ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.20, Supplement, December 2009, pp.66−75

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

一种MAS构件知识复用动态演化模型*

蒋伟进[1,2+], 夏 可[1]

[1](湖南商学院 计算机与电子工程学院,湖南 长沙 410205)

[2](湘潭大学 信息工程学院,湖南 湘潭 411000)

Research on Knowledge Reuse Dynamic Evolvement Model Based on Multi-Agent System and Component

JIANG Wei-Jin[1,2+], XIA Ke[1]

[1](School of Computer, Hu'nan University of Commerce, Changsha 410205, China)

[2](School of Information Engineering, Xiangtan University, Xiangtan 411000, China)

+ Corresponding author: ljxwhj@163.com

Jiang WJ, Xia K. Research on knowledge reuse dynamic evolvement model based on multi-agent system and component. *Journal of Software*, 2009,20(Suppl.):66−75. http://www.jos.org.cn/1000-9825/09009.htm

Abstract: Aiming at quick and proper reuse of existing knowledge in enterprises, a new idea is advanced that separated the business logic of knowledge management from the transactions of knowledge process. Furthermore, an agile knowledge reuse model based on multi-agent and knowledge service is constructed and agile knowledge reuse system is defined. To realize the agile transference of knowledge service, a rule-coordination mode based on multi-agent is established through study on the rule model of business logic and the activity-action model of the software Agent. In this way, the dynamic reuse of knowledge and the dynamic regrouping of knowledge using process reuse can be supported effectively, the process-distribution capability and extensible function of knowledge management system can also be increased. The agents can perform the retrieval requests from users via collaboration in a distributed component repository system. The agents have their own knowledge base. They need the capability to learn new information and update their knowledge base to keep the retrieval results effective. Finally, a case study is given to illustrate the application of this model.

Key words: multi-agent system; distributed component repository; knowledge reuse; knowledge management; knowledge service rule

摘 要: 为提高企业的知识利用效率,增强企业创新能力,针对企业现有知识和系统,提出将企业知识管理的业务逻辑与知识处理事务分开,建立了基于多智能体和构件知识的知识复用模型,设计了知识管理业务逻辑的规则模型和智能体的活动行为模型,讨论了基于多智能体的规则协调模式,有效地支持知识的动态复用和知识使用过程的动态重组,增强知识管理系统的分布式处理能力和规模可扩展能力.在分布式构件库系统中,智能体通过协作联合完成任务要求.每个智能体拥有自己的知识库,并且具备学习能力,能够更新其知识库以保持执行结果的有效性.

关键词: 多智能体;分布式构件库;知识复用;知识管理;知识服务

* Supported by the Natural Science Foundation of Hu'nan Province of China under Grant No.06JJ2033 (湖南省自然科学基金); the Society Science Foundation of Hu'nan Province of China under Grant No.07YBB239 (湖南省社会科学基金)

Received 2008-09-20; Accepted 2009-04-09

智能体(agent)近年来已在越来越多的领域中得到应用和研究[1−5],智能体技术对于目前软件工程界关注的构件库也非常重要.随着软件复用和构件技术的发展,作为对大量构件进行有效管理的基础设施的构件库相关技术也逐步成熟,许多科研机构和软件企业也相应地建立了一系列具有实用价值的构件库[6,7].由于构件库在企业中的普及,导致了在企业间共享构件信息的需求;而企业的地理位置分散性和各自分类描述机制的多样性使整个体系朝分布式异构构件库方向发展.借助智能体技术,能够通过一种协作机制将分布的、异构的构件库整合起来,对外提供一个统一的逻辑视图和概念视图.

知识管理已经在国内外学术界和企业界引起了广泛的关注[8,9].实施知识管理能够提高企业对知识的使用效率和知识创新能力,这就要求企业必须重视知识的复用问题.知识复用是指组织利用其已有的知识去完成某一知识型任务的过程[8].知识敏捷和高效复用已经成为现代企业成功的关键因素,知识管理要求敏捷化的软件系统支持,以满足企业快速创新知识和高效利用知识的需要.知识复用区别于一般知识复用的主要特点是要求知识管理系统能够根据知识需求变化,动态地形成和解体,并进行快速地重构和调整,系统具有可复用性和重构性.目前,关于知识复用方面的研究已经引起了国内外学者的重视,文献[9]归纳出了一种成功的知识复用理论;文献[10]研究了知识处理中适用于群体用户调用的知识复用方法;文献[11]研究了放射形功能网络中的知识增长和复用技术;文献[12]研究了用动态知识地图实现专家隐性知识的复用;文献[13]给出了一个知识系统复用框架;文献[14]认为工业设计中有 90%是基于变种的设计,并且有 70%是复用过去设计方案中的知识;文献[15]研究了研发过程中的知识复用问题;文献[16]研究了知识创新中的知识复用过程.这些研究成果都具有各自的特点,但很少有从如何构造智能体系统(multi agent system,简称MAS)去研究知识复用问题.我们曾在文献[17,18]中开展了多智能体系统的理论和应用研究,并认为将智能体作为一种构件亦可应用于知识复用的研究.本文在上述工作基础上将软件复用思想引入到知识管理中,进一步提出将知识管理的业务逻辑与知识处理事务分开的思想,利用规则推理模型实现企业知识管理业务的变化,运用多智能体系统和知识服务,完成具体的知识管理任务,实现知识的动态复用和应用过程的动态重组,从而增强知识管理系统的分布式处理能力和规模可扩展能力,降低知识管理系统的构造成本,提高系统的稳定性和健壮性.

1 基于 MAS 的知识复用模型

现存构建分布式应用的方法可分为两类:一种是基于传统“软总线”(如 DCOM,CORBA)的分布式环境,而另外一种则是用 Agent 构造分布式环境,以“软总线”作为 Agent 的通信介质.由于 Agent 具有自主性和自适应性,Agent 能够感知外界消息的变化,并根据其知识做出相应反应,因此,采用后者所构建的分布式平台对外界消息变化的适应能力显然更强.考虑前文提出的推理构件的协同问题求解问题,推理构件在推理受阻后寻找协同者时,问题解决方法(problem solving method,简称 PSM)工作方式将发生变化,即由原先的查询方式(query pattern)转变为协同方式(cooperation patten),PSM 向 CORBA 发送的消息也将随之而变.如果采用传统方法构建分布式应用,系统将跟踪不到PSM的消息变化,而后者则可感知此变化,因此,本文采用Agent构造面向推理构件的分布式计算环境.

知识管理的目的是在整个管理过程中最大限度地实现知识共享[19],以便利用这些知识做出科学的决策[20].快速和恰当地获取知识处理事务,共享和应用与之关联的知识是知识复用的核心.

定义 1. 定义知识复用模型 *KR* 为 $KR=\{A_{set},R_{set},S_{set}\}$.

其中,$A_{set}=\{a_1\times a_2\times...\times a_n\}$,$n\in Z$ 表示由多个自治的分布的 Agent 通过通信、协商和合作组成的 Multi-Agent 系统,a_n 表示能够按照一定的规则完成具体知识处理事务自治的 Agent.Agent 之间利用通信原语相互传递知识和需求,按照协商协议进行合作,达到共同的目标.如果将 Agent 定义成企业知识管理过程中最基本的知识处理事务的映射体,那么只要这些最基本的知识处理事务不变,此 Agent 就可以复用.即使某个知识处理事务需要改变,也只要改变相应的单个 Agent,其他 Agent 则不需要改变,从而使系统得到最大限度的复用.

$R_{set}=\{r_i^2|i=1,2,...,n\}$,表示由多个知识处理规则组成的集合.知识的使用是依据一定的知识处理逻辑来进行

的,知识处理逻辑用规则r来表示.知识处理逻辑与具体的知识处理事务分开,可提高知识管理系统的适应性和敏捷性[21].规则集 R_{set} 控制知识管理系统中知识处理的行为.由于Agent的活动是根据规则来执行的,改变规则 R_{set} 就改变了Agent的活动行为,从而改变了整个系统的执行结果.因此,当知识管理系统中的知识处理逻辑发生变化时,只要通过修改规则集,即可使原有系统适应这种变化,使知识的可复用性大大提高.

$S_{set}=\{Domaink, object, property, relations, functions\}$,表示整个知识管理环节中可以共享的知识服务(knowledge services)的集合,知识服务是知识管理活动中能够实现具体知识处理功能的知识处理事务及其相关知识的封装体.知识处理功能是指企业利用知识解决问题的能力,相关知识包括编码化的显性知识和用来对隐性知识进行描述的元知识.影响知识服务定义的因素很多,这里把能否满足企业在知识管理中的知识处理功能,以及多个Agent 之间对知识进行共同的理解作为知识服务定义的要素,具体由以下两部分组成:

(l) *Domain K* 表示领域知识.领域知识是一种具有特定功能的特殊资源,领域知识不仅包括已经显性化了的编码知识,也包括那些隐性化的专家知识和过程知识等.显然,后者是难以封装到知识服务中的,这里将通过封装那些描述隐性化知识存在状态、特征、功能和交互方式等的元知识,达到复用隐性化知识的目的.

(2) *object* 表示知识服务中的对象.它是参与知识处理活动的实体,每个领域知识中包含多个知识对象.

(3) *property* 表示知识对象的属性.它反映了知识对象的活动行为和活动状态及其他一些特征.

(4) *relations* 表示知识对象之间以及对象属性之间的关系,这两种关系是通过关系的表示符来区分的.

(5) *functions* 表示知识处理事务所能实现的功能.

Agent通常都具有 3 个属性,即自治性、适应性和分布性.多Agent系统能够给应用系统带来更大的开放性和可复用性[22].当分解知识管理中的知识处理功能并封装为多个基本的知识服务时,每个基本知识服务映射为一个活动,多个活动的组合完成一个知识处理任务.SA定义为能够完成多个活动的软件实体,当这些活动功能不变时,相应的SA就可以复用.

定义 2. SA(software agent)是由一些软件组件模块集合构成的.Agent 通过这些组件来感知外部环境和自己的内部状态,以决定执行什么样的知识处理活动和将来的状态.

SA=(*Actions*,*KB*,*Capabilities*,*Tasks*).其中,*Actions* 是由知识管理用户定制的 SA 能够执行的一组活动集.Agent 根据自己当前的状态,从活动集中选择相应的活动执行,实现一定的目标,其行为和功能都是通过 Agent 的活动来实现的.

KB 是 Agent 所拥有的知识集,包括内部知识集(实现本身的功能所需要的最基本数据和知识)和共享知识集(和其他 Agent 进行合作时所具有的某一领域的共享知识).

Capabilties 是智能体的问题求解能力的说明,包括能力、操作和条件 3 个部分:① 能力将 SA 是否能够完成某个知识管理任务和 SA 的求解问题的推理过程分开,实现敏捷化;② 操作定义了 SA 中的推理过程,描述了 SA 中的能力是如何被实现的,定义了完成这样能力的推理活动及活动之间的相互作用,操作利用了动态逻辑表达其过程控制;③ 条件是描述 Agent 为完成某一知识处理活动或任务时,所需要的外部或内部条件,也是规则系统进行推理时所需要的基本内容.

Tasks 对需要解决的问题进行任务描述,表示希望 SA 解决与知识应用的问题,*Capabilities* 在描述解决问题的步骤时对领域知识有需要,而对 *Tasks* 的描述不需要领域知识.

SA 通过执行一系列知识处理活动来实现其功能,Agent 模型中的知识处理活动由活动基集、活动准则、活动需求和活动的执行结果 4 个方面组成.活动基集是由一些与基本知识服务对应的原子活动组成的集合,每个原子活动是由活动的名字、活动的先决条件、活动的执行和活动的状态等构成,Agent 执行活动是根据知识处理活动准则来执行的.

假定α是一个原子活动,定义$P_s(\alpha),R_s(\alpha),D_s(\alpha),E_s(\alpha),W_s(\alpha)$是原子活动的原子状态,分别表示$\alpha$是可执行、应执行、正在执行、结束执行和放弃执行的知识处理活动,集合 $S=\{P_s,R_s,D_s,E_s,W_s\}$ 是知识处理活动的状态集.

对任何操作 $OP\in\{P_s,R_s,D_s,E_s,W_s\}$,有 $OP(s)=\{\alpha|Op(a)\in s\}$.状态集是任何基本知识处理活动的状态集合,状态 S 表达了基本知识处理活动的状态信息.如果操作 $Op(\alpha)$在状态集合 S 下发生,则对活动α来说,操作 Op 的

状态为真.例如, $D_s(\alpha)$ $W_s(\beta)\in s$ 意味着将执行活动α,而放弃活动β的执行.当然,并不是任何状态集都是有意义的,如若 D_s 和 E_s,同时存在于S中,S就是不一致的,因为活动α并不能同时正在执行又结束执行.为了表达出有意义的状态集,就需要定义活动一致性规则,即如果活动状态集 S 对任何原子活动α满足下面的规则,则称此状态集满足一致性.

Rule 1. $\forall_s\exists\alpha R_s(\alpha)\in S\rightarrow\neg\exists\alpha\forall_s W_s(\alpha)\in S$.

Rule 2. $\forall_s\exists\alpha D_s(\alpha)\in S\rightarrow\neg\exists\alpha\forall_s(W_s(\alpha)\in S\vee(E,(\alpha)\in S))$.

Rule 3. $\forall_s\exists\alpha P_s(\alpha)\in S\rightarrow\neg\exists\alpha R_s(\alpha)\big|=P_s(\alpha)$.

在整个知识管理系统中,分布式的知识处理是通过多个 SA 协作来完成的.在 MAS 中,Agent 的活动除了改变自身的状态、调节自己的行为以外,也可能改变它生存的周围环境.周围环境包括系统中的服务、通信、功能等 Agent 以及它们共享的知识库内容等,这些环境都处于某一具体的状态中,而状态是通过各种对象的属性及它们之间的关系来描绘的.状态的改变是在一定的条件下,由于事件的触发和活动的执行引起的.知识复用的规则模型就是依据对状态变化的描述来控制 Agent 与其周围环境的作用,即与其他 Agent 之间的相互作用.

知识管理系统中,不同企业知识管理的业务逻辑是有差别的,可以通过构造业务逻辑表达模型来实现不同企业业务逻辑中的元知识和操作[23].这里通过规则模型来实现知识管理的业务逻辑的表达、重构和复用.在知识管理的生命周期中,规则模型控制着MAS的行为路径和活动目标.

在知识复用的规则模型中,规则通常表现为 *LHS*→*RHS*(即前件→后件),即由左边的条件、事实或状态可以引起右边的行为、活动或目标.当外界状态满足此规则左边(*LHS*)的条件时,此规则将引起右边(*RHS*)的活动执行.

定义 3. 对任意一条规则 *r* 可用三元组表示为 $\Gamma\langle Cons,Operation,Goal\rangle$.其中,*Cons* 表示规则推理时所需要的知识资源或条件,知识管理中任意对象 *Object* 及其属性、状态和所属的领域 *Domain K* 的定义都属于 *Cons*;*Operation* 表示当 *Cons* 满足时,规则所要执行的操作或规则表示的事实与 *Goal* 的关系,规则的行为分为:① 当 *Cons* 满足时,执行一系列操作,达到一定目标 *Goal*;② 表达 *Cons* 与对象、事实或状态之间的关系.*Goal* 表示规则的目标,即希望得到的结果,可以是一个事实状态,也可以是一个需要执行的活动名称.

Cons,*Operation* 和 *Goal* 都是由规则中的基本事实单元 *BT* 组成,描述为

$$BT=\{ID,Domain,Type,Object,Property,Relation,State,Task,Action\}.$$

对任意一条规则都有惟一标志其身份的 ID;*Domain* 表明规则表达的事实或条件所适应的领域,而这些领域中事实或条件由类型(type)、对象(object)、属性(property)、关系(relation)及状态(state)等表达.另外,当规则的条件满足时,将会引起任务(task)执行,任务的完成是通过一系列活动(action)的执行实现的.当多个 SA 利用规则进行合作时,它们利用规则所表达的事实进行推理,即根据规则之间的关系,找到一条达到目标的路径.

在知识复用系统中,由于知识的分布性和知识处理事务的异步性,许多活动的执行与其发生的时间顺序直接相关联.

定义 4. 若规则 *r* 上的状态 *State* 或活动 *Action* 受时间 *t*,则有以下 6 种约束关系:

Before of(≥),after of(≤),not before of(>),not after of(<),betweennof(⊕),endof(),分别表示“先于…”、“后于…”、“不先于…”、“不后于…”、“在…之间”和“结束于…”.

将知识管理的业务逻辑利用规则模型表示以后,规则模型利用基于 *r* 规则的正向演绎推理控制着多个 Agent 之间的行为,来实现对知识管理的功能.系统不用修改任何代码,而通过修改规则模型来适应外界环境的变化,从而使得知识服务得以复用.

2 基于规则的多智能体动态协作

当将企业的知识管理业务逻辑利用规则模型表达成推理机可以查询和推理的规则格式后,就可以通过规则模型调度 Agent 执行企业的知识管理活动,实现知识复用.这种规则模型与 Agent 模型的结合,形成了一种基于规则的 MAS 协调模式,其结构为:

WHEN(Event)

IF(Condition) THEN (Action$_1$) ELSE(Action$_2$) OR(NULL)

当事件(event)发生时,如果满足条件(condition),则执行活动(action);否则,执行活动或不执行任何活动.其中,Event 和 Condition 都是可以通过关系连接符 AND 或 OR 关联起来,能够表达更加复杂的事件和条件.

设有 Agent:A_1,A_2,A_3,A_4,A_5,A_6,它们可执行的活动分别为 $A_1(a_1,a_2)$,$A_2(a_2,a_5)$,$A_3(a_3,a_{12})$,$A_4(a_2,a_5)$,$A_5(a_{10})$,$A_6(a_7,a_9,a_{11})$,其中a_x表示活动action.(x=1,2,…,11).现将企业的知识管理业务逻辑表达为一组规则:

WHEN Done(a_1) IF Airfoil. Type='fixed' THEN Do (a_3); Do (a_5);

WHEN Done(a_5) IF Plane,Number>100 THEN Do(a_6);

WHEN Done(a_3)AND Done(a_6) IF Customer. Account=Valid THEN Do(a_4); ELSE Do(a_7);

WHEN Done(a_4) IF Airfoil.Length<=10 THEN Do(a_8);

WHEN Done(a_7) IF Plane. Mass>3000 THEN Do(a_2);

WHEN Done(a_2) OR Done(a_8) IF Airfoil. Revolution 'Fastest' THEN Do (a_9); Do (a_{10});

WHEN Done(a_{10}) IF P1ane. Material='Alloy'THEN Do(a_{11}); ELSE Do(a_{12});

其中,Done(a_x)表示完成action$_x$的事件,Do(a_x)表示执行活动action$_x$.

可以看出,当将规则修改后,Agent 之间的调度就可以改变,不用修改程序的任何代码,系统具有很高的可复用性.同时,当外部环境发生改变时,可通过用户界面重新配置系统,以满足新的知识管理需求,因而系统具有可重构性.

3 知识复用在协同设计中的应用

3.1 多Agent设计环境的结构

设计是一个复杂的知识发现过程,分布的信息和知识在这个过程中被同步处理.设计由软件 Agent 和设计者协作完成,软件 Agent 为人类设计者在设计过程中提供了必要的帮助,多 Agent 环境为连接分布的资源及信息提供了良好的框架.

多 Agent 设计环境采用开放式结构,同一设计组的 Agent 和设计师通过局域网、不同组的 Agent 之间通过 Internet 进行通信,交换设计数据和知识.

该环境中的所有 Agent 组成 Agent 群体.Agent 分为管理 Agent、工具 Agent 和设计 Agent 这 3 类.这些 Agent 位于不同层,Agent 在群体中所处的层次决定了它的功能和权限.

(1) 管理 Agent 位于服务器端,主要功能是管理、控制和决策整个设计组.管理 Agent 的知识库中存放设计组中各 Agent 的名字、地址、功能及以往设计中的绩效等历史记录的信息,这些信息有助于管理 Agent 在新的设计过程中选择合适的 Agent.除了知识库以外,管理 Agent 的存储缓冲器中还存放各设计子任务的执行状态和各个 Agent 的工作状态.

(2) 工具 Agent 包括管理工具 Agent 和设计工具 Agent.工具 Agent 帮助管理 Agent 完成系统的管理任务,如任务分解、设计过程监控、通信、共享数据库和知识库管理、协作冲突协调、设计构件组装以及系统维护等.设计工具 Agent 则是支持设计 Agent 完成设计任务的工具软件.

(3) 设计 Agent 是一类面向特定设计领域的 Agent,它们有特定的设计知识和能力,能够在特定的设计领域帮助设计师完成设计任务.

在该环境中,复杂的设计由多个 Agent 协作完成,每个设计 Agent 有自己独立的知识和设计决策方案,能理解设计状态表示,并能协助设计专家完成设计.Agent 的策略依赖于确定的算法,如遗传算法、分类算法等.

3.2 协同设计中的构件知识

构件库中的构件通过 4 种方式产生:

(1) 自动生成.① 设计 Agent 执行基于数学表示二叉树结构的遗传算法,生成 2 维草图,然后对草图进行旋

转或者扫描以形成实体;② 在生成过程中,设计师与设计 Agent 进行交互,给出适应度值,以使进化过程沿着设计师的引导进行.

(2) 已有构件修改.取出生成的构件进行手工修改,以使其具有更好的艺术性,并符合实际设计要求.

(3) 设计师独立设计.使用系统提供的实体生成及实体修改工具,手工绘制图形,生成构件,或者使用设计师习惯使用的设计工具(工具 Agent),如 AutoCAD,Pro-Engineer,Inventor,MicroStation,Solidworks 等.

(4) 识别、重构、修改或进化.通过图形识别器对原有的以图片形式存放的构件进行识别重构,并在现有构件的基础上进行手工修改或者遗传操作以生成新构件.具体步骤为:① 利用草图识别系统 Digiter,对一些成功的设计图片进行扫描后识别,识别后的图像数据以点的形式存放于扩展名为.dat 的文件中;② 设计系统从.dat 文件中对数据进行分离提取,分离后的数据如果能形成数学函数,则作为种子或作为变异的子树引入设计 Agent 生成构件的进化过程;③ 对实体进行重构,然后利用修改工具进行修改.

构件知识查询,查询的目的是根据需求从构件库中找到合适的构件进行组装.构件检索包括构件需求和目标构件库,即需求方的问题空间和构件库的解空间.问题空间由实际问题细化到设计师所理解的论域问题空间,再到查询空间,即从用户需求到库检索系统能够识别的一个确定过程.为了缩短查询时间,提高检索效率和准确性,采用自动检索和手动检索两种方式,自动检索采用模拟退火算法实现,手动检索的过程是由设计师给出检索条件,Agent 计算各个构件和设计师给出的检索条件之间的匹配度,并向设计师显示满足要求的构件.设计师对构件原型实例库进行检索,检索结果有 3 种情况:① 找到与检索条件完全匹配的构件,直接将其取出来,应用于后续设计;② 只搜索到与检索条件部分匹配的构件,将构件按照其匹配值的大小顺序排列出来,设计师通过预览选择较合适的实例,并将其调人造型设计台进行修改完善,直至满足设计要求;③ 搜索不到与检索条件匹配的构件,设计师从头开始设计,通过完全独立的创作设计出全新的构件.

对于第 1 种情况,不存在新构件知识产生的问题,故而不用更新构件知识库;对于后面两种情况,由于对构件实例进行了修改或者创建出了全新的构件知识,需要对构件知识库实施更新操作.

由于 Agent 所处的环境通常具有不确定性,在许多情况下仅仅可访问其中的局部信息.Agent 能够适应环境中的变化,同时与其他 Agent 合作以实现它们的目标.除了具有进行通信的能力,相互协作的 Agent 通常也共享知识.知识表达了一种静态的认知状态,但是,对于学习型 Agent 来说,需要不断根据环境的变化和其他 Agent 的情况来学习新的知识,使知识实现动态演化.

Agent 的知识更新主要来源于两种情况.第 1 种情况是:几个相互协作的 Agent 共事常识库,该知识库被一个知识管理 Agent 动态地更新.另一种情况是 Agent 接收从环境中传来的信息以及从其他 Agent 发来的更新信息,从而更新知识库.在第 2 种情况中,系统内的每个 Agent 需要维护它自己的知识库.

为了动态维护Agent的知识状态,本文采用声明式方法(declarative method)来实现Agent,该Agent能够将新的信息合并到一个给定的知识库体中.此外,针对Agent的异构性[8],我们使用多Agent框架IMPACT[12]来描述可更新Agent的实现.

声明式方法适用于知识库的更新,该方法已经在非单调(nonmonotonic)知识库领域中得到了发展[9−11].使用声明式方法的原因是它对知识库的更新提供了清晰的语义,能够对知识的更新进行推理,并能够处理不一致的信息,对一些非单调推理的形式进行建模.

本文采用的 Agent 架构的基本层次如下;知识库包括事实(fact)和规则(rule),以逻辑程序的形式表达.知识库表示了对一个世界的局部描述,Agent 在该世界中运作,该世界可以被 Agent 共享.Agent 自身可接收以事实的形式或逻辑编程规则(rule)的形式表达的新信息.Agent 拥有一个处于底层的更新框架,该框架描述了在特定的语义下,新接收到的、可能处于不一致状态的信息如何被合并到原有知识库中.Agent 也拥有一些特定的更新策略,这些策略提供了对于合并特定的知识的灵活性.例如,该策略可以描述对于某个特定的信息.知识库中某些特定规则的变化或者撤销(retraction).

例 1:考虑一个示例性的 Agent,它在网络中搜索拥有某种类型构件的构件库 $c_i, 1 \le i \le n$.假设它的知识库 KB 包含规则:

$r_1: query(c_i) \leftarrow possess(c_i), open(c_i), \text{not} \neg query(c_i)$,

$r_j: try\_query(c_i) \leftarrow query(c_i)$,

$r_{2n+1}: notify \leftarrow \text{not } try\_query$,

$(1 \le i \le n, n+1 \le j \le 2n)$ 以及一条事实r_0:date(0)作为一个初始的时间戳.这里, $r_1,...,r_n$ 表示构件库 c_i 拥有目标构件而且它的服务是开放的,可以直接查询 $r_{a+1},...,r_{2n+1}$ 用于检测是否有站点被查询到.如果没有站点被查询到,会使得notify还为真.

假设有一个事件 E,它是形如 $possess(c_i)$的平凡规则集合,说明构件库 c_i 在日期 t 拥有目标构件,使得 E 包含至多一个时间戳 $date(.)$.更新策略 U 可以被定义为一个如下的 IMPACT Agent 程序.假设该 Agent 程序包含上述规则,以及:

Do always($possess(C) \leftarrow date(T)) \leftarrow$

Do assert ($possess(C) \leftarrow date(T)$), in($C$,$repository()$),in($T$,$dates()$)

Do cancel($possess(C) \leftarrow date(T)) \leftarrow$

in($dates(T)$,$isBel()$),$T \neq T$,in($date(T^1)$,$event()$),

in(C,$repository()$),in(T,$dates()$),

in(T',$dates()$);

Do retract($possess(C) \leftarrow date(T)) \leftarrow$

in($dates(T)$,$isBel()$),$T \neq T$,in($date(T^1)$,$event()$),

in(C,$repository()$),in(T,$dates()$),

in(T',$dates()$);

说明:第 1 条规则确认了一条关于在某个日期构件库中拥有目标构件的信息,它保证了对于给定的日期是有效地:而第 2 条规则说明在某个更近的日期,该构件库不再拥有期望的构件;因此第 3 条删除了原先的拥有信息(假设时间持续增长).这里我们引用了两个代码调用(由 *repository*()和 *dates*()给定),它们返回所有可能的对构件库和日期的赋值.此外,U 中还包含如下规则:

Do retract($date(T)) \leftarrow date(T)) \leftarrow$

in($dates(T)$,$isBel()$),$T \neq T$,in($date(T^1)$,$event()$),

in(T,$dates()$),in(T^1,$date$,$event()$),

Do ignore($possess(C_1) \leftarrow$in($possess(C_1)$,$event()$)

Do ignore($possess(C_1) \leftarrow date(T) \leftarrow$in($possess(C_1)$,$\leftarrow date(T)$,$event()$),in($T$,$date$))

第 1 条规则使时间截"$date(t)$"在KB中保持唯一,因此在有新的日期值被添加时,旧的值将被删除.后面两条说明了与旧日期相关的关于构件库C_1的拥有信息被忽略.

因此,可更新 Agent 通过在它接收到的每个事件上计算一个新的(合理的)状态集,以及通过执行动作 $cmd(t)$,使得动作状态原子 D $cmd(t)$在状态染中,来实现其更新策略.

例如,假设例 1 中描述的更新Agent由事件{($possess(c_1)$,$date(1)$}触发.那么,它的合理状态集包括动作状态原子assert($possess(c_2)$,$assert(date(1)$,以及$retract(date(0)$,说明在日期 1,构件库c_2也拥有了目标构件,通过执行相应的动作,知识库就可以被Agent更新.

3.3 案例分析

下面通过一个设计实例,说明在多 Agent 环境中知识复用的设计过程.

其中设计活动行为可定义为多个SA,加入到Agent集A_{set},其中主要包括界面Agent、交互Agent、通信Agent、各种功能Agent(搜索Agent、响应Agent、参数一图形转换Agent等).

有关知识服务包括:① 概念类知识服务;② 设计方法类知识服务;③ 标准、规范类知识服务;④ 计算类知识服务;⑤ 参数类知识服务;⑥ 2 维、3 维图及运动模拟类知识服务;⑦ 与 Agent 关联的零部件类知识服务;⑧ 工艺类知识服务.这些知识服务所关联的基本领域知识包括[16]:① 典型形式;② 设计方法;③ 强度计算方

法(静强度、疲劳、刚度);④ 周围构件如蒙皮、桁梁、长桁、纵向隔板、接头、系统附件等与框的连接关系;⑤ 作用于框上载荷的作用点和性质;⑥ 外形.这些基本的领域知识存储于构件知识库中.

设计师将设计任务分解后,每个设计 Agent 根据自己的任务协助设计师完成相应的构件设计.采用遗传算法生成构件知识.

算法 1. 基于遗传算法的构件知识生成算法.

Step 1. 初始化种群.种群通过随机在操作数和操作符集中选择形成的数学表达式来生成.采用堆栈及数据结构中相应的算法检查产生的数学表达式是否是一个有效的数学表达式及表达式中的括号是否平衡,然后,把生成的表达式作为字符串,根据运算顺序,用解析(parsing)算法构造数学表示二叉树.如果新产生的树无法用计算机显示生成的形状,则该种子被淘汰.

Step 2. 通过与设计师交互,得到种群中个体的适应度值.设计师可以选择他们感兴趣的二维草图,查看其三维图像,然后再修改适应度值.

Step 3. 根据群体中个体的适应度值,确定新的种群.

Step 4. 对种群执行交叉.分别在父辈树上随机选择一个交叉点,并交换以交叉点为根的两棵子树,产生两个后代.如果新产生的树不能映射力合法的数学表达式,或者无法用计算机显示生成的形状,则被淘汰.以图 1 中的两棵树作为父辈树,在交叉点 A 处实施交叉操作,得到的后代形状(下半部)如图 2 所示.

设计 Agent 对图 2 所示的 2 雏形状扫描,然后经适当的艺术处理后生成的 3 维图像.设计师根据产生的 3 维图像给喜好的形状以较高的适应度值,以便后续过程中产生的后代能继承这些优点,也可以直接选择保存.被设计师选中的图像以.sat 的文件格式保存起来,可以用其他设计工具软件进行完善和渲染.

Step 5. 对种群执行变异操作.

在单父辈树上随机选择一个变异点,然后用一棵随机产生的子树替换以变异点为根的子树.如果产生的后代树不能映射为合法的数学表达式,或者无法用计算机显示生成的形状,则被淘汰.

Step 6. 如果设计师不选择退出,则转 Step 2.

Step 7. 结束.

该选择、交叉及变异运算过程一直进行到被设计师中止.生成的构件被分类并保存在构件库中.装配 Agent 执行基于二进制编码的遗传算法产生各构件的组合布局,然后由设计组工程师最后作出决策.在构件组合过程中,装配 Agent 进行组装限制检查,并针对不符合限制条件的构件,通过通讯 Agent 向设计 Agent 发送要求修改的信息.每个设计阶段结束,设计结果将被评估,由设计专家决策是否进行进一步的修改,重复进行再设计过程,直至满足要求.选定的组合方案以.sat 文件保存,并提交给设计师进行详细设计.

图 1 所示为两棵数学表达式二叉树.表达式分别为 $\cos 2x(1-x)(0.9-x)$和$(\sin x+0.4)(0.3+\sin 3x)$.

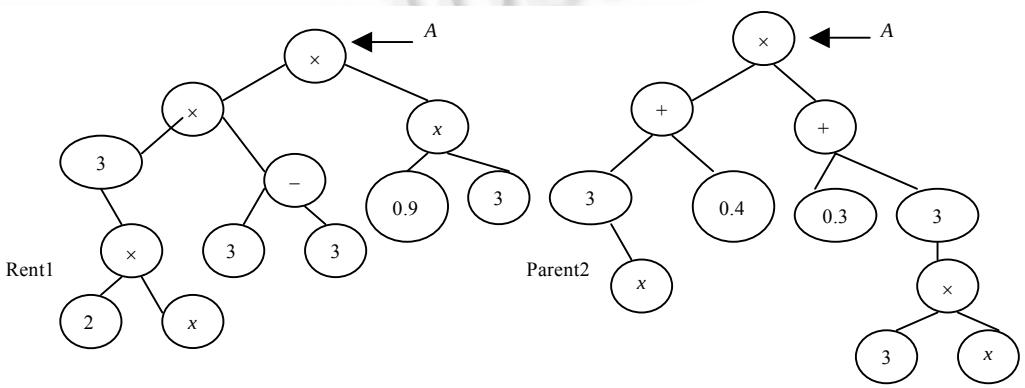

图 1 两棵数学表示二叉树

4 结 论

本文针对如何实现业界知识和软件复用的问题,提出了一种基于多 Agent 和构件技术的知识复用模型,以方便企业实现知识的动态复用和应用过程的动态重组,增强知识管理系统的分布式处理能力和规模可扩展能力,降低知识管理系统的构造成本,提高系统的稳定性和健壮性,有益于企业实现有效的知识管理,实现知识资源的共享和充分运用,提高企业在复杂商务环境下的竞争力.但知识管理中的知识复用问题目前还是一个新的研究课题,无论在理论上还是在实现技术上都还处于初始阶段,所以本文的研究工作旨在为知识复用技术的实践做初步的探索.未来进一步的工作包括提高 Agent 对于大规模知识的处理能力,以及 Agent 对于不完整的、不一致的更新信息的智能处理水平.

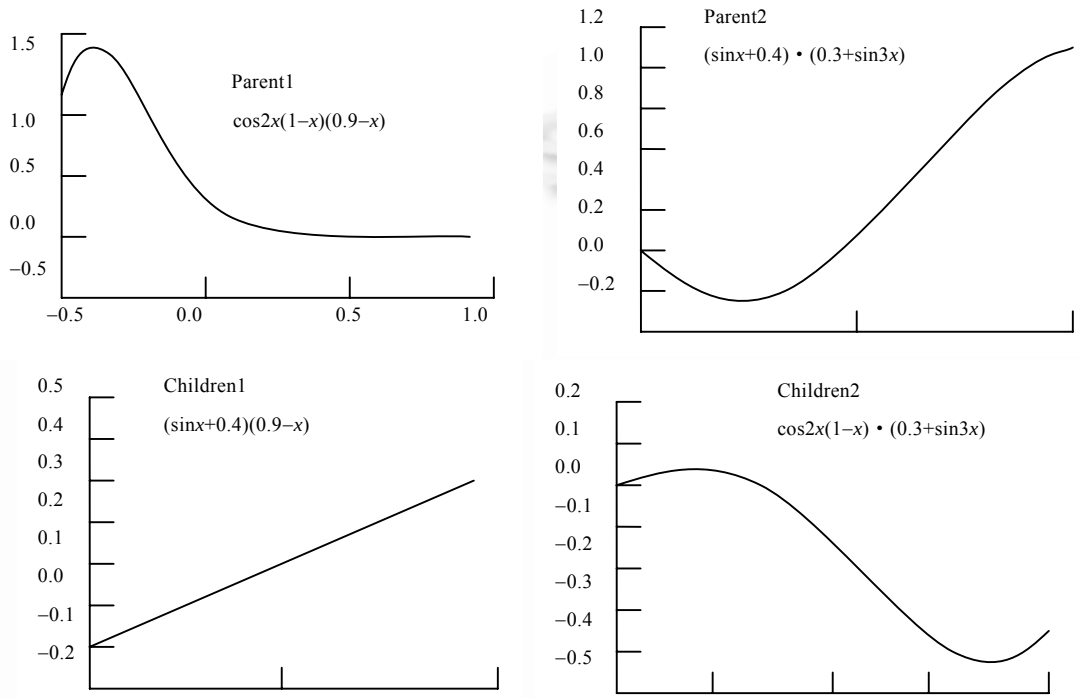

图 2 父辈树及交叉操作生成的后代

References:

[1] Sun X, Zhuang L, Liu W, Jiao WP, Mei H. A customizable running support framework for autonomous components. Journal of Software, 2008,19(6):1340−1349 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/19/1340.htm

[2] Chen X. An approach to refining active components based on component calculus. Journal of Software, 2008,19(5):1134−1148 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/19/1134.htm

[3] Bel-Enguix G, Grando MA, Jimenez-Lopez MD. Grammatical framework for modeling multi-agent dialogues. In: Shi ZZ, Sadananda R, eds. Agent Computing and Multi-Agent Systems. LNAI 4088, Berlin, Heidelberg: Springer-Verlag, 2006. 10−21.

[4] Chen HB, Yang Q, Xu MW. A calculus for MAS Interaction protocol. In: Shi ZZ, Sadananda R, eds. Agent Computing and Multi-Agent Systems. LNAI 4088, Berlin, Heidelberg: Springer-Verlag, 2006. 22−33.

[5] Yuan Y, Chen SQ. Constructing the distributed application orienting to inference components. Mini-Micro Systems, 2003, 24(8):1480−1482 (in Chinese with English abstract).

[6] Yuan Y, Chen SQ. An architecture for reusable KBS based on the interoperations between components. J. CENT. SOUTH UNIV. TECHNOL., 2003,34(2):177−180.

[7] Yang Z, Hu JZ, Hu LJ. Component retrieval based on knowledge base and case reasoning. Computer Engineering, 2005,31(21): 159−162 (in Chinese with English abstract).

[8] Gao ZN. Model checking for epistemic and temporal properties of uncertain agent. In: Shi ZZ, Sadananda R, eds. Agent Computing and Multi-Agent Systems. LNAI 4088, Berlin, Heidelberg: Springer-Verlag, 2006. 46−58.

[9] O'Reilly GB, Ehlers E. Synthesizing stigmergy for multi agent systems. In: Shi ZZ, Sadananda R, eds. Agent Computing and Multi-Agent Systems. LNAI 4088, Berlin, Heidelberg: Springer-Verlag, 2006. 34−45.

[10] Lee J, Kwak B. A task management architecture for control of intelligent robots. In: Shi ZZ, Sadananda R, eds. Agent Computing and Multi-Agent Systems. LNAI 4088, Berlin, Heidelberg: Springer-Verlag, 2006. 59−70.

[11] Lü J, Tao XP, Ma XY, Hu H, Xu F, Cao C. On agent-based software model for Internetware. Science in China (Series E), 2005, 35(12):1233−1253 (in Chinese with English abstract).

[12] Kemke C. Natural language communication between human and artificial agents. In: Shi ZZ, Sadananda R, eds. Agent Computing and Multi-Agent Systems. LNAI 4088, Berlin, Heidelberg: Springer-Verlag, 2006. 84−93.

[13] Liu JJ, Zhang SS, Wang YL. Research on knowledge sharing and reuse among agents with multiple ontologies. Computer Integrated Manufacturing Systems, 1999,5(6):51−54 (in Chinese with English abstract).

[14] Yang FQ, Mei H, Lü J, Jin Z. Some discussion on the development of software technology. Acta Electronica Sinica. 2002,30(z1): 1901−1906 (in Chinese with English abstract).

[15] Chang ZM, Mao XJ, Qi ZC. Component model and its implementation of Internetware based on agent. Journal of Software, 2008,19(5):1113−1124 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/19/1113.htm

[16] Cai WQ, Peng PL, Jiang SS. Knowledge representation and reuse in aircraft design. Computer Integrated Manufacturing Systems, 2004,10(1):55−58 (in Chinese with English abstract).

[17] Jiang WJ, Xu YS, Hao D, Zhen SY. Research on multi-agent system automated negotiation theory and model. In: Proc. of the NPC 2005. LNCS 3779, 2005. 317−320.

[18] Jiang WJ, Lin XH. Research on a novel method diagnosis and maintenance for key produce plant based on MAS and NN. In: Proc. of the ICONIP 2006. LNCS 4233, Springer-Verlag, 2006. 870−879.

[19] Pan X, Wang J, Liu L. Research on framework of knowledge management integration and key technologies in digital manufacturing enterprise. Computer Integrated Manufacturing Systems, 2004,10(S1):90−95 (in Chinese with English abstract).

[20] Liu H, Lin ZK. A cooperative design approach supporting dynamic task assignation. Journal of Software, 2001,12(12):1830−1836 (in Chinese with English abstract). http://www.jos.org.cn/ch/reader/view_abstract.aspx?flag=1&file_no=20011213&journal_id=jos

[21] Wang B, Zhang YX, Chen SQ. A communication method of MAS based on blackboard architecture. Mini-Micro Systems, 2002, 23(11):1355−1358 (in Chinese with English abstract).

[22] Jiang WJ. A novel algorithm of neural network optimized design. Chinese Journal of Electronics, 2006,15(4A):925−928 (in Chinese with English abstract).

[23] Mei H, Cao DG. Enabling separation of crosscutting concerns in component-based software development. Chinese Journal of Computers, 2005,28(12):2036−2044 (in Chinese with English abstract).

蒋伟进(1965－),男,博士生,教授,主要研究领域为智能计算,Agent 计算.

夏可(1966－),女,博士生,主要研究领域为软件工程,软件自动化研究.