ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.19, No.3, March 2008, pp.533−544
DOI: 10.3724/SP.J.1001.2008.00533

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

一种基于组织实体能力的软件过程建模方法*

肖俊超[1,2], 王 青[1+], 李明树[1,3], 张 镭[1,2], 刘大鹏[1,2]

[1](中国科学院 软件研究所 互联网软件技术实验室,北京 100080)

[2](中国科学院 研究生院,北京 100049)

[3](中国科学院 软件研究所 计算机科学重点实验室,北京 100080)

An Organization-Entity Capability Based Software Process Modeling Method

XIAO Jun-Chao[1,2], WANG Qing[1+], LI Ming-Shu[1,3], ZHANG Lei[1,2], LIU Da-Peng[1,2]

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

[2](Graduate University, The Chinese Academy of Sciences, Beijing 100049, China)

[3](Key Laboratory for Computer Science, Institute of Software, The Chinese Academy of Sciences, Beijing 100080, China)

+ Corresponding author: Phn: +86-10-62661800, Fax: +86-10-62661817, E-mail: wq@itechs.iscas.ac.cn, http://www.cnsqa.com

Xiao JC, Wang Q, Li MS, Zhang L, Liu DP. An organization-entity capability based software process modeling method. *Journal of Software*, 2008,19(3):533−544. http://www.jos.org.cn/1000-9825/19/533.htm

Abstract: This paper presents an organizational entity capabilities based software process modeling method (OEC-SPM), aiming at the particularities of software process, it defined the organizational entity with certain capabilities as the core element and the basic unit in the modeling process—process-agent. Process-Agents produce concrete software development processes and production processes according to the goals, the knowledge, the experiences and the capabilities under the defined project goals and constraint environment, through the proactive as well as autonomous reasoning on behaviors to provide effective supports and proper decisions to software project development. The method, owing to its full consideration on capabilities of process executors accomplishing goals when the software processe is being constructed, makes the established processes have good predictability and satisfy the premises for processes' stable executions, resolving the problems of instability and uncontrollability of the software process radically.

Key words: process-agent; organization-entity; capability; process modeling

摘 要: 提出了一种基于组织实体能力的软件过程建模方法(organizational entity capabilities based software process modeling method,简称OEC-SPM),针对软件过程的特殊性,将具有确定能力的组织实体定义为建模过程中的核心要素和基本单元——过程 Agent(过程主体).过程 Agent 根据其自身的目标、知识、经验和能力,在确定的项目目标和约束环境下,通过主动和自治的行为推理,生成具体的软件开发过程和生产过程,为软件项目开发提供正确的决策和有效的支持.该方法由于在建立软件过程时充分考虑过程执行者完成目标的能力,使建立的过程具有良好的可预见性,具备过程稳定执行的前提,因此从根本上解决了软件过程不稳定、难以控制等问题.

* Supported by the National Natural Science Foundation of China under Grant Nos.60473060, 60573082 (国家自然科学基金); the National High-Tech Research and Development Plan of China under Grant No.2006AA01Z185 (国家高技术研究发展计划(863))

Received 2006-03-31; Accepted 2006-11-13

关键词: 过程 Agent;组织实体;能力;过程建模
中图法分类号: TP301 文献标识码: A

软件过程建模的目的是利用适当的建模方法与工具建立和描述软件过程模型,并在特定过程环境中将软件过程模型实例化为实现特定开发目标的软件过程,从而为软件组织实现以过程为中心的软件开发管理提供有力支持,对于软件组织保证软件产品质量,提高开发效率具有重要的理论和实践价值.软件开发是特殊的生产过程,它高度依赖人的能力,同样的过程由不同的执行者执行,会生产不同质量和数量的产品.因此,有效的软件开发过程以及软件过程建模方法不仅需要考虑过程要素以及它们之间的关系,还必须考虑如下 3 方面的问题:

(1) 过程能够做什么,即过程能够实现什么目标.

(2) 过程如何工作.

(3) 过程的执行需要多少资源,例如,需要多少资金、时间、人力等等.

解决上述问题的关键是在建模软件过程时,将软件开发人员的能力作为关键要素予以考虑,从而合理地调度资源,使其按照既定的目标和约束条件进行有效配置.在文献[1]中,中国科学院软件研究所给出了一个针对软件开发全过程的解决方案并实现了一个软件过程管理集成平台 SoftPM.该平台是一个开放的框架,包含一种基于 Agent 的自适应软件过程模型[2,3],以解决软件过程环境易变性带来的问题.

本文在上述工作的基础上,给出一种基于组织实体能力的软件过程建模方法(organization-entity capability based software process modeling method,简称 OEC-SPM).OEC-SPM 应用了 Agent 技术,将一个具有确定资源能力(确定的目标、技能、知识、生产率、经验、历史数据记录、设备等)的组织实体定义为过程的基本单元——过程 Agent(过程主体).基于过程 Agent,OEC-SPM 以组织实体的能力作为决定因素确定软件过程能够实现的目标,以及如何实现、需要多少资源等核心问题,因此,这种建模方法能够更好地反映软件开发过程的本质.

本文第 1 节介绍现有软件过程建模方法和 Agent 在过程建模中的应用及不足.第 2 节给出基于组织实体能力的软件过程建模方法的框架.第 3 节介绍建模方法的核心——过程 Agent 的构成.在第 4 节说明 OEC-SPM 的建模过程.最后是结束语.

1 相关工作

软件过程所涉及的要素很多,要素之间的交互和约束关系也很复杂.软件过程建模方法通常将建模所关注的焦点集中在某一个要素上,并以该要素为中心建立整个软件过程模型.由于建模的焦点不同,软件过程模型的结构也不尽相同,例如,基于活动的方法[4]以活动为模型的核心要素,其他过程要素与活动相关联,从而将软件过程模型描述为一系列活动或步骤的偏序集.类似地有基于制品的方法[5]以及基于角色的方法[6].这些方法通常仅关注于过程和活动之间的关系,人被作为过程的执行者或者过程资源而被动地在预定义的模式下实施软件开发活动,所建立过程的能力由设备能力所决定,只考虑过程能够做什么以及目标如何实现,很少考虑实现目标所需的资源.由此建立的软件过程模型由于对影响软件过程的主要因素:资源能力,尤其是人力资源的能力考虑不足,导致了过程执行的不稳定和不可预测,从而失去了过程建模的重要意义.针对软件生产的特性,一些研究已经开始关注软件过程中工程师和小组的能力.例如,SEI 提出了 PSP[7]和 TSP[8],用以改进软件工程师个人和小组的性能和能力,以生产高质量的产品,从而帮助这些组织改进基于 CMMI[9]的软件过程.

另有一些研究将 Agent 技术应用在过程建模中.Agent 技术起源于分布式人工智能(distributed artificial intelligence,简称 DAI)研究领域[10].由于 Agent 具有自治性、主动性、交互性、社会能力等特性[11],对 Agent 的使用能够支持过程的控制[12]、分布式协同[13]、业务过程的自动化[14]以及表示过程模型灵活的组织结构[15].然而,目前 Agent 大多应用在业务过程的建模中,虽然它们关注了软件过程的部分关键要素及特性,但并没有很好地考虑组织实体的能力及资源的分配问题.

针对上述方法的不足,本文提出了基于组织实体能力的软件过程建模方法.这种方法突破了传统方法中只以过程的实体角色关系来确定过程模型的思想,通过应用 Agent 技术将过程实体的能力纳入过程建模中,力求

解决软件过程中执行者能力极大地影响过程能力的问题.

2 建模方法框架

基于组织实体能力的软件过程建模方法(OEC-SPM)在描述过程的组成要素及相互关系的基础上增加了对组织实体能力的描述,从而在建模中不仅考虑了过程能够做什么以及如何做的问题,还特别考虑了过程完成工作所需的资源,因而它不但使过程具有良好的可预见性,还使过程具备稳定执行的前提.OEC-SPM 的实现包含了一组过程目标、这些目标的约束条件、目标上下文环境中的知识及一组过程 Agent,这些元素放在过程环境中,由过程 Agent 主动地感知目标,并根据自身的能力和过程环境知识自治地完成建模.OEC-SPM 的框架可被描述为一个三元组:$OEC\text{-}SPM=(G,EK,PA)$.这里:

(1) G 代表特定约束条件下的环境目标集合,$G=\{g_1,g_2,\ldots,g_n\}$,其中,每个目标 g_i 被定义为一个四元组,

$$g_i=(gs_i,gse_i,gc_i,gk_i).$$

a) gs_i 是目标的描述(goal statement).它通过一个字符串描述目标是什么.

b) gse_i 是对目标规模的估计(goal size estimation).

c) gc_i 是实现目标 g_i 的约束条件集合,如进度约束(time constraint,简称 TC)、成本约束(cost constraint,简称 CC)、质量约束(quality constraint,简称 QC)、语言约束(language constraint)等.

d) gk_i 是实现目标所需的技能集合.技能具有类别和取值,例如,技能类别 Language 描述的是过程 Agent 在编程语言方面的技能,技能类别 Domain 描述的是过程 Agent 在应用领域方面的技能.若技能类别是 Language,则其技能取值可以为 C++和 Java 等,表示过程 Agent 具有 C++或 Java 等的语言技能.设共有 m 类技能 $\{k_1,k_2,\ldots,k_m\}$,对于每类技能,D_i 是 k_i 的取值集合,则 $gk_i=\{c_1,\ldots, c_t\}$,其中:

$$c_j=(k_{j1}=x_{j1})\wedge(k_{j2}=x_{j2})\wedge\ldots\wedge(k_{js}=x_{js}),$$

这里,$1\le j1<j2<\ldots<js\le m, x_{j1}\in D_{j1}, x_{j2}\in D_{j2},\ldots, x_{js}\in D_{js}$.

(2) EK 是过程环境的知识集合,它由一些事实构成,作为过程 Agent 实现目标的基础及认识世界的前提.

(3) PA 是过程环境中包含的一组具有特定能力的过程 Agent,$PA=\{pa_1,pa_2,\ldots,pa_n\}$,其中,每个 pa 可以被定义为一个二元组,$pa=(PAI,PAE)$.这里:

a) PAI 是过程 Agent 的知识结构(process-agent infrastructure),包括对过程 Agent 的特征、各种过程要素知识和经验知识的描述.PAI 是确定过程 Agent 能力的基础.我们根据过程 Agent 在确定其能做什么、如何做以及需要多少资源时所需的不同知识,将知识结构定义为一个三元组:$PAI=(DK,PK,EL)$,即描述性知识(descriptive knowledge,简称 DK)、过程性知识(process knowledge,简称 PK)和经验库(experience library,简称 EL).

b) PAE 是过程 Agent 的行为引擎(process-agent engine),由过程 Agent 实现目标驱动自治行为的功能模块构成.根据行为引擎的工作机制,将它定义为一个六元组:$PAE=(P,R,EE,RE,NE,LE)$,即感知器(perceptor,简称 P)、效应器(reactor,简称 R)、实施引擎(enactment engine,简称 EE)、推理引擎(reasoning engine,简称 RE)、协商引擎(negotiation engine,简称 NE)、学习引擎(learning engine,简称 LE).

例如,我们要实现一个软件开发项目 WebProject,它有一个模块设计和编码的子目标 g,在过程环境中有一些输入,例如需求文档,并有一组意图实现目标的过程 Agent,则相应的软件过程模型如下:

(1) $g=(gs,gse,gc,gk)$.这里,

a) gs="Design&Code",即目标是为一个模块进行设计和编码;

b) gse="24KLOC",即模块的估计规模是 24KLOC;

c) $gc=\{TC=[2006\text{-}04\text{-}01;2006\text{-}06\text{-}30]\}$,这里,

- TC 是目标的时间约束,表示目标必须在 2006-04-01~2006-06-30 之间的 3 个月时间内实现;

d) $gk=\{(Domain=\text{"Web"}),(Language=\text{"Java"})\}$,

- Domain 要求实现目标的过程 Agent 应该具有 Web 领域的技能;
- Language 要求实现目标的过程 Agent 应该能够使用 Java 语言开发.

(2) $EK=\{ReqDoc(g)\}$,表示在过程环境中有一个为实现目标 g 的需求文档.

(3) $PA=\{pa_1,pa_2,\ldots,pa_n\}$,表示在过程环境中有 n 个过程 Agent:$pa_1,pa_2,\ldots,pa_n$,其中,每个过程 Agent 都具有定义中所描述的知识结构与行为引擎,并能在上述目标的驱动下主动和自治地进行行为推理并给出实现目标的计划.

当需要建立一个特定环境下的软件过程时,如图 1 所示,每个过程 Agent 都能够主动和自动地感知过程环境,并对环境目标和约束条件、环境知识、变化的需求和约束条件作出反应;过程 Agent 通过自身的智能行为及之间基于协商的协同,自适应地建立实现软件开发目标的项目过程体系.在过程执行时,过程 Agent 将收集过程数据,以优化和改进他们的能力,从而逐步提高建模的稳定性和可预测性.

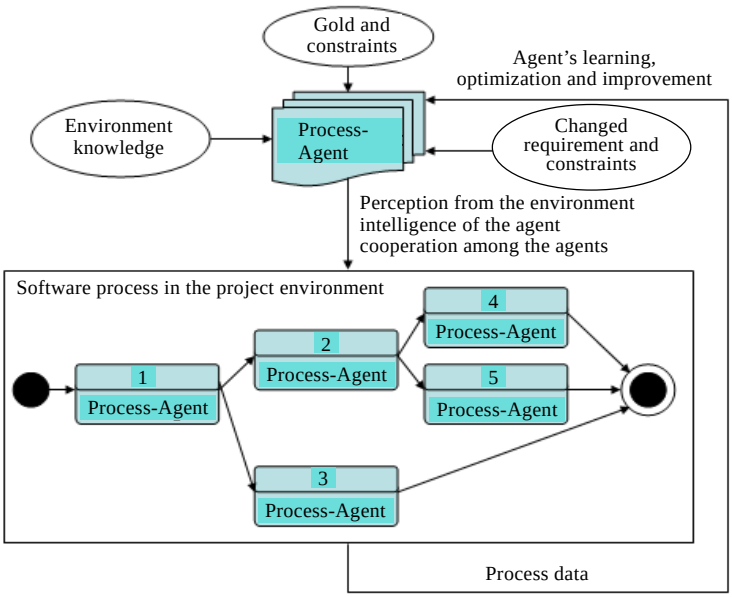


Fig.1 OEC-SPM working process

图 1 OEC-SPM 的工作过程

3 过程 Agent

过程 Agent 是 OEC-SPM 的核心,所以对它进行有效描述以及合理构造是建模的基础.如图 2 所示,我们通过把过程 Agent 分为知识结构(PAI)和行为引擎(PAE)来区分过程 Agent 的知识表述和行为机制.其中,知识结构是过程 Agent 的能力基础,由确定过程 Agent 能力所需的描述性知识、过程性知识和经验库这 3 部分知识构成;行为引擎由完成其目标驱动行为推理的感知器、效应器、实施引擎、推理引擎、协商引擎、学习引擎这 6 部分功能模块构成.本节将对每个部分作详细的描述.

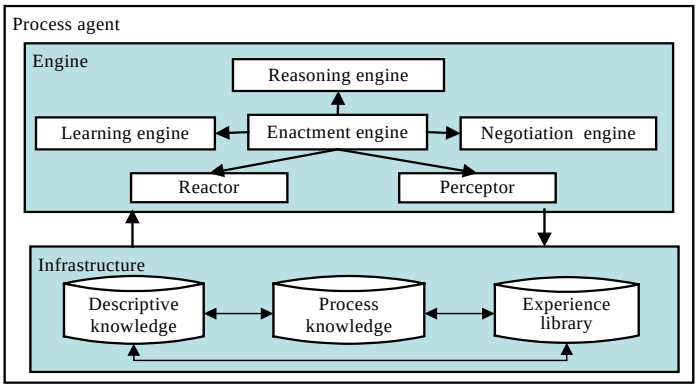


Fig.2 Structure of the process-agent

图 2 过程 Agent 的结构

3.1 描述性知识(DK)

DK 描述一个过程 Agent 能够做什么,被定义为一个六元组:*DK*=(*BA*,*AG*,*CRM*,*SK*,*RC*,*AEG*).

(1) *BA* 是过程 Agent 的基本属性(basic attribute)元组,*BA*=(*EN*,*PAN*,*Description*),其中,*EN* 是过程 Agent 所代表的实体的名称,*PAN* 是过程 Agent 的名称,*Description* 是对过程 Agent 特征的描述.

(2) *AG* 由一个目标字符串构成,它描述了过程 Agent 的目标(agent goal),即过程 Agent 能够实现什么样的目标、做什么工作.

(3) *CRM* 是过程 Agent 的控制规则模型(control rule model),包括前置条件(pre-condition)和后置条件(post-condition),这些前置/后置条件可以是制品、角色等软件过程要素的约束描述.只有满足了前置条件,过程 Agent 才可以实施达到目标的活动步骤;只有满足了后置条件,过程 Agent 才能成功地完成活动步骤并达到目标.因此,控制规则模型控制了过程 Agent 的行为.

(4) *SK* 是过程 Agent 拥有技能(skill)的描述集合,设共有 *m* 类技能,则 $SK=\{sk_1,sk_2,\ldots,sk_m\}$,其中,$sk_i(1\le i\le m)$是每类技能的取值集合,$sk_i\subseteq D_i$.例如,若第 *i* 类技能是语言技能,则 sk_i 的取值可以是{C++,Java}.

(5) *RC* 是过程 Agent 自身的资源约束(resource constraint),即过程 Agent 拥有的可调用资源.过程 Agent 不能实施资源约束之外的行为.*RC* 可以是不同类别的约束,例如时间约束(time resource,简称 tr)、人力资源约束(human resource,简称 hr)、工具约束(tool resource,简称 tlr)等等.

(6) *AEG* 是过程 Agent 基于目标的估算(estimate on the goal),它是对完成给定目标的各种指标的估算,如时间(T)、工作量(E)、人力资源(HR)、成本(C)和质量(Q).这些估算可以从给定目标和过程 Agent 的经验库得出.

例如,针对第 2 节所述的模块设计和编码目标 *g*,假定有 3 个相关过程 Agent,分别具有不同的属性、资源和技能,这 3 个过程 Agent 的描述性知识见表 1.

这里,我们采用人数表示人力资源 hr,它的单位是 P(表示人),AEG 由过程 Agent 感知到的过程环境目标和过程 Agent 自身的经验库共同确定.

3.2 过程性知识(PK)

PK 描述过程 Agent 实现其目标的过程.过程 Agent 根据它在 DK 中描述的能力确定能够做什么,如果过程 Agent 决定实现某个目标,它将通过过程性知识确定如何实现这个目标.PK 由一组能被过程 Agent 稳定实施的过程步骤组成,这些过程步骤是过程 Agent 完成历史项目中的目标时,对所实施的活动在知识层次上的抽象描述,$PK=\{st_1,st_2,\ldots,st_n\}$.其中每个过程步骤 st_i 定义为一个六元组,$st_i=(SID_i,SD_i,SCRM_i,IP_i,OP_i,SEG_i)$,这里,

(1) SID_i 是过程步骤的唯一标识.

(2) SD_i 是过程步骤的描述信息,它由描述该步骤功能的字符串构成.

(3) $SCRM_i$ 是过程步骤的控制规则模型,它包括了过程步骤的前置条件和后置条件,这些前置/后置条件可

以是对制品、角色等软件过程要素的约束.控制规则模型决定了软件过程组成要素之间的相互关系,它描述的是过程 Agent 如何通过过程步骤的组合及执行,从一个初始状态转换到目标状态.

(4) IP_i 是过程步骤的输入参数,如步骤执行需要的制品.

(5) OP_i 是过程步骤的输出参数,如步骤执行产生的制品.

(6) SEG_i 是过程步骤的资源估算,它是为完成给定目标,对执行该步骤的各种度量指标的估算,如时间(T)、工作量(E)、人力资源(HR)、成本(C)和质量(Q).这些估算可以由给定目标和过程 Agent 的经验库得出.

Table 1 Descriptive knowledge of process-agents

表 1 过程 Agent 的描述性知识

DK\Process agent		pa_1	pa_2	pa_3
BA	*EN*	*WebTeam*1	*CompilerTeam*	*WebTeam*2
	PAN	PA_1	PA_2	PA_3
	Description	Web application development team 1	Compiler development team	Web application development team 2
AG		Design & Code	Design & Code	Design & Code
CRM	PreC	$\exists ReqDoc(x)\wedge(x\in G)$	$\exists ReqDoc(x)\wedge(x\in G)$	$\exists ReqDoc(x)\wedge(x\in G)$
	PostC	$Code(x)\wedge(x\in G)$	$Code(x)\wedge(x\in G)$	$Code(x)\wedge(x\in G)$
SK		(Domain="Web") (Language="Java")	(Domain="Compiler") (Language="Java")	(Domain="Web") (Language="Java")
RC	*Tr*	[06-03-01,06-03-31]	[06-02-10,06-06-10]	[06-03-01,06-06-30]
	Hr (*P*)	4	3	4
AEG		—	—	—

例子中的 3 个过程 Agent 在实现目标时都采用设计和编码两个步骤,它们是顺序关系.根据上面说明,这些过程 Agent 的过程性知识的描述见表 2.

Table 2 Process knowledge of process-agents

表 2 过程 Agent 的过程性知识

SID		Design	Coding
SD		Design	Coding
CRM	PreC	$\exists ReqDoc(x)\wedge(x\in G)$	$\exists DesignDoc(x)\wedge(x\in G)$
	PostC	$DesignDoc(x)$	$Code(x)\wedge(x\in G)$
IP		$ReqDoc(x)$	$DesignDoc(x)$
OP		$DesignDoc(x)$	$Code(x)$
SEG		—	—

这里,*SEG* 将在过程 Agent 确定了要实现的意图后,由过程性知识和经验库共同得出.

3.3 经验库(EL)

EL 是过程 Agent 估算其实现目标所需要的资源、判断过程 Agent 自身能力的基础.EL 由过程 Agent 的历史经验数据记录构成,包括实现过程的时间、成本、缺陷等.基于 EL 中的经验数据,过程 Agent 能够建立它的过程性能基线,计算出基线生产率,并估算出执行某个过程步骤需要的时间、成本、质量以及过程步骤之间的资源消耗比率等.EL 中的经验数据记录是一个可扩展的元组 HD,例如,若希望通过历史数据得到过程 Agent 各个步骤的基线生产率和步骤间的资源消耗比率,可将 HD 定义为 *HD*=(*SID*,*PID*,*Size*,*PlanTime*,*ActualTime*),其中:

(1) *SID* 是步骤的标识,它是历史数据记录所对应 *PK* 中的过程步骤的标识;

(2) *PID* 是过程 Agent 执行步骤时所在的历史项目的标识;

(3) *Size* 表示目标的规模,如一个代码开发的目标可以用代码行数来表示;

(4) *PlanTime* 是步骤的计划执行时间;

(5) *ActualTime* 是步骤的实际执行时间.

如果希望得到成本和质量数据,则需要为 HD 加入步骤执行时计划和实际的成本以及计划和实际的缺陷数据.根据上面描述的经验数据知识,可以计算例子中的过程 Agent 实现目标的基线生产率及步骤间的资源使用比例,见表 3 和表 4.表 3 中,生产率的单位 KLOC/P-M 表示千代码行每人月.

Table 3 Baseline productivity of the process-agent

表 3 过程 Agent 的基线生产率

Process agent	pa_1	pa_2	pa_3
Productivity (KLOC/P-M)	3.5	2.2	3.0

Table 4 Resource ratio of each process-agent step

表 4 过程 Agent 实现过程的步骤资源占用比例

Process agent	pa_1		pa_2		pa_3	
Resource ratio (%)	Design	Code	Design	Code	Design	Code
	35	65	50	50	40	60

构成过程 Agent 知识结构的 3 个部分紧密关联,缺一不可.其中,描述性知识的内容由过程性知识和经验库所决定,同时又是这两部分内容的抽象.例如,描述性知识、过程知识及经验库之间存在如下关系:

- *DK.AG=fg*(*PK*),即过程 Agent 的目标 *AG* 是由过程性知识 *PK* 所决定的,这里,*fg* 是 *PK* 到 *AG* 的决策函数;
- *DK.AEG=fe*(*EL*,*G*),即描述性知识 *DK* 所描述的目标估算 *AEG* 是由经验库 *EL* 和给定的环境目标 *G* 共同决定,这里,*fe* 是 *EL* 和 *G* 到 *AEG* 的估算函数.
- *PK.SEG=fse*(*EL*,*G*),即过程性知识 *PK* 所描述的步骤资源估算 *SEG* 是由经验库 *EL* 和给定的环境目标 *G* 共同决定,这里,*fse* 是 *EL* 和 *G* 到 *SEG* 的估算函数.

上述 3 个函数给出了知识之间的关系,根据不同的目的和要求有不同的方法和算法.本文在此给出一个开放的结构,它们的具体描述不是本文的重点,因此不作详细叙述.

此外,经验库中的经验数据是在实施 *PK* 所确定的过程时,采集并分析得到的,也是进一步确定过程行为的依据,*PK* 和 *EL* 都是为了完成描述性知识所述的特性而存在,它们共同构成过程 Agent 行为引擎工作的基础.

3.4 行为引擎(PAE)

PAE 根据过程 Agent 的知识结构实现行为推理,它的 6 个部分如下所述:

(1) *P* 是感知器(perceptor),过程 Agent 通过它感知环境中的环境知识、目标和约束条件等;

(2) *R* 是效应器(reactor),过程 Agent 通过它对环境做动作;

(3) *EE* 是实施引擎(enactment engine),它是过程 Agent 行为引擎的核心,*EE* 被用来控制其他部分,以实现过程 Agent 的智能行为;

(4) *RE* 是推理引擎(reasoning engine),负责过程 Agent 的行为推理;

(5) *NE* 是协商引擎(negotiation engine),负责过程 Agent 与目标发出者之间针对目标的协商;

(6) *LE* 是学习引擎(learning engine),过程 Agent 通过它对过程经验进行自学习.

本文采用理性 Agent 的 BDI 模型[16]实现行为推理,将在过程 Agent 使用行为引擎进行推理并确定给定目标的实现计划的过程中产生或扩展如下 3 个辅助集合:

- 过程 Agent 的信念(belief)集合:Belief 是过程 Agent 通过感知器感知过程环境时所认知到的事实集合.在本文考虑的范围内,我们把过程 Agent 感知到的环境知识作为该过程 Agent 的信念集合.
- 过程 Agent 的愿望(desire)集合:过程 Agent 利用推理引擎,根据自身的信念、感知到的目标、相应的约束条件及自身的能力,推理确定自己可实现的目标,这些可实现的目标构成过程 Agent 的愿望集合.
- 过程 Agent 的意图(intention)集合:对于愿望集合里的每个目标,过程 Agent 都会试图通过协商引擎与发出目标的过程 Agent 进行协商以获取该目标的执行权.如果过程 Agent 获得了目标的执行权,则将该目标加入到它的意图集合中.一个环境目标可以存在于多个过程 Agent 的愿望集合中,表示这些过程 Agent 都能实现这个目标.然而,一个环境目标只能存在于一个过程 Agent 的意图集合中,即该目标只能交给一个过程 Agent 来实现.过程 Agent 利用推理引擎对意图中的目标生成计划并通过效应器将该计划反馈到过程环境中.

4 OEC-SPM 建模过程

OEC-SPM 由目标集合 *G*、环境知识 *EK* 以及过程 Agent 集合 *PA* 构成,在实现软件过程建模时,*G* 是建模的目的,即建立实现目标 *G* 的软件过程模型;*EK* 是建模过程中需要的知识;*PA* 是建模的主体,它的构成是为了实现 OEC-SPM 中基于目标的推理,得到目标软件过程.一方面,知识结构是过程 Agent 能力的基础,并作为行为引擎、行为推理的依据,可以确定过程 Agent 能够做什么,如何做,以及需要多少资源;另一方面,通过行为引擎,过程 Agent 实现了目标驱动的自治行为.

OEC-SPM 的建模过程是:首先,目标 *G*、环境知识 *EK* 以及过程 Agent *PA* 输入到过程环境中;接下来,过程 Agent 主动地感知过程环境.当感知到一个目标时,过程 Agent 通过行为推理,判断它是否能够实现该目标.若不能实现该目标,则放弃对这个目标的行为推理,继续感知新的目标;否则,生成实现该目标的计划,并将计划输出到过程环境中,完成一个目标的建模;最后,计划将在 SoftPM 中执行与监控,并将实际执行的情况不断反馈给过程 Agent.上述建模过程的关键是过程 Agent 行为推理的实现,接下来我们将详细讨论行为推理算法.

4.1 行为推理算法

图 3 显示了过程 Agent 的行为推理过程.首先,过程 Agent(*pa*)把对过程环境的感知转化为自身的信念;然后,经过行为推理,把感知到的目标转化为愿望及意图,最终生成计划.我们将该过程描述为行为推理算法,见算法 1.为了算法描述的方便,我们约定“.”符号表示域操作,即 *A.B* 表示 *A* 的 *B* 元素或 *B* 属于 *A*.例如,*pa.Belief* 是过程 Agent *pa* 的 Belief 集合,*DK.AEG* 表示描述性知识 *DK* 的 *AEG* 元素等.

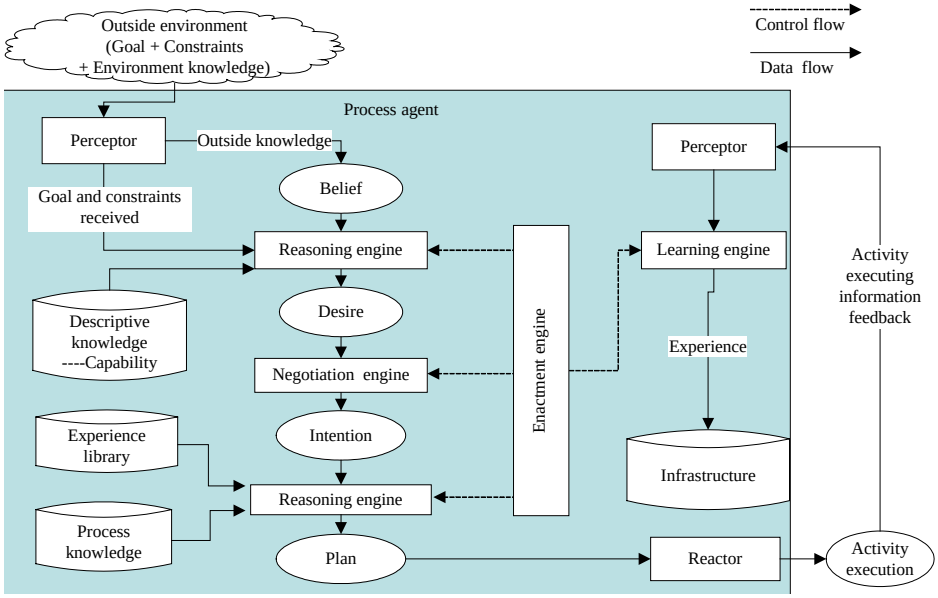


Fig.3 Behavior reasoning process

图 3 行为推理过程

算法 1. 行为推理算法:BehaviorReasoning.

输入:目标 *g*、环境知识 *EK*;

输出:实现目标 *g* 的计划 *Plan*(*g*).

步骤:

(1) 更新 pa 的信念集合:

在对目标进行行为推理之前,pa 通过感知器将环境知识 EK 作为它对环境的感知,更新它的信念集合

pa.Belief=EK.

(2) 生成或扩展 pa 的愿望集合:

基于信念集合,pa 的推理引擎采用如下子步骤推理它是否具有实现给定环境目标 g 的能力,从而获得它的愿望集合:

a) 判断 *pa* 自身目标描述是否满足 *g* 的描述,并且具备实现该目标的前提条件.

For $g \in G$,

If (*g.gs Match DK.AG*) and (*pa.Belief*⇒*DK.CRM.Pre-Condition*) Then *pa* 的目标符合 *g*;

Else EXIT,即取消对 *g* 的进一步推理;

这里,*Match* 是环境目标描述与过程 Agent 目标 *AG* 的匹配函数.

b) 判断 *pa* 的技能是否满足 *g* 的技能约束条件.

令 $g.gk=\{c_1,\ldots,c_t\}$,其中,$c_j=(k_{j1}=x_{j1})\wedge(k_{j2}=x_{j2})\wedge\ldots\wedge(k_{js}=x_{js})$,

这里,$1\le j1<j2<\ldots<js\le m, x_{j1}\in D_{j1}, x_{j2}\in D_{j2},\ldots,x_{js}\in D_{js}$.

If $\underset{1\le i\le m}{\wedge}((sk_i=x_{i1})\vee(sk_i=x_{i2})\vee\ldots\vee(sk_i=x_{iq}))\Rightarrow\underset{1\le j\le t}{\vee}c_j$

Then *pa* 的技能满足 *g* 的技能要求;

Else EXIT.

c) 根据 *g* 的规模和 *pa* 的经验库,得到基于目标的估算 *AEG* 的值.

基于经验库 *EL* 和环境目标的规模,推理引擎使用估算函数 *fe* 得到 *AEG* 的值:

$$DK.AEG=fe(EL,EG).$$

d) 将 *AEG* 与 *pa* 自身的资源约束 *RC* 以及环境目标 *g* 的约束条件加以比较.

根据 *RC* 和 *g*,计算 *pa* 在 *g* 的约束条件下,可用的资源数量,设有一个可用资源(UR)计算函数 *fu*:

$$UR=fu(RC,g.gc),$$

If *UR*>*AEG* Then *pa* 具有实现目标 *g* 的资源;

Else EXIT;

e) 将 *g* 加入到 *pa* 的愿望集合:

经过上述推理,*pa* 具备实现 *g* 的能力,因此将 *g* 加入到 *pa* 的愿望集合中:

$$pa.Desire=pa.Desire+\{g\}.$$

(3) 生成或扩展 *pa* 的意图集合:

pa 通过协商引擎与 *g* 的发出者协商,如果取得 *g* 的执行权,则将 *g* 加入到他的意图集合中:

$$pa.Intention=pa.Intention+\{g\}.$$

(4) 为意图生成计划:

pa 通过推理引擎利用自身过程知识和经验库来生成实现意图 *g* 的局部计划 *Plan*(*g*),计划中的活动由过程知识的部分或全部过程步骤构成,活动之间的关系由步骤间约定的关系确定,同时,根据步骤执行的历史经验,利用步骤资源估算函数 *fse* 为每个活动分配相应的资源.

最后,*Plan*(*g*)通过效应器输出到过程环境中.

算法结束.

算法 1 生成了过程 Agent 实现目标 *g* 的局部计划,如果 *g* 是一个 *G* 的子目标,则该局部计划将被融合到 *G* 的全局计划中.最后,生成的计划在 SoftPM 中被监控和执行,产生的执行信息将反馈给过程 Agent,由学习引擎 *LE* 处理后加入 PAI 中,从而改进过程 Agent 的知识.

4.2 建模过程示例

根据第 2 节、第 3 节的描述,过程环境中有一个设计和编码的目标 *g*、环境知识 *ReqDoc*(*g*)以及 3 个过程 Agent pa_1,Agent pa_2 和 Agent pa_3.过程 Agent 在感知到 *g* 后,针对该目标并结合环境知识,根据算法 1 确定其是否具备实现目标的能力、实现目标的方法以及所需的资源,最终生成实现模块设计和编码目标的实施计划.这

个过程如下:

(1) 更新过程 Agent 的信念集合:

$$pa_1.Belief=pa_2.Belief=pa_3.Belief=EK=\{ReqDoc(g)\}.$$

(2) 生成或扩展过程 Agent 的愿望集合:

基于信念集合,3 个过程 Agent 分别使用自身的推理引擎,采用如下子步骤推理它是否具有实现给定环境目标 g 的能力,从而获得它的愿望集合:

a) 判断过程 Agent 自身目标描述是否满足 g 的描述,并且具备实现该目标的前提条件.

因为对于 pa_1:$DK.AG=g.gs$,

$$(ReqDoc(g)\in pa_1.Belief)\rightarrow DK.CRM.Pre\text{-}Condition,$$

所以 pa_1 具备实现 g 的前提条件.

同理,pa_2 和 pa_3 都具备实现 g 的前提条件.

b) 判断过程 Agent 的技能是否满足 g 的领域和开发语言技能要求.

因为对于 pa_1 和 pa_3,有

$$(Language=Java)\Rightarrow(Language=Java),$$

$$(Domain=Web)\Rightarrow(Domain=Web),$$

所以 pa_1,pa_3 的技能满足目标 g 领域和开发语言技能的要求,因此继续后续的推理.

pa_2 缺少目标要求的领域技能,因此,pa_2 终止对当前目标的行为推理.

c) 根据 g 的规模和过程 Agent 的基线生产率,得到实现给定目标的工作量估算.

因为 pa_1 和 pa_3 都有 4 个人力资源,所以工作量估算如下:

$$pa_1:AEG.E=g.gse/(EL.Productivity\times hr)=24KLOC/(3.5KLOC/P-M\times 4P)=1.7M,$$

$$pa_3:AEG.E=g.gse/(EL.Productivity\times hr)=24KLOC/(3.0KLOC/P-M\times 4P)=2M.$$

d) 将 AEG 与过程 Agent 的时间资源以及环境目标的时间约束比较.

环境目标的时间约束是[2006-04-01,2006-06-30],一共 3 个月.

pa_1 需要 1.7 个月实现目标,然而它在[06-03-01,06-03-31]之间只有 1 个月的可用时间,因此不具有实现目标 g 的时间资源,终止后续对目标 g 的推理.

pa_3 需要 2 个月实现目标,它满足目标的时间资源约束,同时,目标实现所要求的时间恰好在 pa_3 的可用时间区间内,因此, pa_3 满足目标和自身的资源约束条件.

e) 将 g 加入到过程 Agent 的愿望集合.

经过上述推理,pa_3 具备实现 g 的能力,因此,将 g 加入到 pa_3 的愿望集合中,

$$pa_3.Desire=pa_3.Desire+\{g\}.$$

(3) 生成或扩展过程 Agent 的意图集合:

经过上述推理过程,只有 pa_3 满足所有约束条件,在通过协商引擎与 g 的发出者协商取得执行目标 g 的执行权后,将 g 加入到 pa_3 的意图集合中,

$$pa_3.Intention=pa_3.Intention+\{g\}.$$

(4) 过程 Agent 为意图生成计划:

pa_3 实现意图 g 需要设计和编码两个步骤,它们的资源占用比率是 40%:60%,因此,需要的时间资源(SEG)分别估算如下:

Design:40%×2 个月=0.8 个月,

Code:60%×2 个月=1.2 个月.

由于编码步骤 Code 的输入包含设计步骤 Design 的输出:DesignDoc,因此,Design 是 Code 的前驱步骤.根据上述步骤的时间估算以及前驱后继关系,得到 pa_3 实现 g 的局部计划,该局部计划最终被合并到项目 WebProject 的全局计划中.如图 4 所示,计划将在软件过程管理工具 SoftPM 中执行和监控,并将执行情况定期反馈给过程

Agent.

当计划执行发生偏差时,过程 Agent 能够感知到变化的需求及环境约束,并根据它的知识和推理改变决策.同时,实际执行的数据将被 SoftPM 收集并反馈到过程 Agent.SoftPM 将使用这个数据评价并持续改进过程 Agent.

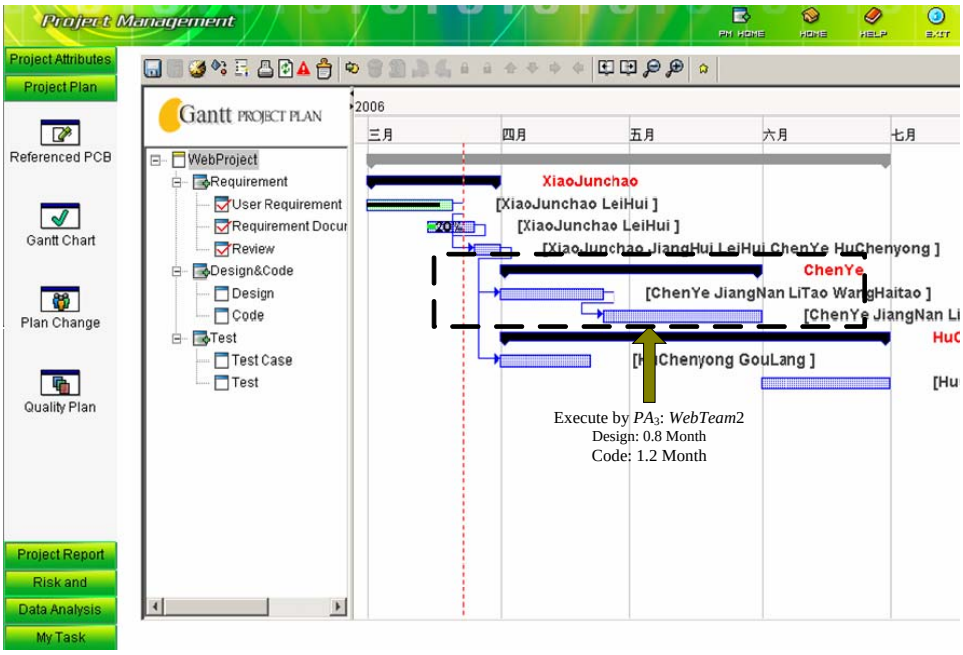


Fig.4 Plan is performed and monitored in SoftPM

图 4 生成的计划在 SoftPM 中执行和监控

5 结束语

本文给出了一种基于组织实体能力的软件过程建模方法.该方法在软件过程建模中充分考虑了执行者能力对过程的影响,将具有确定能力的组织实体定义为过程 Agent.对于每个过程 Agent,知识结构描述了它具有的实现特定目标的能力,行为引擎实现了它基于目标的推理行为.过程 Agent 能够在过程环境的驱动下建立针对特定目标的软件过程体系,并保证由具有相应能力的执行者来执行该过程,从而使建立的过程具有良好的可预见性.该方法保证了过程稳定的前提,为软件项目开发提供了量化的决策和支持.

进一步的工作主要集中于过程 Agent 过程性知识的可视化.我们希望通过图形化的过程描述方法为用户提供直观、易懂的过程性知识表示.

References:

[1] Wang Q, Li MS. Software process management: Practices in China. In: Li MS, Boehm B, Osterweil LJ, eds. LNCS 3840. 2005. 317−331.

[2] 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 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/15/348.htm

[3] Zhao XP, Chan K, Li MS. Applying agent technology to software process modeling and process-centered software engineering environment. In: Proc. of the 20th Annual ACM Symp. on Applied Computing (SAC 2005). ACM Press, 2005. 1529−1533.

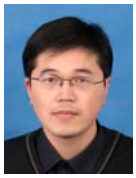
[4] Cass AG, Lerner BS, McCall EK, Osterweil LJ, Sutton SM, Wise A. Little-JIL/Juliette: A process definition language and interpreter. In: Proc. of the 22nd Int'l Conf. on Software Engineering (ICSE 22). ACM Press, 2000. 754−757.

[5] Cugola G, Ghezzi C. Design and implementation of PROSYT: A distributed process support system. In: Proc. of the IEEE 8th Int'l Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprise. IEEE Press, 1999. 32−39.

[6] Briand L, Melo M, Seaman C, Basili V. Characterizing and assessing a large-scale software maintenance organization. In: Proc. of the 17th Int'l Conf. on Software Engineering (ICSE'95). ACM Press, 1995. 133−143.
[7] Humphrey WS. A Discipline for Software Engineering. Massachusetts: Addison-Wesley, 1995.
[8] Humphrey WS. Introduction to the Team Software Process^{SM}. Massachusetts: Addison-Wesley, 2000.
[9] CMMI Product Team. Capability maturity model® integration (CMMI^{SM}). Technical Report, CMU/SEI-2002-TR-0028, Pittsburg, Software Engineering Institute, Carnegie Mellon University, 2002.
[10] Shi ZZ. Intelligent Agent and Its Application. Beijing: Science Press, 2000 (in Chinese).
[11] Zambonelli F, Omicini A. Challenges and research directions in agent-oriented software engineering. Autonomous Agents and Multi-Agents Systems, 2004,9:253−283.
[12] 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/
[13] Wang AI. A process centred environment for cooperative software engineering. In: Chang SK, ed. Proc.of the 14th Int'l Conf. on Software engineering and knowledge engineering. Ischia: ACM Press, 2002. 457−468.
[14] 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.
[15] Glaser N, Derniame JC. Software agents: Process models and user profiles in distributed software development. In: Proc. of the 7th Int'l Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises. Washington: IEEE Computer Society Press, 1998. 45−50.
[16] Rao AS, Georgeff MP. Modeling rational agents within a BDI-architecture. In: Allen J, Fikes R, Sandewall E, eds. Proc. of the 2nd Int'l Conf. on Principles of Knowledge Representation. Morgan Kaufmann Publishers, 1991. 473−484. http://citeseer.ist.psu.edu/rao91modeling.html

附中文参考文献:

[2] 赵欣培,李明树,王青,陈振冲,梁金能.一种基于 Agent 的自适应软件过程模型.软件学报,2004,15(3):348−359. http://www.jos.org.cn/1000-9825/15/348.htm
[10] 史忠植.智能主体及其应用.北京:科学出版社,2000.



肖俊超(1978－),男,吉林珲春人,博士生,主要研究领域为软件过程技术.

张镭(1982－),男,博士生,主要研究领域为软件过程技术.

王青(1964－),女,博士,研究员,博士生导师,主要研究领域为软件过程技术与质量管理.

刘大鹏(1981－),男,博士生,CCF 学生会员,主要研究领域为软件过程技术.

李明树(1966－),男,博士,研究员,博士生导师,CCF 高级会员,主要研究领域为软件过程技术与需求工程.