一种基于 Agent 的自适应软件过程模型*

赵欣培[1+], 李明树[1], 王 青[1], 陈振冲[2], 梁金能[2]

[1](中国科学院 软件研究所,北京 100080)

[2](香港理工大学 电子计算学系,香港)

An Agent-Based Self-Adaptive Software Process Model

ZHAO Xin-Pei[1+], LI Ming-Shu[1], WANG Qing[1], Keith Chan[2], Hareton Leung[2]

[1](Institute of Software, The Chinese Academy of Sciences, Beijing 100080, China)

[2](Department of Computing, The Hong Kong Polytechnic University, Hong Kong, China)

+ Corresponding author: Phn: +86-10-82620803, E-mail: xinpei@itechs.iscas.ac.cn, http://www.itechs.com.cn

Received 2003-05-23; Accepted 2003-09-26

Zhao XP, Li MS, Wang Q, Chan K, Leung H. An Agent-based self-adaptive software process model. *Journal of Software*, 2004,15(3):348~359.

http://www.jos.org.cn/1000-9825/15/348.htm

Abstract: Traditional software process models are mostly static, mechanical, and passive. Traditional approach requires modeler to determine all the possible conditions the software process will encounter and to define explicitly the solutions into a process model. It lacks the ability to allow further deliberations when the modeled environment changes. This paper presents an Agent-based self-adaptive software process model. In this approach, software process is modeled as peers: process Agents. These software process Agents can adapt themselves to the software process environment and act with initiative and autonomy. When the process environment changes, the process agents can dynamically change their behavior to ensure that the development goal can still be achieved.

Key words: software process model; software process; self-adaptive; Agent; artificial intelligence

摘 要: 传统的软件过程模型大多是静态的、机械的、被动的,它们要求软件工程人员在描述软件过程时预期所有可能发生的情况,并且显式地定义这些问题的解决方案.当软件过程所处的环境发生变化时,软件过程无法自适应地对这些变更作出相应的调整.提出了一种基于 Agent 的自适应软件过程模型.在这种软件过程模型中,软件过程被描述为一组相互独立而对等的实体——软件过程 Agent.这些软件过程 Agent 能够对软件过程环境的变化主动地、自治地作出反应,动态地确定和变更其行为以实现软件开发的目标.

关键词: 软件过程模型;软件过程;自适应;Agent;人工智能

---

* Supported by the National Natural Science Foundation of China under Grant No.60273026 (国家自然科学基金); the National High-Tech Research and Development Plan of China under Grant No.2001AA113080 (国家高技术研究发展计划(863))

作者简介: **赵欣培**(1977－),男,北京人,硕博连读生,主要研究领域为软件质量管理,人工智能;**李明树**(1966－),男,研究员,博士生导师,主要研究领域为智能软件工程,实时系统;**王青**(1964－),女,研究员,主要研究领域为软件质量管理,过程建模,知识管理,软件协同工作,实时系统开发方法;**陈振冲**(1959－),男,副教授,主要研究领域为软件工程,数据挖掘,计算智能;**梁金能**(1957－),男,副教授,主要研究领域为软件测试,软件维护,过程工程和质量改进,软件度量,软件外包.

中图法分类号: TP311 文献标识码: A

软件过程建模技术已经有十几年的发展历史,这期间提出了很多软件过程建模方法和软件过程模型[1].传统的软件过程建模方法沿用了制造工业过程的建模方法对软件过程进行建模,例如,有限状态自动机[2]、Petri网[3]等.但是,软件过程与传统制造业的过程有着很大的不同.软件过程具有明显的不确定性,过程能力和人、环境、技术以及对用户需求的理解都有很大的依赖性和相关性.传统的软件过程模型大多是静态的、机械的、被动的,它们要求软件工程人员在描述软件过程时预期所有可能发生的情况,并且显式地定义这些问题的解决方案.当软件过程所处的环境发生变化时,这种软件过程难以自适应地对这些变更作出相应的调整,只能依赖软件工程人员的干预而对其进行修改,这就使得软件工程人员不得不将大量的时间和精力浪费在软件过程的变更上,加大了软件组织(企业)的管理成本.

解决软件过程模型“刚性”问题的一个有效途径是在软件过程模型中引入自适应特性,即模型对应的软件过程能够自治地调整自身的行为以适应软件过程环境的变化,并确保在性能测度最大的条件下实现软件开发的目标.Agent 是一种能够主动地观察所处的环境,并在这些观察的基础上自治地实施其行为的系统[4],它具有良好的自适应特性.因此我们考虑在软件过程建模中引入 Agent 技术,使用 Agent 来描述软件过程的组成要素,并使得 Agent 能够表现软件过程的行为,从而利用 Agent 的自适应机制实现软件过程在任务规划、资源分配、过程产品管理以及过程协同方面的自适应特性.另一方面,Agent 技术本身是为了解决分布式人工智能问题而提出的,具有先天的分布特性.应用该技术而构造的基于 Agent 的软件过程模型也能够有效地支持软件产品的多点开发.

1 相关工作

近年来,将 Agent 技术引入过程建模的尝试日益得到广泛关注.这些尝试主要分为两种类型,即 Agent 增强的过程模型(Agent-enhanced process model)以及基于 Agent 的过程模型(Agent-based process model).

Agent 增强的过程模型关注于将 Agent 技术与已有的工作流形式的过程模型进行集成.在 Agent 增强的过程模型中,应用于过程建模的形式化方法仍然采用传统的工作流形式,而在管理层面上,例如过程变更和分布式协同上,由 Agent 实现控制[5].这种过程模型的优点在于对已有的过程和系统的重用上,这样的解决方案能够在保留组织已经存在并投入应用的过程以及系统的基础上,通过引入 Agent 技术来实现过程的自适应特性.其缺点在于保留了传统过程模型的因素,对于过程自适应特性的实现总具有片面性.例如:应用了 Agent 来实现组织间的过程协同,但是组织内部的过程仍然不具有自适应特性.而且,这样的解决方案也不是统一的,它需要引入很多异种的 Agent 来对过程的各个方面实施管理,这从一定程度上增加了问题的复杂程度.近年来,关于 Agent 增强的软件过程模型的研究工作主要集中在分布式的软件过程协同上,例如文献[6].

基于 Agent 的过程模型使用 Agent 对过程的元素及其相互的联系直接予以描述.这样的模型从根本上引入了 Agent 的自适应特性.基于 Agent 的过程模型的思想产生于人工智能领域,其目的在于建立基于 Agent 的业务模型,从而实现对服务的整合以及任务的分布式管理,例如:Zeng 提出的 ABWFMS 模型[7]以及 Gou 提出的 ABMEs 模型[8].ABWFMS 模型引入了 3 种 Agent,即过程 Agent、监控 Agent 以及服务 Agent.其中,过程 Agent 实现任务的动态分配和分布式协同;监控 Agent 负责在本地监控任务的实施;服务 Agent 封装了任务实现的方法.通过这 3 种 Agent 的协同,ABWFMS 能够在分布条件下实现动态的工作流整合和服务整合.ABMEs 的目的是应用 Agent 技术对虚拟组织建立模型.ABMEs 模型引入了两种 Agent,即实现过程活动动态整合的活动 Agent 和封装了活动实现的角色和方法的资源 Agent,这两种 Agent 通过合同网络协议实现协同.上述基于 Agent 的过程模型的问题主要在于其并非针对软件过程领域,因而忽略了很多软件过程中的关键要素,例如过程活动的执行角色,过程的资源分配等等.另一方面,目前的基于 Agent 的过程模型中大多包含多种职能的 Agent.这样的结构使得 Agent 之间存在着异种的交互关系,因而结构上较为复杂.

针对上述过程模型的不足,我们在基于 Agent 的过程模型基础上提出了一种自适应的软件过程模型.首先,

我们在模型中引入了过程资源和过程角色的描述.过程 Agent 在确定其行为的过程中,每个环节上都要考虑其是否拥有足够的资源,并且在实施其行为的时候能够确定其对应的角色是否拥有适当的能力.此外,我们试图通过统一的一种过程 Agent 来建立这种基于 Agent 的过程模型.在我们的模型中,过程 Agent 封装了与之相关的过程知识、活动、资源、环境以及活动的实施角色等等,这种模型具有高内聚、低耦合的特性,具有更为良好的系统结构.

2 基于 Agent 的自适应软件过程模型

软件过程模型描述了软件过程要素(如:活动,资源,角色,过程产品等)以及这些要素之间的关系.本文所提出的基于 Agent 的自适应软件过程模型的基本思想就是通过过程知识的形式描述软件过程要素的语义,以及在这些要素之间建立关联的规则,并通过 Agent 的推理和自适应机制针对环境的变化而动态地将相关的软件过程要素组织成软件过程,从而实现在任务规划、资源分配、产品管理以及过程协同方面的自适应处理.

图 1 显示了一个自适应软件过程模型及其在 ISPW-6[9]软件过程范例所描述的目标和过程环境条件下的一个过程实例.

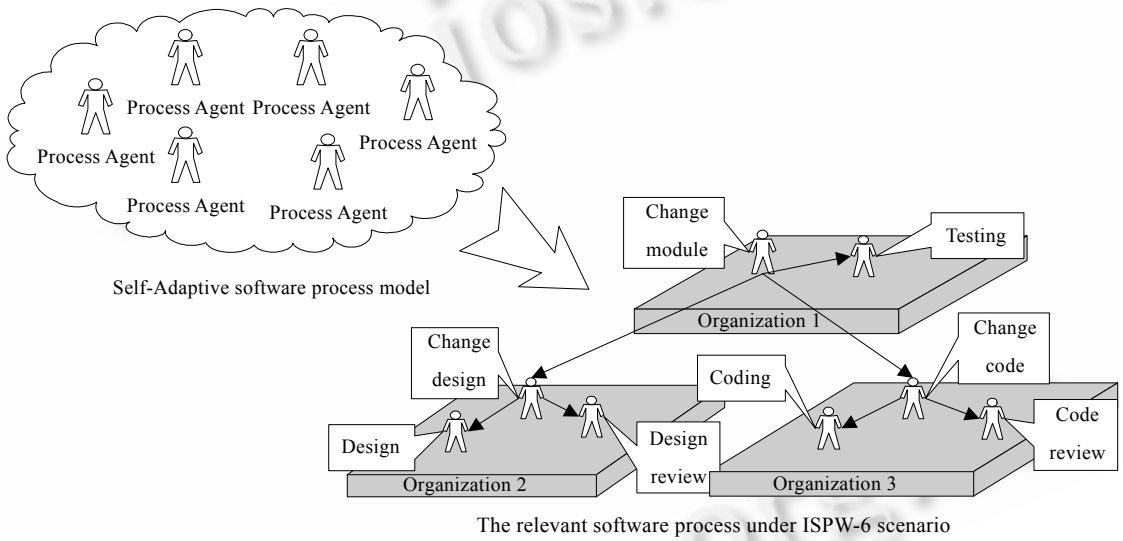

Fig.1 Self-Adaptive software process model and its relevant software process

图 1 自适应软件过程模型及其所描述的自适应软件过程

事实上,自适应软件过程模型给出了一组过程 Agent 的描述,其中每个过程 Agent 的描述封装了与之相关的过程要素,以及如何在这些要素之间建立关系的过程知识.当过程被实现时,过程 Agent 根据特定的目标和环境状态动态地确定过程要素之间的关系以及过程 Agent 之间的协作关系,从而系统地实现软件开发的目标,并且在目标和环境变化时自治地调整这些关系,从而在新的条件下确保目标的实现(图 1 中仅显示了自适应软件过程在某个特定目标和过程环境状态下的状态.事实上,自适应软件过程中过程要素之间的关系和过程 Agent 之间的关系是动态的).

本节中我们将给出一个一般的自适应软件过程模型,并由此讨论自适应软件过程的组成、结构及其自适应特性的实现.自适应软件过程具有静态特征(过程知识和过程要素)和动态特征(在过程要素之间动态地建立关系以及针对目标和环境的变化自治地进行调整等),这里我们通过对这两个侧面分别建立模型(过程描述模型和过程实施模型)的方法来对自适应软件过程进行描述.其中,过程描述模型通过过程 Agent 描述了软件过程的组成要素以及如何在这些要素之间建立关系的过程知识;过程实施模型描述了过程 Agent 如何通过其知识,动态地确定过程要素之间的关系(单一过程 Agent 的实现模型)及其相互之间的关系(过程 Agent 的协同模型).

图 2 显示了本文中讨论的各个模型之间的关系(图中,模型之间的关系是“由…组成”).

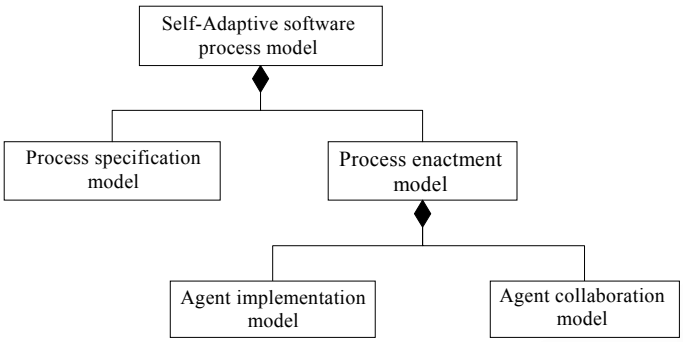

Fig.2 Relationship among the models

图 2 模型之间的关系

为了不失一般性,我们约定在以后的讨论中,在不引起混淆的情况下,并不区分过程和软件过程的概念.

2.1 过程描述模型(process specification model)

过程描述模型从结构上描述了软件过程的组成要素及其相互关系.为了简化模型的描述,我们首先定义资源约束和控制规则模型的概念:

定义 1. 一个资源约束(resource constraint)是一个 3 元组 $Res=(hr,t,f)$,其中,hr 描述了人力资源约束,一般以人月为单位;t 描述了时间约束,一般以天为单位;f 描述了资金约束,一般以元为单位.我们还定义针对资源约束的运算:$Res_1\ op\ Res_2=(hr_1\ op\ hr_2,t_1\ op\ t_2,f_1\ op\ f_2)$(其中 op 为+,–),以及资源约束之间的关系 $Res_1\ re\ Res_2$ 当且仅当 $hr_1\ re\ hr_2,t_1\ re\ t_2,f_1\ re\ f_2$(其中 re 为<,>,=).

资源约束被用来描述过程要素(如目标,过程知识等)在资源上的约束.

定义 2. 一个控制规则模型(control rule model)是一个 3 元组 $CRM=(PreC,PostC,Inv)$,其中

(1) $PreC$ 是一组前件(preconditions):$PreC=\{prec_1,prec_2,\ldots,prec_n\}$

(2) $PostC$ 是一组后件(postconditions):$PostC=\{postc_1,postc_2,\ldots,postc_n\}$

(3) Inv 是一组不变量(invariants):$Inv=\{inv_1,inv_2,\ldots,inv_n\}$.

控制规则模型描述了对某个过程行为的控制规则:当前件被满足时,该行为可以被实施;后件是行为预期的结果;不变量是行为实施过程中的约束,即在行为实施过程中,不变量的偏离将造成该行为的失败.

直观地讲,一个自适应软件过程由一组过程 Agent 组成.这些过程 Agent 位于某个特定的环境并能够实施主动、自治的行为以实现软件开发的目标:

定义 3. 形式地,一个自适应软件过程是一个 3 元组:$SASP=(S,G,PA)$,其中

(1) S 是一组过程环境状态 $S=\{s_1,s_2,\ldots,s_n\}$;

(2) G 是一组过程目标(goal),$G=\{g_1,g_2,\ldots,g_n\}$.其中,一个过程目标是一个 2 元组 $g_i(GS_i,GC_i)$:

a) GS 是一组目标描述(goal statement),$GS=\{gs_1,gs_2,\ldots,gs_n\}$;

b) GC 是目标约束(goal constraint),它是一个资源约束;

(3) PA 是一组相互独立而对等的过程 Agent,$PA=\{pa_1,pa_2,\ldots,pa_n\}$.

定义 3 中的过程 Agent 被定义为:

定义 4. 一个过程 Agent 是一个 5 元组 $pa_i=(S_i,PP_i,Res_i,A_i,K_i)$.其中

(1) S_i 是相对于该过程 Agent 的局部过程环境状态,同时有 $S_i\subseteq S$;

(2) PP_i 是该过程 Agent 的对应角色的描述.它描述了该角色所应具备的条件.

(3) Res_i 是该过程 Agent 的资源约束,它描述了 Agent 所拥有的资源数量.

(4) A_i 是一组过程 Agent 能够实施的活动(activity),$A_i=\{a_{i1},a_{i2},\ldots,a_{in}\}$.其中,一个活动是一个 4 元组 $a_{ij}=(CRM_{a_{ij}},AP_{a_{ij}},Res_{a_{ij}},Imp_{a_{ij}})$.

a) $CRM_{a_{ij}}$ 是活动的控制规则模型,它描述了如何控制该活动的实施;

b) $AP_{a_{ij}}$ 是活动的对应角色的描述.它描述了该角色所应具备的条件;

c) $Res_{a_{ij}}$ 是活动的资源约束,它描述了实施该活动所需要的资源.一般地, $Res_{a_{ij}} \le Res_i$;

d) $Imp_{a_{ij}}$ 描述了活动的实现方法,它的取值可以是人工实现(manual)、过程应用(application)或者子过程(subprocess).

(5) K_i 描述了过程知识(process kowledge),它由一组过程步骤描述组成:$K_i=\{st_{i1},st_{i2},\ldots,st_{in}\}$.其中,一个过程步骤描述是一个 3 元组 $st_{ij}=(CRM_{st_{ij}},Res_{st_{ij}},Imp_{st_{ij}})$:

a) $CRM_{st_{ij}}$ 是步骤的控制规则模型,它描述了如何控制该步骤的实施;

b) $Res_{st_{ij}}$ 是步骤的资源约束,它描述了实施该步骤所需要的资源;

c) $Imp_{st_{ij}}$ 描述了步骤的实现方法,它的取值可以是人工实现、过程应用或者子过程.

自适应软件过程的核心是过程 Agent,它们封装了局部的过程相关元素,包括:环境状态(活动的执行状态、输入和输出等)、对应的角色、拥有的资源、过程活动以及赖以确定过程行为的过程知识.

定义 3 和定义 4 给出了自适应软件过程的过程描述模型.在该模型中,过程要素之间的关系并不是被模型显式地描述的,例如活动的执行顺序、角色与活动之间的对应关系、过程资源以及目标的资源约束之间的关系等等.事实上,过程知识描述了在过程要素之间建立关系的规则,而关系则是在过程被实现时针对特定的目标和过程环境而动态建立的.

可以注意到,定义 4 中过程 Agent 的活动描述与过程知识描述是类似的,但是我们把过程 Agent 的活动和过程知识中的步骤定义为不同的概念.尽管具有类似的结构,过程知识描述的是过程 Agent 对于如何确定和控制其行为在知识层次上的抽象,它独立于过程 Agent 实际能够实施的活动.这种结构能够使过程知识和活动在相对独立的情况下进行变更.活动同过程知识通过绑定(bindings)实现关联.

定义 5. 过程知识和活动之间依靠绑定关系相互关联,一个绑定是二元组 $B=(a_i,st_j)$,其中 $a_i=(CRM_{a_i},AP_{a_i},Res_{a_i},Imp_{a_i})$,$st_j=(CRM_{st_j},Res_{st_j},Imp_{st_j})$, $CRM_{a_i}=(PreC_{a_i},PostC_{a_i},Inv_{a_i})$, $CRM_{st_j}=(PreC_{st_j},PostC_{st_j},Inv_{st_j})$,且有

(1) $PreC_{st_j} \to PreC_{a_i}$, $PostC_{a_i} \to PreC_{st_j}$, $\neg(Inv_{a_i} \to \neg Inv_{st_j})$, $\neg(Inv_{st_j} \to \neg Inv_{a_i})$;

(2) $Res_{a_i} \le Res_{st_j}$;

(3) $Imp_{a_i} = Imp_{st_j}$.

2.2 过程实施模型(process enactment model)

自适应软件过程的描述模型描述了自适应软件过程的组成要素及其相互的关系.结构上,一个自适应软件过程是由一组过程 Agent 组成的.它们都具有一定的过程知识,这些知识描述了在过程要素(活动,角色,资源等)之间建立联系的规则.在这些过程知识的基础上,过程 Agent 能够根据对环境的观察自主地决定所要采取的行为.

在图 1 中,我们给出了一个针对 ISPW-6 的自适应软件过程.这个软件过程中引入了 8 个过程 Agent,这 8 个过程 Agent 形成了某种结构.连接于箭头尾段的过程 Agent 被称作管理者 Agent(manager Agent),它拥有实现某个目标的知识,并针对目标的实现确定一系列的活动.对于某些自身无法实现的活动,管理者 Agent 能够将其转化为子目标进行发布,并交给意图实现这些子目标的下属 Agent(underling Agent,即连接于箭头头部的过程 Agent)予以实现.例如,在图 1 的例子中,负责设计变更的过程 Agent 将设计变更的目标分解为设计和设计评审两个子目标,并交由其他两个过程 Agent 予以实现.过程运行时,管理者 Agent 跟踪其下属的行为,并对变更进行处理.过程产品也将被管理者 Agent 收集和处理,并向它的上级传递.

上面这些过程 Agent 的行为是由自适应软件过程的实施模型所确定的.本节中,我们将对该模型进行讨论.实施模型描述了过程 Agent 如何根据其知识动态地确定其行为,自治地适应软件过程环境的变化,并相互协作地实现软件开发的目标.它包括两个部分:单一过程 Agent 的实现模型以及过程 Agent 间的协同模型.

2.2.1 单一过程 Agent 的实现模型——反射模型

对于每种可能的感知序列,过程 Agent 能够在感知序列所提供的证据和过程知识的基础上确定其所要采取的活动,使它的性能测度为最大,亦即在消耗最少资源的前提下实现其期望的目标.我们采用一种反射模型来实现过程 Agent.这种反射模型可以通过下列要素描述:一组关于世界的信念;过程 Agent 当前打算达到的一组目标(愿望);一组规划,描述怎样达到目标和怎样改变信念.

2.2.1.1 过程 Agent 的信念(belief)

信念描述了过程 Agent 所相信的事情.这里我们定义信念算子 $\mathcal{Bel}$,对于过程 Agent pa 以及命题 P,$\mathcal{Bel}_{pa}P$ 的含义为过程 Agent pa 相信 P.直观地讲,一个过程 Agent 应该相信其推理所依据的前提(过程知识以及过程 Agent 对环境的感知)并且相信所有基于这些前提的逻辑推论,同时,它不相信在这些前提及其推论以外的事情.

定义 6. $\mathcal{B}_{Apa}$ 是过程 Agent pa 的一个基于一组前提 A_{pa} 的信念集合,当且仅当 $\mathcal{B}_{Apa}$ 是下面集合的元素及其所有逻辑推论所组成的集合,$A_{pa}\cup\{\mathcal{Bel}_{pa}P\mid P\in\mathcal{B}_{Apa}\}\cup\{\neg\mathcal{Bel}_{pa}P\mid P\notin\mathcal{B}_{Apa}\}$.

由定义 6,过程 Agent 的信念是依赖于其过程知识及其对过程环境的感知的,因而也会随着过程知识和过程环境的变化而变化.定义 6 中的 B 是一个自认知逻辑(autoepistemic)下的理论(theory).自认知逻辑能够刻画 Agent 对于自己的知识和信念进行推理的规律,它具有非单调、唯理性和内省性的特点[10].

信念算子描述了过程 Agent 相信为必然的事情.同时,为了描述可能性,我们也定义了可能性算子 $\mathcal{Pos}$.对于过程 Agent pa 以及命题 P,$\mathcal{Pos}_{pa}P$ 的含义为 Agent 认为 P 是可能的,或者说 Agent 不相信 P 不可能.因此,我们可以在信念算子的基础上给出可能性算子的定义,亦即 $\mathcal{Pos}_{pa}P$ 被定义为 $\neg\mathcal{Bel}_{pa}\neg P$.

2.2.1.2 过程 Agent 的愿望(desire)

直观地讲,愿望是过程 Agent 所期望达到的目标,或者换句话说,是过程 Agent 相信其能够或者可能能够达到的目标.

定义 7. 过程 Agent 的愿望 $\mathcal{D}$ 是一组目标.对于某个自适应软件过程 $SASP=(S,G,PA)$*,以及某个过程 Agent $pa_i\in PA$,$pa_i=(S_i,P_i,Res_i,A_i,K_i)$,$\mathcal{D}_i$ 为 pa_i 的愿望当且仅当 $\mathcal{D}_i\subseteq G$,且对于 $\forall d\in\mathcal{D}_i$,$d=(GS,GC)$,都有 $\mathcal{Bel}_{pai}GS$ 或者 $\mathcal{Pos}_{pa_i}GS$,且 $GC\le Res_i$.

2.2.1.3 过程 Agent 的规划(how-to-do)

规划是过程 Agent 为实现其目标而确定的一系列步骤.形式上,过程 Agent 的规划 $\mathcal{H}$ 是一组过程步骤集合的序列(例如:$\{\{st_1\}\{st_2,st_3\}\{st_4\},\ldots\}$).

定义 8. 对于某个自适应软件过程 $SASP=(S,G,PA)$,以及某个过程 Agent $pa_i\in PA$,$pa_i=(S_i,P_i,Res_i,A_i,K_i)$,$\mathcal{H}_i$ 是一组过程步骤集合序列所组成的集合,且 $\mathcal{H}_i\subseteq(\mathcal{P}(K_i))^*$.

过程 Agent 的规划通过 Agent 的逆向推理机制实现,这种推理方法也被称作目标驱动的推理或自顶向下的推理[11].

算法 1. 令 $SASP=(S,G,PA)$为一个自适应软件过程,pa_i 为某个过程 Agent,$pa_i\in PA$,$pa_i=(S_i,P_i,Res_i,A_i,K_i)$,$\mathcal{D}_i$ 为该过程 Agent 的愿望,$d\in\mathcal{D}_i$,$d=(GS_d,GC_d)$,过程 Agent 通过实施下列步骤实现逆向推理:

(1) 令 Y 为一棵树,其根结点为空集,当前节点为根节点;

(2) 如果 $S_i|=GS_d$,则算法成功返回,否则

(3) 对于每一个步骤集合 $STS_i\in\mathcal{P}(K_i)$,如果 $(\bigcup_{st_{ij}\in STSi}st_{ij}.CRM.PostC)\cup S_i\ |=\mathcal{Bel}_{p_{ai}}\ GS_d$ 或者 $(\bigcup_{st_{ij}\in STSi}st_{ij}.CRM.PostC)\cup S_i|=\mathcal{Pos}_{p_{ai}}\ GS_d$,并且 $\sum_{st_{ij}\in STSi}st_{ij}.Res\le GC_d$,则将 STS_i 作为当前节点的一个子节点加入 Y;

(4) 对于当前节点的每一个子节点 STS_i,对 $d_i=(GS_i,GC_i)$ 递归(2)~(4)步,其中,$GS_i=\bigcup_{st_{ij}\in STSi}st_{ij}.CRM.PreC$,$GC_i=GC_d-\sum_{st_{ij}\in STSi}st_{ij}.Res$.如果失败返回,则删除该子节点;

* 这里我们使用过程描述模型来描述自适应软件过程,下同.

(5) 如果执行完步骤(4)后当前节点存在子节点,则成功返回,否则失败返回.

对树 Υ的深度和广度进行考察,可以证明:

定理 1. 在有限资源以及有限过程步骤的条件下,算法 1 是可终止的.

对每一个 $d\in\mathcal{D}_i$ 执行算法 1 后,我们都可以得到一棵树,该树中每一条从叶结点到根结点的路径都描述了过程 Agent 在实施愿望目标 d 时可能的步骤序列.我们选择累计开销最小的步骤序列作为规划集合的一个元素 h_{ij}. 对于过程 Agent 的愿望中的每一个元素实施上面的算法,我们可以得到规划集合 $\mathcal{H}_i$.

2.2.1.4 过程 Agent 的反射模型

前面我们给出了过程 Agent 的信念、愿望和规划的定义.形式上,过程 Agent 的反射模型可以被描述为下面一组映射:

定义 9. 对于某个过程 Agent $pa=(S,P,Res,A,K)$,其反射模型可以被描述为下面一组映射:

Belief: $S\times K\rightarrow\mathcal{B}$

Desire: $G\times\mathcal{B}\times Res\rightarrow\mathcal{D}$

How-to-do: $\mathcal{D}\times\mathcal{B}\times K\times Res\rightarrow\mathcal{H}$

Action: $\mathcal{H}\rightarrow A$

首先,过程 Agent 在对环境的感知及其过程知识的基础上确立信念集合 $\mathcal{B}$.基于过程 Agent 的信念及其所拥有的资源情况,过程 Agent 从过程目标集合中确定一组自己能够实现的目标生成其愿望集合 $\mathcal{D}$.接下来,过程 Agent 在其信念、过程知识以及资源情况的基础上,通过逆向推理过程针对其愿望建立规划集合 $\mathcal{H}$.每个规划都是一个过程步骤集合的序列,其中的每一个步骤都绑定于一个或多个过程活动,过程 Agent 选择其中资源消耗最小的活动予以实施.过程 Agent 的活动关联了特定的角色,这是由过程活动角色描述决定的.具体地说,只有满足角色描述约束的过程角色(例如,项目经理,系统工程师,测试员等等)才能确保活动的有效实施.另外,过程活动也关联了过程应用,这就使得过程 Agent 能够支持用户在一定的工具辅助下对过程活动予以实现.活动的实施最终影响到环境,并形成新的感知反馈给过程 Agent.过程 Agent 在这些新的感知和过程知识的基础上,修改自己的信念进而影响其愿望和规划,最终决定针对环境的变化而应采取的新活动.

上述 Agent 的推理过程可以被描述为图 3.这样的反射机制是由过程 Agent 自主实施的,不需要外来的干预.这种机制实现了过程 Agent 对过程环境的自适应特性. v_5 v_5 Rhode Island: AAAI Press, 1997. 50~56.

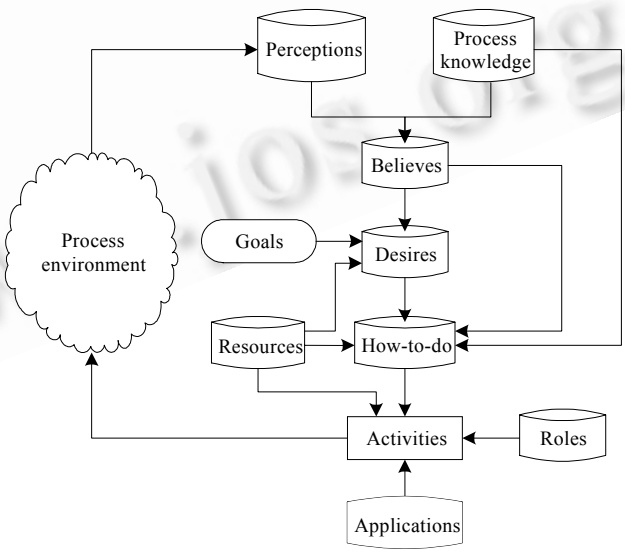

Fig.3 The reflective model of the process Agent

图 3 过程 Agent 的反射模型

2.2.2 过程 Agent 的协同模型

每个过程 Agent 都拥有针对软件开发目标的部分的、分布的知识,而软件开发目标的总体解决方案需要通过多过程 Agent 的协同才能实现.过程 Agent 通过形成某种组织实现协同,这样的组织我们称作管理网络,因为过程 Agent 之间的关系是管理与被管理的关系.

形式地,对于某个自适应软件过程 $SASP=(S,G,PA)$,一个管理网络被定义为

定义 10. 管理网络是一个四元组 $MN=(S_{MN},G_{MN},PA_{MN},R_{MN})$,且有 $S_{MN}\subseteq S,G_{MN}\subseteq G,PA_{MN}\subseteq PA$.其中 R_{MN} 是二元偏序关系(pa_1,pa_2),记作 $pa_1\ R_{MN}\ pa_2$ 或者 $pa_1\succ pa_2$.

直观地讲,关系 R_{MN} 是管理者 Agent 和下属 Agent 之间针对子目标的发布实现关系.事实上,对于其规划中无法绑定活动的步骤或者被描述为由子过程实现的步骤,过程 Agent 会将其作为子目标发布,该子目标会进而被其他某个过程 Agent 实现,从而在这两个过程 Agent 之间建立关系 R_{MN}(如图 4 所示).

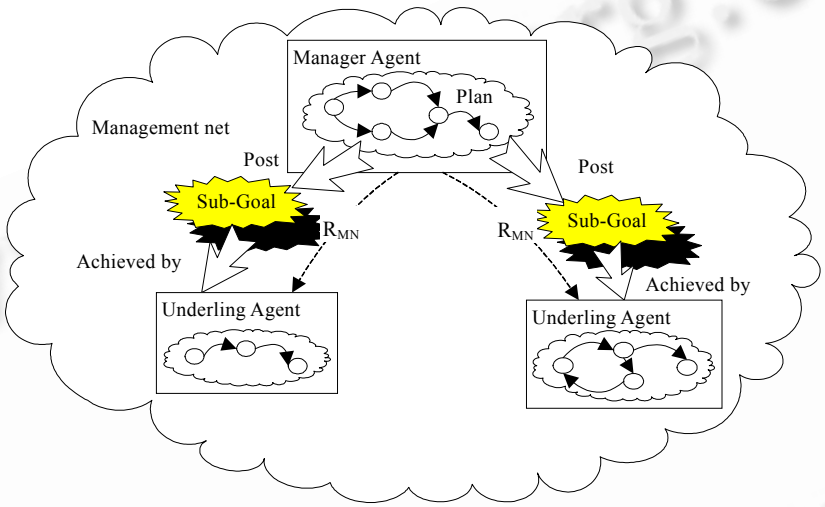

Fig.4 The management net

图 4 管理网络

形式上,我们用目标发布算子"⇒"来定义关系 R_{MN}.

定义 11. 定义算子⇒,$pa\Rightarrow g$ 表示过程 Agent pa 发布一个目标(子目标)$g=(GS,GC)$.

定义 12. $pa_1\ R_{MN}\ pa_2,pa_1,pa_2\in PA_{MN}$,当且仅当$\exists g,pa_1\Rightarrow g$ 且 $g\in \mathcal{D}_{pa_2}$.

管理网络的基本思想来源于分布式人工智能领域所广为使用的合同网络[12],即通过投标——合同的方式实现任务在多个节点上的分配.进一步地,管理网络在软件过程管理的角度上对合同网络进行了扩展.管理网络的结构是动态的,它可能因为过程 Agent 的信念、愿望和规划的变化而相应地发生变化.例如,作为管理者的过程 Agent 可能因为目标或过程环境的变化而变更其之前作出的规划.对规划的变更可能导致管理者 Agent 取消或者变更其之前发布的子目标,或者发布新的子目标,从而造成管理网络结构的变化.同时,作为下属的过程 Agent 也可能因为环境的变化或者一些突发情况(如某个活动产生了非预期的结果或者在活动执行时消耗了超过预期的资源等)而改变其信念和愿望,最终导致其退出管理网络.为了处理这些情况,管理网络包括一系列的针对软件过程的任务分派、跟踪、变更、过程度量及过程改进的算法和协议.限于篇幅,我们将在以后的文章中对管理网络的相关细节作进一步的讨论.

3 一个例子

为了通过实例来检验上面提出的模型,我们使用该模型描述了一个针对 ISPW-6 软件过程范例的自适应软件过程(即图 1 中给出的例子).本节中我们选择了 ISPW-6 的一个子过程——设计变更过程来进行讨论.它包括 4 个过程 Agent:设计变更、设计、设计修改以及设计评审.应用过程描述模型,该子过程被描述为 $SASP=(S,G,PA)$,

其中

S 为一组初始状态.S={*Requirement*(r),*RequirementChangeOf*(r,rc),*DesignOf*(r,d)};

目标 G 包含一个目标 g=(GS,GC),其中 GS={*FinalDesign*(d,fd)};

PA 包含 4 个过程 Agent,PA={pa_{Change_Design},pa_{Design},$pa_{Correct\_Design}$,pa_{Review_Design}},其中各个过程 Agent 的过程知识描述为:Kpa_{Change_Design}={st_1,st_2,st_3},Kpa_{Design}={st_4},Kpa_{Review_Design}={st_5},$Kpa_{Correct\_Design}$={st_6},其中

st_1=(CRM_{st_1} , Res_{st_1} , Imp_{st_1}),

其中 CRM_{st_1} :

Preconditions: *Requirement*(r)∧*RequirementChangeOf*(r,rc)∧*DesignOf*(r,d)

Postconditions: *Design*(d)∧*NewDesign*(d,nd)∧¬*Reviewed*(nd)

Invariants: ¬*ChangeVersion*(rc)

Imp_{st_1} =SUBPROCESS

步骤 st_1 描述了针对一个需求变更而对相应的设计文档进行修改.

st_2=(CRM_{st_2} , Res_{st_2} , Imp_{st_2}),

其中 CRM_{st_2} :

Preconditions: *Design*(d)∧*NewDesign*(d,nd)∧¬*Reviewed*(nd)

Postconditions: *Reviewed*(nd)∧*DesignReviewFeedback*(drf,nd),

Reviewed(nd)∧*Approved*(nd)→*FinalDesign*(d,fd),

DesignReviewFeedback(drf,nd)→$\mathcal{P}os$(*Approved*(nd))∧$\mathcal{P}os$(¬*Approved*(nd))

Invariants: ¬*ChangeVersion*(nd)

Imp_{st_2} =SUBPROCESS

步骤 st_2 描述了对设计文档的评审.

st_3=(CRM_{st_3} , Res_{st_3} , Imp_{st_3}),

其中 CRM_{st_3} :

Preconditions: ¬*Approved*(nd)

Postconditions: *Design*(d)∧*NewDesign*(d,nd)∧¬*Reviewed*(nd)

Invariants: ¬*ChangeVersion*(nd)

Imp_{st_3} =SUBPROCESS

步骤 st_3 描述了对于评审未通过的设计文档需要进行进一步的修改.

包含上述过程知识的过程 Agent *Change_Design* 可以被认为对应于一个负责设计变更的一线经理的角色,它知道如何根据需求变更的要求而最终得到一个变更了的设计文档.可以注意到,*Change_Design* 的所有过程步骤都被描述为子过程,即过程 Agent *Change_Design* 并不了解每个步骤应该如何被实现.事实上,对于每个步骤的实现被分别封装在过程 Agent *Design*,*Review_Design* 和 *Correct_Design* 中,这些过程 Agent 通过建立管理网络实现彼此的协同.

过程 Agent *Design*,*Review_Design* 和 *Correct_Design* 对应的过程步骤 st_4,st_5 和 st_6,其控制规则模型分别与过程 Agent *Change_Design* 的过程步骤 st_1,st_2 和 st_3 的控制规则模型相类似,所不同的是,过程 Agent *Design*,*Review_Design* 和 *Correct_Design* 的过程步骤知识中还描述了实现这些步骤的方法.

在下面的小节中,我们将通过过程实施模型讨论该过程的实现以及该过程如何自治地适应环境的变化.

3.1 软件过程及管理网络的建立

首先,对于目标 g,过程 Agent pa_{Change_Design} 通过对初始状态的观察及其过程知识建立了 $\mathcal{P}os_{paChange\_Design}$*FinalDesign*($d$,$fd$) 的信念,同时由于 $GC \le Res_{paChange\_Design}$,因而该 Agent 形成了愿望 $\mathcal{D}_{paChange\_Design}$={$g$}.针对愿望 $\mathcal{D}_{paChange\_Design}$,在过程知识和信念的基础上,该过程 Agent 生成了规划 $\mathcal{H}_{paChange\_Design}$={{$st_1$}{$st_2$}},亦即 *Change_Design* 意图通过先实施一个设计变更步骤然后再实施一个设计评审

步骤来实现目标 g.

进一步地,过程 Agent *Change_Design* 发现 st_1 和 st_2 这两个步骤均被定义为由子过程实现,因此该过程 Agent 先后发布两个子目标 $pa_{Change\_Design} \Rightarrow g'$,$pa_{Change\_Design} \Rightarrow g''$,其中 $g'=(\{Design(d) \wedge NewDesign(d,nd) \wedge \neg Reviewed(nd)\}, Res_{st_1})$,$g''=(\{Reviewed(nd) \wedge DesignReviewFeedback(drf,nd)\}, Res_{st_2})$.这两个子目标进而分别被 pa_{Design} 和 pa_{Review_Design} 作为愿望而予以实现 $\mathcal{D}_{paDesign}=\{g'\}$,$\mathcal{H}_{paDesign}=\{\{st_4\}\}$,$\mathcal{D}_{paReview\_Design}=\{g''\}$,$\mathcal{H}_{paReview\_Design}=\{\{st_5\}\}$.这样,过程 Agent pa_{Design} 和 pa_{Review_Design} 作为 pa_{Change_Design} 的下属而形成了管理网络 $MN=(S_{MN},G,PA_{MN},R_{MN})$,其中 $PA_{MN}=\{pa_{Change\_Design},pa_{Design},pa_{Review\_Design}\}$,$R_{MN}=\{(pa_{Change\_Design},pa_{Design}),(pa_{Change\_Design},pa_{Review\_Design})\}$.

图 5 给出了在我们目前开发的实验平台上,上述的过程 Agent 建立软件过程和管理网络的情况.

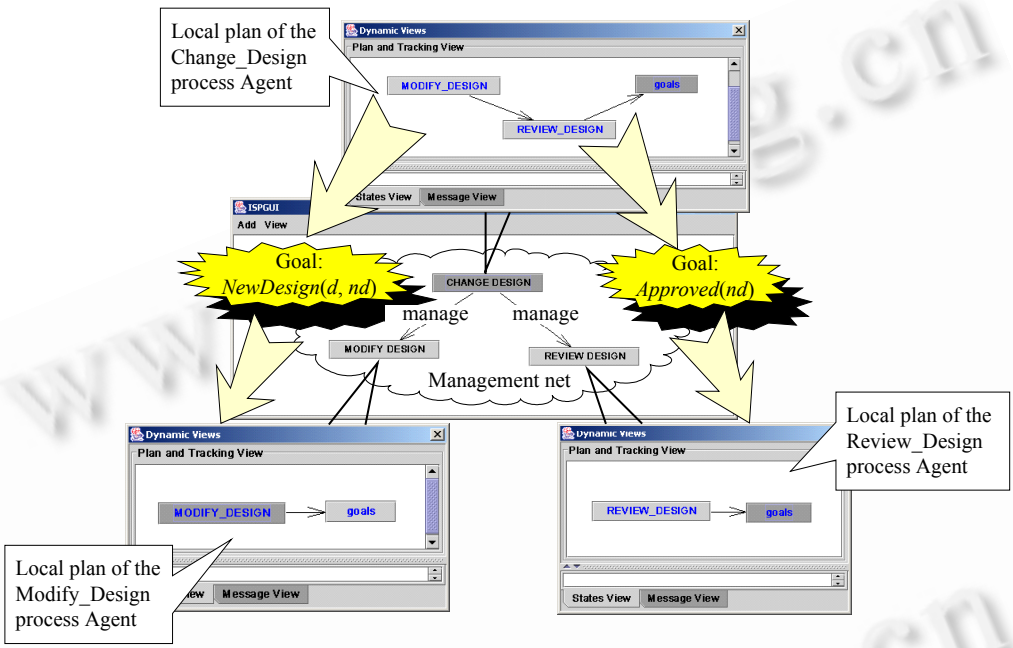

Fig.5 The setup of the software process and the management net

图 5 软件过程及管理网络的建立

3.2 对环境变化的自适应

本节我们将通过一个例子来介绍过程 Agent 如何在过程环境发生非预期变化时,自适应地对软件过程及管理网络进行调整.

前面建立的管理网络 MN 对应了这样一个过程,即通过实施一个设计活动以及一个后续的设计评审活动能够实现设计的变更.形成这样的结果是由于 pa_{Change_Design} 形成了 $\{\{st_1\}\{st_2\}\}$ 的规划,理由是其开销最小.事实上,这里隐含了一种异常的情况,即修改后的设计文档未能通过设计评审.实验中,我们使得设计评审活动产生一个评审未通过的结果.在这种情况下,上述过程无法实现其预期的目标.通过对环境的观察,过程 Agent pa_{Change_Design} 的信念会发生变化,亦即有 $Bel_{paChange\_Design} \neg Approved(nd)$.通过其反射机制,过程 Agent pa_{Change_Design} 会自治地根据这个非预期的状态重新进行规划,并最终形成 $\mathcal{H}'_{paChange\_Design}=\{\{st_3\}\{st_2\}\}$,亦即针对未通过设计评审的设计文档,根据评审意见进行修改,然后再次进行设计评审.进一步地,管理网络的结构也因此发生变化,$MN'=(S'_{MN},G',PA'_{MN},R'_{MN})$,其中 $PA'_{MN}=\{pa_{Change\_Design},pa_{Correct\_Design},pa_{Review\_Design}\}$,$R'_{MN}=\{(pa_{Change\_Design},pa_{Correct\_Design}),(pa_{Change\_Design},pa_{Review\_Design})\}$.

图 6 显示了在我们的实验平台上,过程 Agent 对上述变化进行处理的情况.可以看出,过程 Agent *Change_Design* 变更了其本地规划,相应地,管理网络的结构也发生了变化(加入了过程 Agent *Correct_Design*).

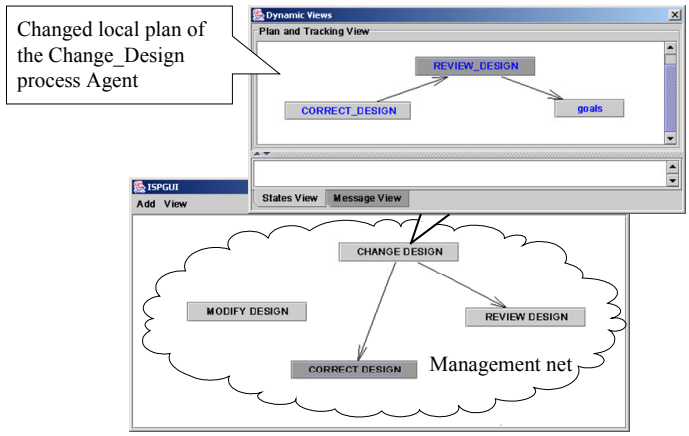

Fig.6 Self-Adaptation to the change of the environment

图 6 对环境变化的自适应

4 结论及进一步的工作

本文中,我们形式化地给出了一种基于 Agent 的自适应软件过程模型,目的在于提供一个一般的可计算模型来描述这种自适应软件过程的定义和实现.在实践中,我们基于过程描述模型开发了一种自适应软件过程建模语言.应用该语言,软件过程建模人员能够通过描述一组过程 Agent 来建立自适应软件过程模型.另一方面,实施模型描述了软件过程 Agent 基本的实现方法,在此基础上,我们开发了一个基于自适应软件过程模型的 PSEE(process-centered software engineering environment)原型系统.该系统能够通过载入用户定制的过程 Agent 描述而自动生成过程 Agent 的实例,并实现自适应的软件过程控制.

该研究进一步的工作主要集中在过程建模的自然化、可视化方面.在我们目前开发的过程建模语言中,过程知识的描述是基于自认知逻辑的,对于用户来讲过于复杂.我们希望能够通过结构化的自然语言以及图形的方式为用户提供一种简单、直观的模型描述方法.同时,我们也正致力于 PSEE 原型系统的产品化研究工作,力求使该系统能够应用于软件工程实践.

致谢 本课题由中国科学院软件研究所互联网软件技术实验室和香港理工大学计算学系协同开展,并得到香港理工大学计算学系 A Multi-Site Software Development Environment 项目的支持,在此一并表示感谢.

References:

[1] Finkelstein A, Kramer K, Nuseibeh B. Software Process Modeling and Technology. Taunton: Research Studies Press Ltd., 1994.

[2] Engels G, Groenewegen L. SOCCA: Specification of coordinated and cooperative activities. In: Software Process Modeling and Technology. Taunton: Research Studies Press Ltd., 1994. 71~100.

[3] Bandinelli S, Fuggetta A, Lavazza L, Loi M, Picco G. Modeling and improving an industrial software process. IEEE Trans. on Software Engineering, 1995,21(5):440~454.

[4] Weiss G. MultiAgent System: A Modern Approach to Distributed Artificial Intelligence. Cambridge: The MIT Press, 1999.

[5] Shepherdson JW, Thompson SG, Odgers BR. Cross organisational workflow coordination by software Agents. In: Bussler C, Grefen P, Ludwig H, Shan M, eds. Proc. of the Workshop on Cross-Organisational Workflow Management and Coordination (WACC'99). 1999. http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-17/

[6] Wang AI. A process centered environment for cooperative software engineering. In: Chang SK, ed. Proc. of the 14th Int'l Conf. on Software engineering and knowledge engineering. Ischia: ACM, 2002. 457~468.

[7] Zeng LZ, Ngu A, Benatallah B, Dell M. An Agent-based approach for supporting cross-enterprise workflows. In: Orlowska M, Roddick J, eds. Proc. of the Australiasian Database Conf. Gold Coast: IEEE Press, 2001. 123~130.

[8] Gou HM, Huang BQ, Liu WH, Li Y, Ren SJ. Agent-Based virtual enterprise modeling and operation control. In: Bahill T, Wang FY, eds. Proc. of the IEEE Int'l Conf. on Systems, Man, and Cybernetics. Tucson: IEEE Press, 2001. 2058~2064.

[9] Kellner MI, Feiler PH, Finkelstein A, Katayama T, Osterweil LJ, Penedo MH, Rombach HD. ISPW-6 software process example. In: Katayama T, ed. Proc. of the 6th Int'l Software Process Workshop: Support for the Software Process. IEEE Computer Society Press, 1991. http://www.idt.unit.no/~epos/ispw6.html

[10] Moore R. Autoepistemic Logic. In: Smets P, Mamdani A, Dubois D, Prade H. eds. Non-Standard Logics for Automated Reasoning. London: Academic Press, 1988. 105~132.

[11] Shi CY, Huang CN, Wang JQ. Principles of Artificial Intelligence. Beijing: Tsinghua University Press, 1993 (in Chinese).

[12] Smith RG. The contract-net protocol: High-Level communication and control in a distributed problem solver. IEEE Trans. on Computers, 1980,C-29(12):1104~1113.

附中文参考文献:

[11] 石纯一,黄昌宁,王家钦.人工智能原理.北京:清华大学出版社,1993.

2004 年全国理论计算机科学学术年会

征 文 通 知

由中国计算机学会理论计算机科学专业委员会主办，海军工程大学信息与电气学院承办的“2004 年全国理论计算机科学学术年会”将于 2004 年 10 月在武汉召开。会议录用论文将收录在正式出版的论文集中，欢迎大家积极投稿。现将有关征文要求通知如下：

1．应征论文应未在其他刊物或学术会议上正式发表过。特别欢迎有创见的论文和有应用前景的论文。

2．稿件要求用计算机打印，格式为 38 行×38 字，字体为 5 号宋体。稿件中的图形要求画得工整、清晰、紧凑，尺寸要尽量小；图中字体要求为六号宋体。稿件正文不超过六千字。标题、作者姓名、作者单位、摘要、关键词采用中英文间隔行文。稿件各部分依次为：一、引言；二、...；最后是结束语。附录放在参考文献之后；参考文献限已公开发表的，文中最好不要出现文献序号。参考文献的格式为：

序号 作者・书名・出版社所在地：出版社名，出版年代

序号 作者・论文名・出处，年代・卷号（期号）：起迄页码

务必附上第一作者简历（姓名、性别、出生年月、职称、学位、研究方向等）、通信地址和联系电话。并注明论文所属领域。请提供打印稿和电子稿各一份。来稿一律不退，请自留底稿。

3．征文范围

程序理论（程序逻辑、程序正确性验证、形式开发方法等）；计算理论（算法设计与分析、复杂性理论、可计算性理论等）；语言理论（形式语言理论、自动机理论、形式语义学、计算语言学等）；人工智能（知识工程、机器学习、模式识别、机器人等）；逻辑基础（数理逻辑、多值逻辑、模糊逻辑、模态逻辑、直觉主义逻辑、组合逻辑等）；数据理论（演绎数据库、关系数据库、面向对象数据库等）；计算机数学（符号计算、数学定理证明、计算几何等）；并行算法（分布式并行算法、大规模并行算法、演化算法等）。

4．征文截止日期：2004 年 5 月 1 日

5．论文投寄地址：（430033）武汉 海军工程大学信息与电气学院 张志祥 收

联系电话：027-83443985,83443984（张志祥，贲可荣）

电子信箱：tcs2004@vip.sina.com; hgzzx@163.com