面向 Agent 的程序设计*

姚 郑 高 文

(哈尔滨工业大学计算机科学与技术系 哈尔滨 150001)

摘要 本文针对所谓合作 Agent 应用问题阐述了一种面向 Agent 的程序设计 AOP(agent oriented programming)方法框架. 其中提出了一种新的 Agent 编程语言(AOPL),设计并实现了其程序设计系统(AOPS),该系统支持 AOPL 到 C 的转换. 同时,提出了一种新的 Agent 关系模型,讨论了该模型的组成及其在体现合作 Agent 应用系统的体系结构、指导 Agent 之间的协作行为和支持对系统结构特点的深层理解方面所发挥的重要作用. 最后讨论了 AOP 在多功能感知系统中的应用.

关键词 Agent,面向 Agent 程序设计,面向 Agent 程序设计语言,Agent 关系模型.

中图法分类号 TP312

随着 Agent 相关研究工作的深入进行,所谓合作 Agent 应用研究受到了高度重视,其特点是系统中的 Agent 都服务于某个共同目标,Agent 总是愿意同其它 Agent 合作,它们有意避免冲突的发生,系统整体目标的实现取决于 Agent 之间的合作(在此,Agent 可以是人类 Agent,也可以是软件 Agent). 目前,有关这方面的研究主要集中在 Agent 模型/语言和 Agent 关系模型两个紧密相关的方向上,Agent 模型/语言用于定义和构造 Agent,Agent 关系模型则通过 Agent 间的相互依赖性和相互关系来表现,由多个合作 Agent 组成的系统的体系结构已经取得许多成果. 它们大多致力于解决"如何做"这样一个问题,即如何定义和构造 Agent,如何定义与实施多 Agent 间的相互作用关系,从而实现系统的总体目标. 然而,当我们在改进或重新设计这样一个多 Agent 系统时,常常还需要对系统有更深层的理解,即理解那些隐藏在"如何做"后面的理由,这种需求已经在软件设计方法学中得到普遍认同.[1]基于上述讨论,本文以近年来出现的面向 Agent 的程序设计 AOP(agent oriented programming)技术为基础[2],提出了一种 AOP 方法框架,该框架兼顾了 Agent 模型/语言和 Agent 关系模型两方面问题,试图为解决合作 Agent 应用问题提供新途径. 本文第 1 节详细介绍 AOP 方法框架,第 2 节讨论了 AOP 在多功能感知系统开发中的应用;第 3 节总结和介绍下一步要进行的工作.

* 本文研究得到国家 863 高科技项目基金和国家教委跨世纪人才计划基金资助. 作者姚郑,1969 年生,博士生,主要研究领域为面向 Agent 技术,软件工程. 高文,1956 年生,博士,教授,博士导师,主要研究领域为人工智能,智能计算机接口.

本文通讯联系人:姚郑,哈尔滨 150001,哈尔滨工业大学计算机科学与技术系

本文 1996-12-18 收到修改稿

1 面向 Agent 的程序设计

面向 Agent 的程序设计是一种以计算的社会观为基础的新型程序设计风范,Agent 之间的合作是完成系统计算任务的关键所在. 目前,AOP 还没有形成公认的模式,其中以美国 Stanfrod 大学的 Y. Shoham 教授提出的 AOP 理论模型最有代表性.[3]他将 Agent 定义为包含了诸如信念(Belief)、承诺(Commitment)、能力(Capability)和决定(Decision)等精神状态(Mental State)的实体,并利用基于显式时间点的逻辑(Explicit Time Point-Based Logic)语言对精神状态进行了形式化定义. 他认为 AOP 理论框架应当包含 3 个基本元素:Agent 形式化定义;Agent 编程语言;Agent 形成器(用于将中性元素转化为可编程的Agent). 而从工程的观点来看,AOP 则是面向对象程序设计(OOP)的特例,一个计算是由相互之间进行通知、请求、提供、接受、拒绝、竞争和帮助等言语行为(Speech Act)[4]的 Agent 组成. Shoham 对 AOP 框架的前两个元素进行了深入讨论,提出了一种 AOP 语言(AGENT0)及其解释器. 框架的第 3 个元素尚未深入研究.

有关 AOP 的研究还有许多,比如一些学者从协作分布式问题求解(CDPS)应用出发,提出了相应的 Agent 模型/语言以及 Agent 关系模型,为 CDPS 系统的设计与开发提供了支持,这方面的工作主要是针对 CDPS 系统中的软件 Agent 的构造与协作来进行,很难推广到同时包含了人类 Agent 与软件 Agent 的应用环境中,比如象计算机支持的协同工作(CSCW)、软件过程(Software Process)等研究领域就同时包含了人类 Agent 与软件Agent,为此已经有人提出了人机协同工作 HCCW(human computer cooperative work)的概念[5],但尚未形成统一认识,有待于进一步的研究工作. 另外,现有成果在帮助深层理解系统结构特点方面没有提供足够支持.

可以看出,AOP 技术方法的应用前景是乐观的. 为了应用 AOP 技术解决合作 Agent 应用问题,我们在 Shoham 工作的基础上加以改进与扩展,提出如下 AOP 方法框架:Agent 形式化定义;Agent 编程语言;Agent 形成器(用于将中性元素转化为可编程的Agent);Agent关系模型. 在 Agent 形式化定义方面采用了 Shoham 的成果,对 Agent 采用精神元素描述,使我们能够从高层次来考察其行为,这一点对于那些我们尚未深入了解的系统,如人、机器人以及某些软件 Agent 非常有用. 但是,目前还无法自动实现中性元素到 Agent 的转换,即框架的第 3 个元素——Agent 形成器目前还没有做进一步的研究工作,只能根据具体情况手工提取软件元素的精神状态.

1.1 AOP 语言 AOPL 及其支持系统 AOPS

为了定义和编程 Agent,我们提出了一种 AOP 语言(AOPL),该语言的原型是 AGENT0.[6]其语法的 BNF 表示如下:

```
〈程序体〉        ::=AGENT (〈agent〉,〈组 agent 名表〉)
                          CAPABILITIES:= (〈动作〉〈精神条件〉)*
                          INITIAL BELIEFS:=〈断言〉*
                          COMMITMENT RULES:=〈承诺规则〉*
                      END_AGENT
〈组 agent 名表〉 ::=〈组 agent 名表〉$〈agent〉|〈agent〉
〈承诺规则〉      ::= (COMMIT〈消息条件〉〈精神条件〉(〈agent〉〈动作〉)*)
〈消息条件〉      ::=〈消息条件〉OR〈消息模式〉|〈消息条件〉AND〈消息模式〉|〈消息模式〉|true
```

```
〈消息模式〉    ::=(〈agent〉 INFORM 〈断言〉)|(〈agent〉 REQUEST〈动作〉)|(NOT〈消息模式〉)
〈精神条件〉    ::=〈精神条件〉OR〈精神模式〉|〈精神条件〉AND〈精神模式〉|〈精神模式〉|true
〈精神模式〉    ::=(B〈断言〉)|(〈CMT〈agent〉)〈动作〉)|(NOT〈精神模式〉)
〈动作〉        ::=(DO 〈时间〉〈私有动作〉)|
                   (INFORM 〈时间〉〈agent〉〈断言〉)|
                   (REQUEST 〈时间〉〈agent〉〈动作〉)|
                   (UNREQUEST 〈时间〉〈agent〉〈动作〉)|
                   (REFRAIN 〈动作〉)|(IF〈精神条件〉〈动作〉)
〈断言〉        ::=(〈时间〉(〈谓词〉〈参数〉*))|〈变量〉
〈时间〉        ::=〈integer〉|now|〈变量〉|(〈时间〉+〈integer〉〈时间常数〉)|(〈时间〉-〈integer〉〈时间常数〉)
〈时间常数〉    ::= m|h|d|w|y      ;m(minute)=60,h(hour)=3600,etc
〈agent〉       ::=〈alphanumeric_string〉 | 〈变量〉
〈谓词〉        ::=〈alphanumeric_string〉
〈参数〉        ::=〈alphanumeric_string〉 | 〈变量〉
〈变量〉        ::=? 〈alphanumeric_string〉
```

与AGENT0不同的是,AOPL是一种编译型语言,并且在AOPL中引入了多重继承机制,单个Agent可以继承其所属于的"组Agent"的精神状态(特指信念和能力),在语义方面则基本与AGENT0相同.每个Agent由其自身的AOPL程序所控制,程序体就是一些承诺规则(用以决定是否作出各种承诺)以及该Agent的初始信念和能力.每个Agent总是不断地重复两类动作:作出某个有关将来的承诺和履行先前定义的执行时间已到的承诺,Agent只能对那些它能够直接执行的原始动作作出承诺.动作可以是私有的或通信的,而且,动作也可以是有条件的或无条件的.私有动作相当于OO语言中对象内部的方法,它是实现Agent特定功能的关键所在.通信动作包括INFORM,REQUEST,UNREQUEST和REFRAIN等言语行为;有条件的动作依赖于所谓精神条件(即精神状态),而承诺的作出则同时依赖于精神条件和消息条件(当前的输入消息).AOPL程序的解释从原理上来讲很简单,它由以下两个重复执行的步骤组成:(1)读取当前消息,修改自身的精神状态,包括信念和承诺(AOPL程序是这种修改的关键);(2)执行当前时刻的各项承诺,可能导致进一步的信念修改(本阶段独立于AOPL程序).

作为一种编译型语言,AOPL语言的执行效率高于AGENT0,同时适用范围更广,这来自于我们对AOPL采取的实现策略.在具体实现AOPL时,我们采用了预编译策略,即将AOPL转换为C语言,这样AOPL本身只需提供构造Agent的基本元素以及Agent通信原语,而一些特定的任务则可以利用C语言来完成(即私有动作用C实现),从而能够借助C语言的许多重要设施来弥补AOPL本身的不足.从某种意义上讲,AOPL可以看作是一种外壳语言,它可以用来对由C语言书写的软件元素加以包装,使得这些软件元素成为所谓Agent,从而可以进行更为高级的通信和合作行为.为了支持AOPL程序设计任务,我们开发了其原型支持系统AOPS,其系统结构见图1.[7]

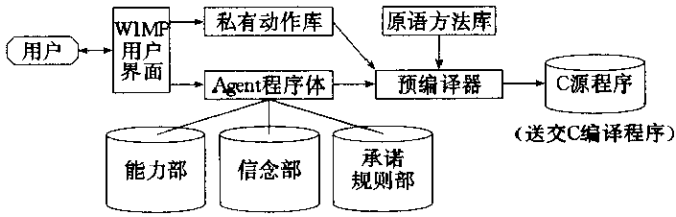

图1 AOPS系统结构

WIMP(window icon menu point & click)用户界面为用户提供了一个友好的编程环境,支持"填充"(Fill-in)式的程序设计,程序员可以分别完成 Agent 程序体的能力部、信念部和承诺规则部定义,方便了程序设计任务.

Agent 程序体是由 AOPL 书写的文本文件,它必须经过预编译后才能得到我们所定义的 Agent 本身(含各种精神元素).需要说明的是,Agent 的能力(即私有动作)是固定的,决定则被简单地看作是 Agent 对自己的承诺.

私有动作库保存了 Agent 的私有动作(用 C 语言实现),是 Agent 实现其自身功能的基础,目前的 AOPS 系统要求私有动作实现为不含参数的 C 函数.

原语方法库保存了一些扩展 C 库函数,包括针对 Agent 精神状态的各种操作以及用于 Agent 通信的高层通信原语如 INFORM,REQUEST 等,供私有动作使用.

预编译器是 AOPS 系统的核心机构,它将 Agent 程序体、私有动作库和原语方法库经过转换连接最终生成模拟 Agent 本身行为的 C 语言程序,在预编译时要给高度抽象的 AOPL程序加入操作性语义以模拟 Agent 的两步行为,并利用 UNIX 的 socket 机制来实现 AOPL 的通信原语.

1.2 Agent 关系模型

在 CDPS 领域已经对 Agent 之间可能形成的各种关系进行了深入研究,并探讨了如何通过 Agent 协作关系来表现 CDPS 系统结构以及利用这些协作关系来实现系统总体目标[8],这些工作关注的是"如何做"这个问题,然而当我们改进或重新设计由多个合作 Agent 所组成的系统时,常常还需要对系统有更深层的理解,现有成果没有为此提供足够支持.为此,我们提出了一种 Agent 关系模型,该模型可用以体现合作 Agent 系统的体系结构,指导 Agent 之间的协作行为,同时还为深层理解系统结构特点提供了有效支持.Agent 关系模型包含了 Agent 依赖性(Dependency)和 Agent 关系图两方面内容.

1.2.1 Agent 依赖性

在合作 Agent 之间存在着各种依赖关系,我们认为可以参考人与人在合作过程中所形成的各种依赖关系来进行描述.这方面已经有相关成果,比如 E. S. Yu 等人从软件过程模型角度对软件开发过程中所涉及到的各类人员之间的依赖关系进行了分类.[9]在此我们借鉴了他们的研究成果,并进行了相应修改以适应合作 Agent 应用的特点,将 Agent 依赖性区分为承诺依赖和资源依赖两类.其中承诺依赖指一个 Agent 依赖于另一个 Agent 作出某项承诺(如达到某个目标,或执行某个动作);资源依赖指一个 Agent 依赖于另一个Agent以得到某个资源(物理的或信息的)的可用性.我们称具有依赖关系的两个 Agent 分别为依赖者(Depender)和被依赖者(Dependee),称二者之间所形成的依赖关系为依赖体(Dependum),依赖者通过对于被依赖者的依赖而达到某些自己无法实现的目标,但是也由于这些依赖而具有脆弱性(Vulnerable),即被依赖者的失败将对依赖者产生副作用.同时,我们将依赖性区分为两种强弱程度:

(1) 开放(Open)依赖性——依赖者希望依赖体得到保证(如某个目标被实现、某个动作被执行,或某个资源可用),从而能够被自己用于某个动作的执行中去,但是依赖体的失效不会对依赖者产生重大影响.对被依赖者而言,开放依赖性是被依赖者关于该依赖体能够得到保证的一种宣言.

(2) 关键(Critical)依赖性——这种依赖体的失效将对依赖者自身目标的实现造成严重影响,依赖者所有与该依赖体相关的动作过程都要失败.在这种情况下,依赖者不仅要关心该直接依赖体的有效性,同时还要关心被依赖者自身与其它 Agent 之间所形成的依赖体的有效性以及更深层次的依赖关系.对被依赖者而言,关键依赖性意味着被依赖者必须尽最大努力来保证该依赖体的有效性,比如它要确保自身已经形成的依赖性是可靠的.

1.2.2 Agent 关系图

根据 Agent 之间的依赖性可以得到如下 4 类 Agent 合作类型:

(1)水平型合作——每个 Agent 可以独立取得问题答案,而不必依赖于其它 Agent,但如果它们之间相互合作,则可以提高答案的可信度,Agent 之间存在着开放依赖性.

(2)树型合作——一个高级 Agent 必须依靠其它的低级 Agent 才能取得其答案,上下级 Agent 之间存在着关键依赖性.

(3)递归型合作——为了取得问题的答案,在各个 Agent 之间形成了依赖关系环,Agent 之间存在着关键依赖性.

(4)混合型合作——它是前 3 种合作类型的有机结合.

针对合作 Agent 应用特点可以利用由树型合作和水平型合作结合成的混合型合作模式来体现系统的体系结构,即总体上采用树型合作模式而相同层次上采用水平型合作模式,可以形象化地用图 2 表示,我们称之为 Agent 关系图 ARD(agent relation diagram).

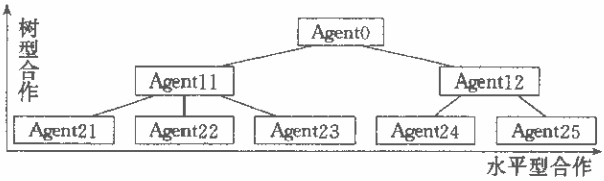

图2 Agent关系图

ARD 提供了一个有关系统组成结构的概念简图,其中方框表示系统中的 Agent,在纵轴方向上下层 Agent 具有树型合作关系(方框之间的连线表示 Agent 之间的关键依赖性),水平轴方向 Agent 具有水平型合作关系(出于简洁性考虑,未画出 Agent 之间的开放依赖性).同时,ARD 对于 Agent 间的协作行为提供了指导,上级 Agent 根据关键依赖性可以要求下级 Agent 作出某项承诺或提交某个资源以供其利用,下级 Agent 则应尽最大努力完成要求,由此可能进一步形成一个关键依赖链.这种树型合作关系是由系统为了实现整体目标强加给各个 Agent 的,Agent 自身无权改变.同级 Agent 之间则可动态建立合作关系,从而达到优化组合的目标.从原理上讲,同级 Agent 之间还可以形成递归合作关系,目前我们尚未对之进行深入研究.

值得注意的是,通过对 Agent 之间依赖性的分析,可以了解到各个 Agent 在系统中处于何种地位,其脆弱性如何,其功能和性能的好坏对系统有何影响等等,这些信息会帮助我们对系统结构特点有更深层的理解,有利于系统的改进或重新设计.另外,对 Agent 依赖性的具体定义影响到 Agent 的功能设计,比如某个 Agent 与另一个 Agent 存在着承诺依赖性,它可能只关心由被依赖者执行承诺的最终结果而不管其具体实施过程,也可能指定具体实施步骤,这将影响 Agent 之间的功能分配.

为了避免模糊性并且使得 Agent 关系模型中所表达的知识能够由计算机处理,目前我们正在对之进行形式化描述工作.

2 应用举例

本节介绍 AOP 技术在多功能感知系统开发中的应用.[10]

多功能感知系统是一个面向人类语言(自然语言和人体语言)感知的软硬件平台,人类语言结构的复杂性导致对于人类语言的感知是一个多通道、多信息的综合决策过程,各通道的信息经过相应处理后,所得到的信息还要融合加工成一个信息或命令序列.从信息处理角度来看,多功能感知系统的功能元素(即 Agent)应包含语音识别、文字识别、表情识别、手势识别、口形识别、自然语言理解、人体语言理解和融合器等 Agent,各 Agent 既独立工作又要与其它 Agent 协作才能最终实现整个系统功能,为此可利用 AOP 技术来进行系统的设计与开发工作.根据 Agent 之间的依赖关系可以得到多功能感知系统的 ARD 图(如图 3 所示,同时考虑了硬件和算法基础),该图表现了系统的组成结构,为 Agent 之间的协作提供了指导,并且通过对 ARD 的分析还可以对系统结构特点有深层次的理解.比如融合器 Agent(FA)的工作关键依赖(同时具有承诺依赖和资源依赖关系)于自然语言理解器 Agent (NA)和人体语言理解器 Agent(BA),FA 对于 NA 和 BA 的承诺依赖表现在 FA 需要 NA 和 BA 分别完成自然语言和人体语言的理解任务,资源依赖则表现为 FA 需要由 NA 和 BA 发送来的各自的理解结果以供融合处理,如何合理分配各自的功能职责取决于对它们之间依赖性的具体定义.目前该图仅作为概念框架提供指导作用,有关详细的依赖关系定义与各 Agent 的具体功能设计密切相关,涉及到人类语言感知融合模型的研究,是我们进一步的研究目标.每一特定 Agent 可利用 C 语言开发其底层功能(即私有动作),并利用 AOPL 对之加以包装(需要人工提取 Agent 的精神状态,目前尚无统一方法),使得 Agent 之间的通信与协作可借助 AOPL 的高级通信原语得以自然实施.

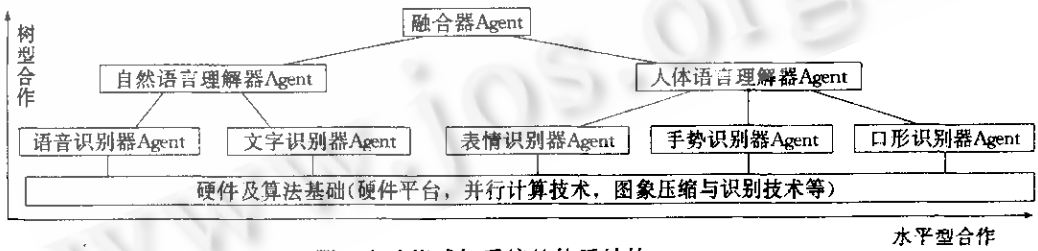

图3 多功能感知系统的体系结构

为了体现 AOPL 编程特点,在此我们给出手势识别器 Agent 的一种简单实现以供参考.我们把手势识别器 Agent 的识别功能抽象为其能力,将已识别的手势(可表现为手势词汇名及别名)序列抽象为 Agent 的信念,以便需要时向其它 Agent 通告(例如提供给人体语言理解器 Agent 以使之进行综合理解),初始信念库为空,随着识别过程的进行而得到填充,并可以通过人工增删.能力库仅含一个条目(recognize_hand_guesture true).信念条目形如(time (hand_guesture_type alias_name)).

手势识别器 Agent 的 AOPL 程序如下:

```
AGENT(hand_guesture_recognizer)
```

```
CAPABILITIES          :=(略)
INITIAL BELIEFS       :=(略)
COMMITMENT RULES      :=0
  (COMMIT (? agt REQUEST (IF (B ? p) (INFORM ? t ? agt ? p)))
        true
        (? pass
        (IF (B ? p) (INFORM ? t ? pass ? p))))
  (COMMIT true
        (NOT ((CMT ? anyone) (INFORM ? time ? anyone ? anything)))
        (myself (DO ? time recognize_hand_guesture)))
END_AGENT
```

解释:第1个承诺规则用来产生回答相关Agent提问的承诺,另一个则用于产生执行私有动作recognize_hand_guesture(完成手势识别任务,应由C语言实现)的承诺.

3 结 论

本文从合作Agent应用出发提出了一种面向Agent程序设计技术的方法框架,重点论述了该框架的两个重要组成部分——AOP语言(AOPL)和Agent关系模型.编译型的AOPL语言在执行效率和应用范围上有一定的优越性,Agent关系模型则为更深层的理解系统结构特点提供了支持,这些使得应用AOP技术解决合作Agent应用问题成为可能.

本文所提出的AOP方法框架还存在许多有待完善之处,比如缺乏提取Agent精神元素的统一方法(理想情况是由Agent形成器自动完成),AOPL语言和Agent关系模型不完善等;另外,为了在诸如CSCW,HCCW或软件过程等同时存在人类Agent和软件Agent的领域中应用AOP技术,需要解决更多的技术问题,其实用性还有待于在实践中进一步加以验证与改进.

我们进一步的工作主要集中在以下三方面:(1)完善AOPL及其支持系统AOPS;(2)Agent关系模型的形式化定义;(3)应用AOP技术设计并实现一个软件过程管理工具.

参考文献

1 Potts C, Bruns G. Recording the reasons for design decisions. In: Tan Chin Nam ed. Proceedings of 10th IEEE International Conference on Software Engineering, Singapore, California: IEEE Computer Society Press, 1988. 418～427.

2 姚郑,高文. 面向Agent的程序设计风范. 计算机科学,1995,**22(6)**:7～11.

3 Shoham Y. Agent oriented programming. Artificial Intelligence, 1993,**60(1)**:51～92.

4 Searle J R. Speech acts: an essay in the philosophy of language. New York: Cambridge University Press, 1969.

5 Lux A, Greef P *et al.* A generic framework for human computer cooperation. In: Huhns M, Papazoglou M P, Schlageter G eds. Proceedings of International Conference on Intelligent and Cooperative Information Systems, Rotterdam, The Netherland, California: IEEE Computer Society Press, 1993. 89～97.

6 Torrance M, Viola P. The AGENT0 Manual. Technical Report STAN-CS-91-1389, Department of Computer Science, Stanford University, Stanford, CA, USA, 1991.

7 Yao Z, Gao W. AOPS: an agent oriented programming system. In: Lin Z, Barthes J P eds. Proceedings of International workshop on CSCW in Design, Bejing: International Academic Publishers, 1996. 39～43.

8 马志方,刘大有,吕为公. 面向Agent的程序设计语言:DL-1. 计算机研究与发展,1996,**33(2)**:93～101.

9 Yu E S, Mylopoulos J. Understanding "Why" in software process modelling, analysis, and design. In: Kavanaugh

M E ed. Proceedings of 16th IEEE International Conference on Software Engineering, Sorrento, Italy, California: IEEE Computer Society Press, 1994. 159～168.

10 姚郑,高文. 多功能感知系统中的面向 Agent 技术. 软件学报,1996,7(3):163～167.

AGENT ORIENTED PROGRAMMING

YAO Zheng　GAO Wen

(*Department of Computer Science　Harbin Institute of Technology　Harbin* 150001)

Abstract　In this paper, an AOP(agent oriented programming) framework is described from the view of the so-called cooperative agent application problem. In the framework, a new agent language (AOPL) is presented, the design and implementation of its programming system called AOPS are discussed. AOPS can translate AOPL to C. Moreover, a new agent relation model is presented. Detailed discussion is the composition of the model and the importance on embodying the architecture of the cooperative agent application system, guiding the cooperative behavior among agents, and supporting the in-depth apprehension about the characteristics of the system architecture. Finally, the application of AOP in the multiple functional perception system which is now being implemented is discussed.

Key words　Agent, agent oriented programming, agent oriented programming language, agent relation model.

Class number　TP312