义好活动平台。在此基础上，标识系统的参与者，即 Agent 和用户[13]，并基于系统的运作方式描述用户和系统以及 Agent 之间的交互关系。

(2)*Agent 体系结构模型* 基于 Agent 的体系结构，对 Agent 的基本特性进行刻画，如类型(主动或被动)、代理特征(移动性、思考性、担任的角色等)，并对 Agent 的 BDI 状态、推理、感知、反应、知识库等部分进行描述，这是 Agent 能够通过感知从而产生反应动作、完成特定功能、实现 Agent 之间交互的基础。

(3)*代理-能力-条件-协作者(AACC)模型* 这个模型刻画了 Agent 行为的外因，与 OO 的类-责任-协作者(Class-Responsibility-Collaborator)模型[14]类似，代理-能力-条件-协作者(Agent-Ability-Condition-Collaborator)模型提供了标识和组织系统中 Agent 的手段。其中，"代理"的内容有：代理名、代理类型、代理特征等；"能力"表明了 Agent 在给定场景下可以产生的活动，当然这只是 Agent 产生活动的必要条件，还必须满足充分性，Agent 才能去执行动作，这个充分性就是"条件"，这就界定了 Agent 的动作范围，也就是说，Agent 的活动必须是有能力和约束的，这也与现实世界的活动是一致的，即什么样的环境下(条件)什么人(Agent)可以做什么事情(能力)；"协作者"是 Agent 在合作时的关系，由于 AO 系统的特点，这种关系是双向的。可以仿照 OOA 的 CRC 索引卡片，得到 AOA 的 AACC 索引卡片，如表1所示。

表1 AACC 模型的索引卡片

Agent 名：		
Agent 类型：主动或被动		
Agent 特征：移动性、思考性、担任的角色		
能力：	条件：	协作者：

(4)*代理-交互模型* 这是对 Agent 之间以及 Agent 与平台之间关系的描述，可以参照 AACC 索引卡片刻画出协作者

网络，这也定义了协作者之间的消息路径。交互模型可以由以下属性组成：

·协作目标：交互的期望结果；

·发起者：负责发起交互的 Agent；

·响应者：合作 Agent 或平台；

·条件：交互所应满足的条件；

·异常处理：当交互的目标由于某种原因（如某 Agent 放弃协作）未实现时协作的 Agent 组应采取的行动，是放弃交互还是重新分析以开始新的交互，重新交互通常是协商的过程。

·协商策略：当需要重新交互时，需要重新定义响应者或条件，以从另外的途径实现协作目标。

·影响：交互对环境的影响。

在这个模型中，突出了 AO 系统的思想，即现实世界是联系的，而不是孤立的，因此，Agent 不仅需要和合作 Agent 进行交互，而且需要和平台进行交互，同时要反映出交互所带来的影响。在交互过程中，可能会出现各 Agent 之间的冲突，从而导致协作失败；因此需要考虑如何消解冲突，这是通过各 Agent 之间进行协商以及采用一定的妥协策略来实现的。

(5)代理-感知-反应(ASA)模型　这个模型实际上是对 Agent 行为的内因刻画。作为自治的软件实体，Agent 具有对外界环境感知的能力。在此情况下，在知识库的基础上，结合条件对所感知的环境进行推理，从推理的结果得到应该做出的反应，即行为。由此得到的代理-感知-反应（Agent-sense-action）模型可以用索引卡片的形式进行描述。

但这并不意味着可以对 Agent 的行为及影响进行完全的预测，这是与 OO 完全不同的。在 OO 中，由于 Object 的行为被动性，可以对事件按时间的顺序进行安排，因此 Object 的行为是可知的。而在 AO 中，由于 Agent 是主动了解环境的变化，因此其行为的时序性是未知的，导致了行为后果也是未知的；但可以通过条件（即约束或行为准则）来进行宏观上的控制，以使系统免遭毁灭性的灾难。

3.3　面向 Agent 的设计

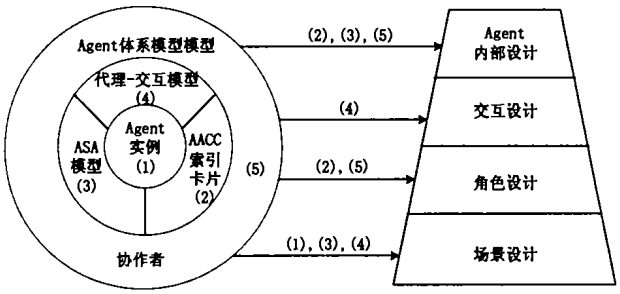

图2　AO 分析模型到 AO 设计模型的转换

AOD 是将用 AOA 所创建的分析模型转化为构造现实世界蓝图的设计模型。与 OOD 相比，AOD 在抽象、信息隐蔽、功能独立性和模块性等方面能达到更高的要求。我们用图2表示 AO 分析模型到 AO 设计模型的转换。

上图中，转换是自底向上的。首先对 Agent 活动的每个场景进行转换，场景的总和是 Agent 活动的平台，每一个场景对应系统的一个模块，使得客户需求得以体现。其次进行角色设计，每个角色可以对应多个 Agent，而多个 Agent 也可以因环境的不同而对应于多个角色。再次进行 Agent 的交互设计，对代理-交互模型进行实例化。最后进行各个 Agent 的内部设计，基于对 Agent 的角色理解，对 Agent 体系结构进行实例化。

(1)场景设计　在场景中，需要明确有哪些 Agent、Agent 实例之间和 Agent 与环境之间存在哪些影响并如何反应。因此，在场景设计中，要描述场景中应出现的 Agent 实例，而实例之间和 Agent 与环境之间的影响以及 Agent 的反应是通过 Agent 的交互和反应来实现的，即要将 ASA 模型和代理-交互模型转换到场景中。这需要将分析模型进行划分，得到不同的场景，并定义场景之间的交互接口。为保证系统设计的可靠性，场景应分层设计，而且最上层的场景不应该太多；场景下可以继续划分为多个子场景。

(2)角色设计　通过对 AACC 索引卡片和 Agent 分析模型的抽象，提取公共模式，可以以通用的形式来逼近特殊需求，即以角色来表示通用的功能以及其能够解决的问题，这有利于软件复用。基于角色设计和场景设计，描述 Agent 与角色之间的对应关系，这种关系可以是一对多、多对多的。

(3)交互设计　对 Agent 之间和 Agent 与平台之间通信的细节进行设计，这不仅仅包括通过结构化的形式来传递消息本身，而且要设计相应的语义信息。KQML 消息格式为[4]：

(〈performative〉{:〈参数关键字〉{〈单词〉|〈表达式〉}}*)

其中，“performative”表示消息类型，不仅包括 ask、tell、reply 等常用类型，还包括信息流类型（stream-one，stream-all）、网络类型（forward，register，pipe）、兴趣和能力陈述类型（advertise，subscribe，monitor）以及代理类型（broker，recruit）等。

参数包括 Content（消息内容），language（内容信息的表示语言），ontology（内容信息遵循的本体论），reply-with（指示消息发送者期望的应答标记），sender（消息发送者），recervier（消息接收者）等。

通过丰富的消息类型，KQML 能够表示代理-交互模型中的各组成部分；通过设计 KQML 消息的具体内容，如接收者、发送者以及协作目标、协商策略等各种消息内容，促进了信息流的结构化，提高了通信的有效性。

(4)Agent 内部设计　在对 Agent 角色的理解基础上，对 Agent 体系结构进行实例化，同时包括算法和数据结构的设计。具体内容包括 Agent 所应具有的意识态度、所采用的数据结构、Agent 动作的描述以及算法实现、Agent 从信念到动作的转化方法（如规划设计）、协商以及冲突消解方案设计等。

3.4　AO 方法的复用性

软件复用是指利用已有的软件资源来进行软件系统的开发[15]。由于软件复用可以提高软件生产率和软件系统的质量，降低开发成本，因此现有的软件开发技术几乎都提供了相应的机制来支持复用，如面向对象技术、构件技术等。

就 AO 方法而言，由于 Agent 具有的良好的封装性，从而达到高层次的抽象、高度的信息隐蔽，使得 Agent 之间的耦合极为松散，因此能更为有效地支持软件复用。一旦 Agent 被定义，则形成了在建模、设计和实现不同级别上复用的基础。由于 Agent 是系统中的构成单位，因此系统的资源必须分布在各个不同的 Agent 中，Agent 之间通过交互和协作来对系统中的资源进行复用，这在分布、开放的系统中尤为重要，由于 Agent 的主动性，这种复用是动态的、不确定的。

4　实例分析

我们将上述的 AOA 和 AOD 方法应用到电子政务领域中的政府公共行政服务系统中进行验证，该项目得到863项目“基于国产数据库系统的政府公共行政服务系统”的支持。政府公共行政服务系统是面向企业和个人的电子政务应用系统，其目的在于为企业和个人提供电子化的政府部门协作办公，提供个性化、整合式的政府公共行政服务，建立政府、企业

和个人三者互惠的电子化政府公共行政服务环境。

4.1 政府公共行政服务系统的AOA

政府公共行政服务系统的要求为：

(1)系统中包括四类用户：社会公众(企业和个人用户)、负责行政服务业务受理的工作人员(业务受理人员)、负责行政审批事务的工作人员(行政审批人员)和负责监督管理行政服务的管理人员(服务监督人员)。

(2)社会公众：通过发布服务获取各种信息，如办事程序、下载表格等；通过咨询服务来向办事人员咨询；在行政审批业务进行时，通过业务状态查询服务获取审批业务的进行情况。

(3)业务受理人员：定制行政审批业务流程；通过受理服务接受用户的审批业务，根据业务情况确定是否需要并联审批；将审批的结果(包括是否成功、失败原因等)返回给社会公众。

(4)行政审批人员：当一个业务进入并联审批流程后，业务数据将被分发给涉及不同部门的业务审批服务，由各部门的行政审批人员进行审批；当由于某些原因导致审批失败(如用户数据错误、缺失或不符合审批条件等)，则将失败原因返回给业务受理人员。

(5)服务监督人员：监督和统计分析所有的行政审批业务；处理社会公众对审批的投诉。

根据上面的要求，系统将包括六个角色：个人代理、企业代理、业务受理代理、审批代理、服务监督代理、工作流代理。其中，将个人和企业代理分开的原因是两者所能够申请的审批项目和所要求提交的审批数据不同；工作流代理将屏蔽其他代理对工作流系统的访问；一个业务需要多个审批代理协作审批，审批代理代表审批部门的审批人员。

在Agent角色的基础上，对每个角色的基本特性进行描述。如业务受理代理能够感知个人和企业代理的审批请求和提交的审批数据，并通过知识库进行推理得到应该启用的审批流程，同时通知工作流代理启动相应的工作流程；工作流代理控制着审批流程的推进，不仅要感知审批流程的当前位置，而且要将有关的审批数据分发给各个审批代理；审批代理需要对工作流代理的审批驱动进行响应，根据审批知识库进行业务审批，同时将审批结果返回给业务代理。

由于一个审批业务需要多个审批代理共同完成，因此，每个审批代理都需要和其他审批代理通过协作完成审批业务。以个人注册餐饮店的一个简化审批业务为例，审批部门涉及到工商局、交通局和卫生局，审批协作过程如图3所示。

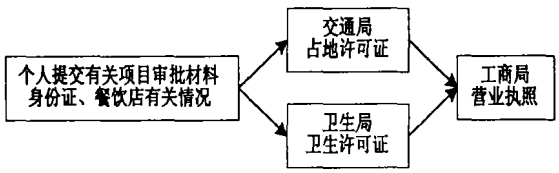

图3 个人注册餐饮店的简化行政审批协作过程

为完成这个审批过程，个人代理、交通审批代理、卫生审批代理、工商审批代理需要协作，工商审批代理能够通过审批、发放营业执照的条件是交通审批代理和卫生审批代理同时通过审批，并提交有关证明：占地许可证和卫生许可证。

在AACC模型的基础上，设计代理-交互模型。这里主要考虑异常处理和协商策略。以审批过程为例，当个人提交的审批信息缺失或错误时，或者审批请求不满足审批条件时，审批流程发生异常。审批代理对异常进行判断，如果是由于缺失或错误信息引起的异常，审批代理将暂停审批业务流程，通过业务受理代理和个人代理进行协商，要求个人提供所需要的信息；如果是由于审批条件不满足所引起的异常，审批代理将停止审批业务流程，并将审批结果通知业务受理代理。

知识库在ASA模型中占有很重要的地位。无论是审批代理对审批请求的审批，还是服务监督代理对用户投诉处理，都和知识库有关。以服务监督代理为例，当其感知到个人代理的投诉时，如对审批结果不满意，监督代理将基于知识库，根据用户投诉得到与其投诉相关的审批信息，判断审批结果是否合理，并产生相应的动作，或者是给个人代理返回审批无误的信息，或者是要求有关的审批代理给出相应的解释。

4.2 政府公共行政服务系统的AOD

在进行AOD时，考虑到UML的描述能力，我们采用了自顶向下的设计方法，以UML作为设计工具。

(1)首先进行场景设计。根据系统分析，设计四个场景：门户、业务受理、审批、监督等场景。其中，门户场景为个人和企业提供界面接口，通过个人代理、企业代理完成信息获取、信息查询、投诉等功能；业务受理场景由业务受理代理获取个人代理和企业代理提交的审批请求和信息，并返回有关审批结果；审批场景由审批代理对审批请求进行处理，由工作流代理进行流程驱动，并向业务受理代理返回审批结果；监督代理则获取个人和企业代理的投诉，并和有关审批代理进行交互。系统用例如图4所示。

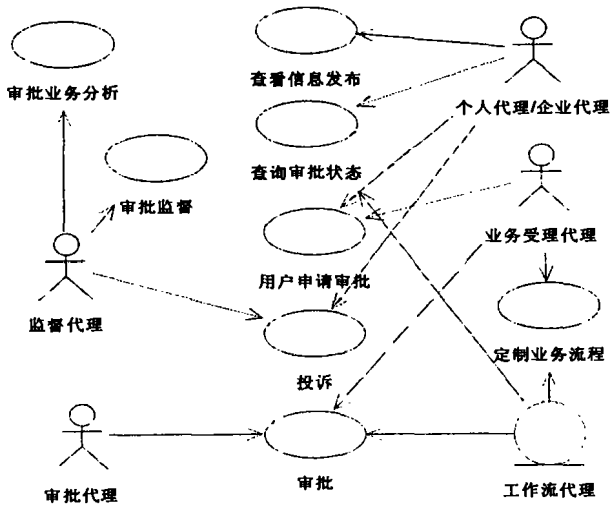

图4 政府公共行政服务系统用例图

(2)对个人代理、企业代理、业务受理代理、审批代理、服务监督代理、工作流代理等六个代理角色进行设计，这需要根据前面的分析结果将角色与Agent实例进行对应。在本系统中，角色与用户的关系为多对多，即一个角色可以对应多个用户，一个用户也可以对应多个角色。例如审批代理角色对应工商审批代理、卫生局审批代理等；由于审批业务存在主办单位和协办单位，一般主办单位承担业务受理和本单位的审批工作，因此某个用户既可以是业务受理代理角色，也可以是审批代理角色。

(3)Agent之间的消息交互在UML中可以表示为时序图。例如一个审批业务从提起审批请求到返回审批结果的过程如图5所示。

消息可以通过结合KQML和XML来实现[16]。KQML实现内容层面的封装，用XML封装KQML，实现通信层面的封装。例如，个人Agent A向业务受理Agent B查询200307130020号业务(表示2003年7月13号第20号业务)的审批情况时，封装成XML的KQML语句为：

```
(performative)
  (name)ask-one(/name)
```

```
〈sender〉A〈/sender〉
〈receiver〉B〈/receiver〉
〈in-reply-to/〉
〈reply-with〉N1〈/reply-with〉
〈content〉
        〈approval-query〉
                〈number〉200307130020〈/number〉    〈!--审批
                业务号--〉
                〈result inquire="yes"/〉    〈!--要求返回的审
                批结果--〉
                〈remark inquire="yes"/〉    〈!--要求返回的审
                批结果说明--〉
        〈/approval-query〉
〈/content〉
〈language〉XML〈/language〉
〈ontology〉EGOV〈/ontology〉
〈/performative〉
```

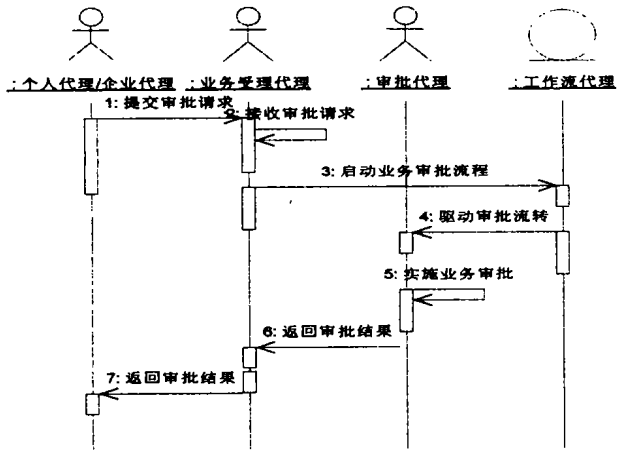

图5　行政审批时序图

(4)最后进行 Agent 的内部设计，本系统中主要对 Agent 的行为及状态转换进行设计。当 Agent 接收到其他 Agent 发出的消息时，通过判断 Agent 处于何种状态，并根据当前的条件以及行为策略，决定对消息进行何种处理。例如，个人 Agent 在提交审批申请后，经进入等候结果状态，在此期间，可以进行审批查询或提交其他审批请求的操作，当审批结果被返回时，则其状态调整正审批预结束状态；如果对审批结果有异议，则进入投诉状态，反之则进入审批结束状态。其状态转换如图6所示。

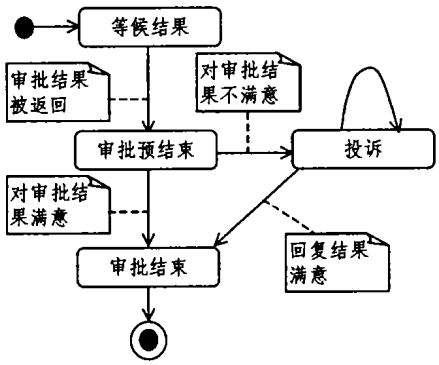

图6　个人 Agent 审批过程的状态转换图

基于 Agent 的体系结构和状态转换图，可以设计 Agent 的逻辑结构。信息获取模块将捕获其他 Agent 发送的消息和取得有关的环境信息。推理模块将根据当前的状态、条件以及行为策略判断所执行的行为，这里需要和本地知识库进行交互。行为执行模块将推理模块的行为指示转化为实际操作，并通过通信模块和其他 Agent 交互。Agent 管理模块将根据推理结果和行为执行结果，改变当前状态，并根据行为情况进行知识学习和行为修正，如由于个人代理对审批条件的理解错误导致审批不能通过时，个人代理将进行投诉，但是通过得到的投诉回复，个人代理将重新理解审批条件，进行知识学习，并修正有关行为。Agent 的逻辑结构如图7所示。

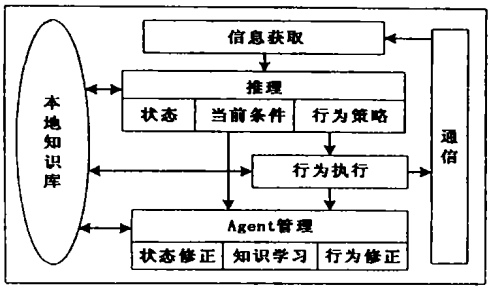

图7　Agent 的逻辑结构图

结论　面向 Agent 的技术在以自然的方式映射现实世界，模拟人认识问题、解决问题的习惯和方法等方面显示了巨大的潜力，而且由于其本身的高层次抽象特性使得 AO 方法具有良好的可复用性，因而受到了广泛的关注。如何用软件工程的思想来指导 AO 系统的开发，使 AO 技术成为复杂系统的解决方案，目前还没有统一的标准，因此，在 AOA 和 AOD 等方面进行的研究和探索，有待于在实践中检验和完善。本文阐述了一种面向 Agent 的软件分析和设计方法，并通过政府公共行政系统这一实例加以验证，取得了较好的设计和分析效果。但是由于系统的复杂性，本文提出的 AOA 和 AOD 方法也面临着一些问题，如由于高度自治性所导致的 Agent 冲突消解问题，由于 Agent 行为在时序上的不确定性而导致的尤为复杂和困难的 AO 系统测试问题，以及 AO 系统的可控性等，这些都将是日后的研究内容。

参考文献

1　刘大有，杨鲲，陈建中. Agent 研究现状与发展趋势. 软件学报，2000，11(3)：315～321

2　聂亚杰，刘大昕. 面向 Agent 的软件工程. 小型微型计算机系统，2002，23(4)：417～420

3　顾绍元，祝琛琛，施鸿宝. 一种基于 Agent 的需求分析与建模方法. 计算机工程与应用，2001，37(5)：38～40

4　高济. 基于知识的软件智能化技术. 浙江：浙江大学出版社，2000

5　Rao A S，Georgeff M P. BDI agents：from theory to practice. In：Georgeff M P，ed. Proc. of the 1st Intl. Conf. on Multi-Agent Systems (ICMAS-95). San Francisco. ACM Press，1995. 312～319

6　赵龙文，侯义斌. 智能软件：由面向对象到面向 Agent. 计算机工程与应用，2001，37(5)：41～43

7　段永强，张申生，高国军. 基于对象的软件代理的设计和实现. 计算机工程，2000，26(5)：21～25

8　Wooldridge M J. Agent-based software engineering. IEEE Transactions on Software Engineering，1997，144(1)：26～37

9　吴元斌. 面向对象技术与面向 agent 技术的比较研究. 计算机工程与应用，2001，(19)：137～139

10　Ciancarini P，Wooldridge M. Agent-oriented software engineering. In：Proc. of Intl. Conf. on Software Engineering (ICSE-2000). Los Alamitos. 2000. 816～817

11　Finin T，Fritzson R，McKay D，et al. KQML as an agent communication language. In：Proc. 3rd Intl. Conf. on Information and Knowledge Management，Gaitherburg Maryland，1994

12　Finin T，Labrou Y，Mayfield J. KQML as an agent communication language. In：Bradshaw J，ed. Software Agents. Cambridge：MIT Press，1997

13　Liu A，Lin F. A study in analysis and design of multi-agent systems. In：Proc. of Intl. Symposium on Multimedia Software Engineering (ISMSE-2000). 2000. 225～228

计算机科学2004Vol.31№.6

程序时序属性的自动测试

马晓东 董 威 王 戟 齐治昌
(国防科学技术大学计算机学院 长沙410073)

摘 要 测试预言是一种用来检测被测系统的测试执行是否正确的方法。文中,作者设计并实现了一种根据程序的线性时序逻辑(LTL)的性质产生测试预言的方法。首先,作者将一线性时序逻辑公式转换为一个有限状态自动机,然后,管理源代码,以便抽取与线性时序逻辑性质有关的状态序列。最后,用该信息来模拟状态自动机,并决定程序执行是否满足线性时序逻辑的性质。

关键词 测试预言,线性时序逻辑,FSA

Testing of Temporal Properties of Program

MA XIAO-Dong DONG Wei WANG Ji QI Zhi-Chang
(College of Computer Sicence,National University of Defense Technolgy,Changsha 410073)

Abstract Test oracle is a method of checking whether the system under test has behaved correctly on a particular execution. In this paper, we design and implement a method of producing test oracle from program's LTL(Linear Temporal Logic) property. First, we transfer a LTL formula to a finite state automaton; then, we manage the source code in order to extract state sequence which is relative to the LTL property; in the end, we use this information to stimulate the automaton and decide whether the execution of the program satisfies the LTL property.

Keywords Test oracles, LTL, FSA

1 引言

反应式系统通常由一系列与环境交互的行为组成,通过反应动作来响应外界的激励。这种连续的交互特性使得它与传统的顺序程序不同,后者通常把某一输入转换为输出;它们的另一个不同在于反应式系统通常是并发执行的,而并发系统中所存在的不确定性使得对它属性的描述和测试都变得更加困难。因此,对于反应式系统而言,常用系统中事件发生的相对顺序来描述系统属性。有多种逻辑可以描述这样的属性,而使用最广泛的是由 Pnueli 于1977年引入的时序逻辑(Temporal Logics)。时序逻辑主要分为两类:LTL(Linear Temporal Logic)和 CTL(Compute Tree Logic),它们的表达能力是相交的,没有包含与被包含的关系[1]。时序逻辑提供了对于系统在时间顺序上的行为的公式化描述,使得我们能够利用公式来准确地刻画系统的行为。对于反应式系统而言,测试它的时序逻辑属性很重要,本文主要提出了一种测试系统的 LTL 公式属性的方法及其实现。

测试预言(Test oracle)是用来判断被测程序的测试执行是否正确的方法,判断标准就是描述程序性质的规范。测试预言要做到对规范中的约束的覆盖,但又不能施加额外的约束,而且还要做到高效率的检验。对测试预言的这些要求是相互冲突的,这就要在其实现时做出折衷。理想的测试预言应该能够依据对程序预期行为的自然描述准确地判断程序执行的正确与否。

测试预言的种类有很多。有的通过嵌入式断言来实现,该方法把所期望的表达式直接插入程序源代码中去。典型的嵌入式断言可以在程序的任何控制点放置待检测的属性,C 语言中的 assert 宏可以看作嵌入式断言的原型。有的测试预言是基于规范描述文档,该方式下程序的规范描述与实现分离,这可以使得程序的规范描述语言与程序的实现语言之间的联系更松散,或者完全与程序的实现语言无关[2]。本文中实现的是对 LTL 公式所描述的性质进行判断的一种测试预言,可以用于测试并发反应式系统。

2 测试预言框架

测试预言的种类不同,其实现框架也不相同。对于嵌入式断言系统,在实现时要考虑的主要有下列几个问题:首先,嵌入式断言的通常做法是在特定的位置插入一个表达式,但是对于那些与控制点无关的全局属性,则不是在程序的所有控制点都插入表达式,而是通过为整个过程加上前置条件或后置条件来实现;其次,需要状态缓存,某些过程的后置条件要用到执行该过程之前的程序状态或变量的值,这就要求对这些信息保存一个副本供以后使用;再次,为了保存副本,需要使用一些原本不存在的辅助变量;最后,对于量词的处理,通常使用循环的方法,但是该方法并不总是有效。对于规范描述文档和程序分离的情况,首先要对规范文档进行分析,从文档中生成可执行的判断标准;然后处理待测试系统的源代码,或者对其进行运行时的监控,提取状态信息,实现测试预言[2]。

马晓东 硕士研究生,研究方向为软件测试与验证;董 威 博士,研究方向为软件可靠性与软件工程;王 戟 教授,研究方向为软件理论与软件工程;齐治昌 教授,研究方向为软件理论与软件工程。

14 (美)普瑞斯曼(Pressman,R.S.)著,黄柏素,梅宏译. 软件工程-实践者的研究方法. 北京:机械工业出版社,1999
15 毛新军,王怀民,齐治昌. 基于 Agent 技术的软件重用模式. 计算机科学,2001,28(3):90~92
16 魏晓斌,Unland R,Bacmendo B. 一个基于 XML 的 Agent 通信框架. 计算机应用研究,2001,10:108~112