基于过程模式的软件过程建模*)

梁义芝[1,2] 王延章[1] 刘云飞[3] 缪旭东[2]
(大连理工大学信息与决策技术研究所 大连116024)[1]
(海军大连舰艇学院作战软件研究中心 大连116018)[2]
(海军大连舰艇学院教育技术中心 大连116018)[3]

Process Pattern Based Software Process Modeling

LIANG Yi-Zhi[1,2] WANG Yan-Zhang[1] LIU Yun-Fei[3] MIAO Xu-Dong[2]
(Institute of Information and Decision Technology, Dalian University of Technology, Dalian 116024)[1]
(Combat Software Research Center, Dalian Naval Academy, Dalian 116018)[2]
(Educate Technology Center, Dalian Naval Academy, Dalian 116018)[3]

Abstract Software process patterns can be used to summarize and express software process knowledge, and process pattern based software process modeling can realize the reuse and dynamic improvement of software process knowledge. In this paper, Petri net and UML activity diagrams are utilized to define the formal description method of software process model , and the formal description method is utilized to describe software process patterns, the useful method of utilizing formally described process patterns to model the software process is proposed and verified.

Keywords Software process pattern, Software process knowledge, Software process modeling

1 引言

在对软件产品开发及维护过程的研究中,人们积累了大量的软件过程知识,这些知识可分为两种:一种是可通用于所有软件过程的一般性知识,一种是可应用于某软件组织或某一具体软件项目的领域知识。这两种知识均具有以下特点:

(1)可归纳总结与可表示性:软件过程知识反映了人们对软件过程本质的认识,对知识的归纳总结、构造和表达过程是对知识的更深层次的理解过程,过程知识的形式化表示易于知识的使用与完善;

(2)可指导与可重用性:过程知识可指导过程模型的建立并可指导具体软件项目的实施,过程知识可重用于各种软件开发过程;

(3)动态进化性:过程知识在使用中需不断完善与更新。

软件过程模式提供了软件过程知识的总结与表示方法,基于过程模式的软件过程建模可实现过程知识的重用和动态改进。

本文介绍了基于过程模式建立软件过程模型的方法并对方法进行了验证。

2 软件过程模型及其形式化描述

软件过程是指在软件生存周期中所进行的一系列具有目的性的活动。软件过程模型是对软件过程的抽象描述,即对软件过程实体(一般包括活动、角色、产品)及其关系的抽象描述[1]。过程实体关系反映了过程实体间的相互联系,正是这些联系将各个独立的过程实体组合成一个完整的过程模型。对过程实体关系的分类如图1所示。

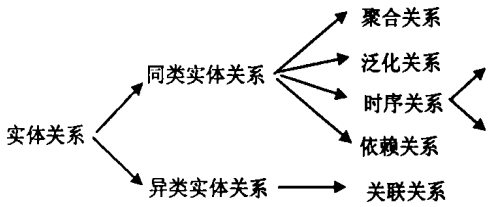

图1 过程实体关系分类图

异类实体关系描述不同过程实体之间的关系,可将其抽象为具有一定含义的关联关系,如角色与活动之间为执行关系,活动与产品之间为使用或生成关系,任何一个过程实体均可通过关联关系找到与其关联的其它过程实体。

同类实体关系描述相同过程实体之间的关系,同类实体关系相对复杂,是软件过程模型复杂性的主要体现。同类实体关系可抽象为四种类型:聚合关系、泛化关系、时序关系和依赖关系(各关系均采用 UML 中的定义与语义)。角色、活动、产品均可以分解、细化,形成聚合、泛化关系;活动之间存在时序关系,时序关系可进一步抽象为顺序、并行、循环等关系;产品之间存在依赖关系,如概要设计是详细设计的依据,详细设计报告与概要设计报告之间存在依赖关系。

对软件过程实体及异类实体关系的描述体现了软件过程系统的静态结构,对同类实体关系的描述体现了软件过程系统的动态行为,与此对应,软件过程模型由静态视图和动态视图组成。过程模型静态视图定义过程实体的属性和操作以及异类实体关系,过程模型动态视图定义同类实体关系,本文定义的软件过程模型为以活动为中心的过程模型,因此将主要定义活动之间的时序关系。

*)国家自然科学基金资助项目(70271045)。梁义芝 副教授,博士生,主要研究领域为决策支持系统、软件工程。王延章 教授,博士生导师,主要研究领域为管理决策支持系统、经济系统分析、数学规划。刘云飞 高工,主要研究领域为网络技术,多媒体技术,软件工程。缪旭东 副教授,主要研究领域为决策支持系统、软件工程。

2.1 软件过程模型的静态视图

定义1 软件过程模型的静态视图为满足以下条件的五元组 Msv=(A,P,Rol;Rpa,Rra):

(1) A 是非空有限活动集;

(2) P 是非空有限产品集;

(3) Rol 是非空有限角色集;

(4) Rpa⊆P×A∪A×P,为活动与产品之间的关联关系集,其中"×"为笛卡儿积;

(5) Rra⊆Rol×A,为角色与活动之间的关联关系集;

(6) $(\forall r)(r\in Rol\rightarrow(\exists a)(a\in A\wedge(r,a)\in Rra))$

且$(\forall a)(a\in A\rightarrow(\exists r)(r\in Rol\wedge(a,r)\in Rra^c))$

即:任何角色都必须执行一定的活动,反之任何活动都必须由对应的角色执行。其中 Rra^c 为 Rra 的逆关系。

(7) $(\forall a)(a\in A\rightarrow(\exists p1,p2)(p1\in P\wedge p2\in P\wedge(p1,a)\in Rpa\wedge(a,p2)\in Rpa))$

即:任何活动都必须使用并生成活动产品。

(8) $(\forall p)(p\in P\rightarrow(\exists a)(a\in A\wedge((p,a)\in Rpa\vee(a,p)\in Rpa)))$

即:任何产品都必须由活动使用或生成。

对各过程实体属性及操作的定义本文略,可参阅文[3]。

2.2 软件过程模型的动态视图

定义2 软件过程模型的活动变迁网为满足下列条件的三元组 NA=(A,T;F):

(1) A 是非空有限活动集;

(2) T 是非空有限变迁集;

(3) $A\cap T=\phi$ 且 $A\cup T\neq\phi$;

(4) F⊆A×T∪T×A,为活动与变迁之间的控制流关系集;

(5) dom(F)∪cod(F)=S∪T

其中,dom(F)={x|∃ y:(x,y)∈F},cod(F)={y|∃ x:(x,y)∈F}分别为 F 的定义域和值域。

由定义2可知,软件过程模型的活动变迁网满足 Petri 网的定义。

活动分为原子活动、复合活动和伪活动。原子活动不可再分解;复合活动可再分解;伪活动用于辅助说明活动的执行顺序,包括初始(init)、分叉(fork)、汇合(join)、分支(branch)、合并(merge)和终止(final)伪活动,一个活动变迁网 NA 中只能有一个初始活动和一个终止活动。定义活动类型映射 TYPE:A→{ATOMIC,COMPOSITE, PSEUDO},其中 PSEUDO={init,fork,join,branch,merge,final}。

由于复合活动可以逐层分解形成活动树,为说明活动树的层次结构,特给出以下约定:

(1)CHILD:$A\rightarrow 2^A$,为活动映射,刻画活动树中的父子节点关系。对 a∈A,CHILD(a)为 a 的子活动集。

(2)$CHILD^+=\bigcup_{i\geqslant 1}CHILD^i$ 为关系 CHILD 的传递闭包。对 a∈A,$CHILD^+(a)$为 a 的所有子孙的集合。

为保证活动树的严格树型结构,有以下条件需满足:

(1)$(\forall a)(a\in A\rightarrow a\notin CHILD^+(a))$;

(2)$(\forall a1)(\forall a2)(a1\in A\wedge a2\in A\wedge a1\notin CHILD^+(a2)\wedge a2\notin CHILD^+(a)\rightarrow(CHILD^+(a1)\cap CHILD^+(a2))=\phi)$;

(3)$(\forall a1)(a1\in A\wedge a1\neq root\rightarrow(\exists a2)(a2\in A\wedge a1\in CHILD(a2)))$。

设对活动 a∈A 有函数 INp、OUTp、INpstate、OUTpstate,其中 INp(a)求得活动 a 的输入产品集合;OUTp(a)求得活动 a 的输出产品集合(新创建的产品和状态被改变的产品均为活动的输出产品);对任意产品 p∈INp(a),INpstate(p)求得 a 中的活动所需的输入产品 p 的状态集合;对任意产品 p∈OUTp(a),OUTpstate(p)求得 a 中的活动产生的输出产品 p 的状态集合。若 a∈A 为伪活动,则:INp(a)=OUTp(a)=ϕ。

变迁说明一个活动完成时触发下一个活动的控制流,显示一个活动到下一个活动的路径。一个变迁由3部分组成:源活动集、监护条件、目标活动集。活动完成时将使变迁发生,该活动所组成的集合为该变迁的源活动集;变迁结束时将触发的活动的集合为该变迁的目标活动集;监护条件是变迁发生需满足的一个布尔表达式,若变迁没有监护条件,则认为该变迁的监护条件为真。

设对变迁 t∈T 有以下函数:source(t)求得 t 的源状态集;target(t)求得 t 的目标状态集,guard(t)求得 t 的监护条件,guard(t)=ture/false。

活动和变迁的图示均采用 UML 中的表示方法,如图2所示。

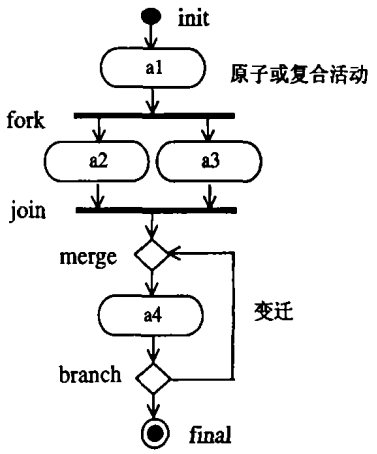

图2 活动变迁网示意图

定义3 设 X=A∪T 为 NA 的元素集,对任意 x∈X:

$^{\cdot}x=\{y|(y,x)\in F\}$称为 x 的前集,

$x^{\cdot}=\{z|(x,z)\in F\}$称为 x 的后集。

规则1 $(\forall t)(t\in T\rightarrow|^{\cdot}t|=|t^{\cdot}|=1)$。

规则2 $(\forall a)(a\in A\wedge TYPE(a)\in\{ATOMIC,COMPOSITE\}\rightarrow|^{\cdot}a|=|a^{\cdot}|=1)$。

由于活动变迁网 NA 中不存在活动 a∈A 和变迁 t∈T,使得 $a\in{}^{\cdot}t\cap t^{\cdot}$,因此 NA 为单纯网。

定义4 软件过程模型的动态视图为满足下列条件的六元组 Mdv=(A,T;F,P,Mo,Me):

(1) NA=(A,T;F)为活动变迁网,称为 Mdv 的基网;

(2) P 是非空有限产品集,为活动使用或生成的产品,对任意产品 p∈P,函数 state(p)求得产品的当前状态集;

(3) M=MA∪MP 为 Mdv 的当前格局,其中:

MA 表示当前活动集,MP 表示当前产品集;

Mo 为 Mdv 的初始格局,Mo=MAo∪MPo,其中:

MAo={init},为初始活动集;

MPo 为初始产品集;

Me 为 Mdv 的终止格局,Me=MAe∪MPe,其中:

MAe={final},为终止活动集;

MPe 为终止产品集;

(4) 只有一个变迁 t∈T 满足(init,t)∈F 且该变迁 t 没有监护条件;

(5) 只有一个变迁 t∈T 满足(t,final)∈F;

(6) 对∀ t∈T 有:(t, init)∉F ∧ (final, t)∉F。

定义5 软件过程模型的动态视图 Mdv 中,∀ t∈T 在 M 可以发生的条件为:

(1)·t=source(t) ∧ ·t⊆MA ∧ MA∩t·=ϕ;

(2)(∀ a)(∀ p)(a∈t· ∧p∈INp(a)→p∈MP ∧ INpstate(p)⊆state(p));

(3) guard(t)=ture。

对定义5的解释:变迁 t 在当前格局 M 下有发生权需满足以下条件:

(1) 变迁 t 的前集与其源状态集相同,t 的前集是当前活动集的子集,当前活动集与 t 的后集的与为空;

(2) 所有包含于 t 的后集中的活动所需的输入产品一定存在于当前产品集 MP 中,所需的产品状态集一定是该产品的当前状态集的子集;

(3) 变迁 t 的监护条件应满足。

变迁 t 在当前格局 M 下有发生权记为:M[t⟩。

定义6 软件过程模型的动态视图 Mdv 中,若 M[t⟩,则变迁 t 在 M 可以发生,将当前格局 M 变为 M 的后继 M′=MA′∪MP′,具体结果为:

(1) MA′=t·∪(MA−·t);

(2) $MP'=(\bigcup_{s\in t\cdot}OUTp(s))\cup MP$

为下文描述方便,定义以下表示符号:

在格局 M 下,变迁 t 将 M 变成 M′记作 M[t⟩M′。

在格局 M 下,从 M 出发经变迁发生可以到达的所有格局的集合称为 Mdv 从 M 出发的可达集,记为[M⟩。[Mo⟩为 Mdv 的可达集。

规则3 (∀ a)a∈A ∧TYPE(a)=fork,a·中的所有变迁没有监护条件且必须同时发生,发生的条件为 a·中的所有变迁发生条件的与,发生的结果将当前格局 M 变为 M′,则具体结果为:

(1) MA′=MAA·∪(MA−{a})

其中:$MAA=(\bigcup_{t\in a\cdot}t\cdot)$

(2) $MP'=(\bigcup_{a\in MAA}OUTp(a))\cup MP$。

规则4 (∀ a)a∈A ∧TYPE(a)=join,·a 中的所有变迁没有监护条件且必须同时发生。发生的条件为·a 中的所有变迁发生条件的与,发生的结果将当前格局 M 变为 M′,则具体结果为:

(1) $MA'=\{a\}\cup(MA-(\bigcup_{t\in\cdot a}\cdot t))$;

(2) MP′= MP 。

规则5 (∀ a)a∈A ∧TYPE(a)=branch,a·中的所有变迁必须有监护条件,任何情况下必须有且仅有一条监护条件被满足。

3 基于过程模式的软件过程建模的可行性与必要性

定义5中变迁 t∈T 的发生权是由作为全局资源分布的标识 M 来定义的,但实际上 M[t⟩只与 t 的外延·t·(·t·=·t∪t·)有关,M 中·t·以外的资源对 M[t⟩没有影响,换句话说,动态视图 Mdv 的全局状态不是变迁的控制因素,变迁 t 满足局部确定性,即变迁能否发生只依赖于它的外延,与全局状态无关。由于变迁 t 发生的结果是触发 t·中的活动的执行,因此我们说活动 a∈A 同样满足局部确定性。

过程模型中的复合活动可分解为原子活动或复合活动,因此一个复合活动可以有其内部结构、内部行为和内部状态。根据局部确定性,这些内部行为和内部状态的存在不会影响其它活动的行为,其它活动的行为也不会影响该活动的内部行为和内部状态,因而在考虑过程模型系统整体行为时可以把有内部结构的复合活动当作没有中间状态且不可中断的原子活动,在考虑复合活动的行为时可以把其作为一个相对独立的完整的系统研究其内部结构、内部行为和内部状态。本文第2部分对软件过程模型的定义同样适合于描述一个复合活动,从另一个角度我们可以把一个软件过程看作是粒度最大的复合活动。

由于可以相对独立地分别研究软件过程中的活动、为其建立过程模型,并在此基础之上组合完整的软件过程模型,使得软件过程研究和软件过程建模的复杂度大大降低,同时增加了过程建模的灵活性,增加了过程知识的重用程度,对软件过程中各活动的研究成果及活动的过程模型可重用于不同的软件过程模型。

软件过程中的活动在不同的活动约束条件下和不同的软件过程模型总体要求下其活动过程是不同的,同一个活动可以对应多个实现该活动的活动过程,如何描述和重用这些过程知识是软件过程研究和软件过程建模必须解决的问题。软件过程模式可以说是描述和重用活动过程模型的一个非常好的方法。软件过程模式描述了软件过程中经过验证的成功的方法或一系列活动,是软件过程模型中成熟的过程步(即活动)按模式概念体系的抽象[2]。过程模式描述的主要内容有:

模式名称

模式意图:描述模式的基本原理、目的、意图;

针对问题:过程模式可以解决的问题描述;

设计要素:解的部分必须满足的要求、必须考虑的约束条件、解应具备的特性等;

活动:使用该过程模式提供的策略可以完成的活动,每个模式对应一个活动(一个活动可对应多个模式);

解:给出在满足设计要素的条件下,可以解决针对问题、完成对应活动的方法指导和实用建议,具体给出一组子活动列表及子活动之间的关系(即活动过程模型);

初始上下文:描述问题和问题的解重复出现的前提,为过程模式施用之前的系统格局;

结果上下文:过程模式应用后产生的结果,为过程模式施用后的系统格局。

过程模式只用于描述可分解的复合活动。

关于过程模式的其它内容参见文[2,3]。

过程模式可以描述和重用软件过程知识,是构建软件过程模型的可重用的构造部件。一方面可以利用过程模式总结、积累和重用过程知识,为软件组织建立成熟的过程模型,实现过程模型的动态生成以适应过程实施过程中的不断变化,实现过程的持续改进以不断完善过程知识;另一方面利用过程模式建模可以建立适合不同层次的过程管理和实现人员使用的不同粒度的过程模型,支持不同层次的过程任务的实施。

4 基于过程模式的软件过程建模

基于过程模式建立软件过程模型,可将过程模式作为构建过程模型的基本构件,不必一切从零开始;可用过程模式逐

步组合出过程模型，用过程模式逐层细化软件过程中的活动，从而可以方便地调整过程模型的描述粒度，动态调整和生成适合当前软件过程需要的过程模型。

基于过程模式的软件过程建模需解决以下问题：过程模式的描述及其正确性保证；过程模式的匹配选择与完善；过程模式的使用，即如何用过程模式组合过程模型；在过程模式正确的条件下如何保证用其组合生成的过程模型的正确性。

4.1 过程模式的描述及其正确性保证

过程模式中可以并需要进行形式化描述的是解中的活动过程模型及与之对应的初始上下文和结果上下文。活动过程模型可采用软件过程模型的静态视图 Msv 和动态视图 Mdv 描述，其中动态视图 Mdv 描述了一个动态系统，应给出其描述的动态系统的正确性定义与正确性条件。

定义7 动态视图 Mdv 是一个动态系统，若其在初始格局 Mo 下，经过一系列变迁的发生，最终一定能够到达终止格局 Me，则称动态视图 Mdv 的构造是正确的。

规则6 下列条件为动态视图 Mdv 的正确性条件：

(1)可终止：$Me \in [Mo\rangle$。由于对$\forall t \in T$ 有：$(final,t) \notin F$，因此 Me 下不会有新的变迁发生；

(2)无死锁：$(\forall M)(M \in [Mo\rangle \wedge M \neq Me \rightarrow (\exists t)(t \in T \wedge M[t\rangle))$；

(3)无死循环：$(\forall M)(M \in [Mo\rangle \wedge M \in [M\rangle \rightarrow (\exists M')(M' \in [M\rangle \wedge M \notin [M'\rangle))$。

除终止格局以外的任何格局 $M \in [Mo\rangle$ 均有可发生变迁，则称 Mdv 无死锁。

若格局 M 为自身的可达状态(即 $M \in [M\rangle$)，则一定存在格局 M′为格局 M 的可达状态(即 $M' \in [M\rangle$)且 M 不为 M′的可达状态(即 $M \notin [M'\rangle$)，则称在格局 M 处无死循环。若 Mdv 中，在所有格局 $M \in [Mo\rangle$ 处均无死循环，则称 Mdv 无死循环。

过程模式中的初始上下文即是动态视图 Mdv 的初始格局 Mo，结果上下文是动态视图 Mdv 的终止格局 Me。

根据规则6可写出自动判断动态视图 Mdv 正确性的算法，本文略。

4.2 过程模式的匹配选择与完善

过程模式是活动在一定约束条件下执行时应遵循的基本模式，活动描述做什么，过程模式描述应如何做。一个活动可以对应多个过程模式，为活动选择过程模式时，应从该活动对应的过程模式集合中查找针对问题和设计要素满足该活动要求的过程模式，完全满足要求的过程模式可直接使用；略有不足者可经修改后使用，经过使用验证可形成该活动的新的过程模式。应注意在过程模式的使用中不断完善过程模式。

4.3 使用过程模式建立软件过程模型

使用过程模式建立过程模型需解决的主要问题是如何利用过程模式组合出各种粒度的软件过程模型和如何利用过程模式细化过程模型中的复合活动。

4.3.1 过程模式的组合 复合活动 a 由多个子活动复合而成，其所对应的过程模式描述了这些子活动的时序关系，若需减小 a 的描述粒度，细化描述程度，用 a 的子活动的子活动来描述 a，只需组合 a 的子活动的过程模式，形成 a 的进一步细化的活动过程模型或过程模式。同理，可对复合活动 a 的描述进行任意程度的细化，直至用原子活动描述。因此我们说利用过程模式可组合出任意粒度的软件过程模型。

本文以两个过程模式的组合说明组合方法，可类推到多个过程模式的组合。

若 a 中的两个子活动 a1和 a2(均为复合活动)由变迁 t 相连，由于变迁 t 连接的是两个复合活动，变迁 t 没有监护条件。示意图如图3所示。

子活动 a1和 a2分别对应过程模式 Pat_a1和 Pat_a2，Pat_a1和 Pat_a2中活动过程模型的动态视图分别为 Mdv_a1和 Mdv_a2，如图4、图5所示。图中 a11、a12、a21、a22可为任意原子活动、复合活动或伪活动。

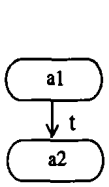

图3

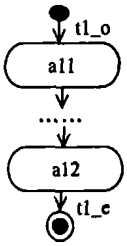

图4 Mdv_a1

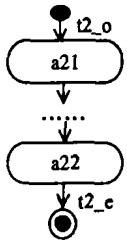

图5 Mdv_a2

Pat_a1的结果上下文与 Mdv_a1的终止格局相同，记为 Me_a1，Pat_a2的初始上下文与 Mdv_a2的初始格局相同，记为 Mo_a2，则有 MAe_a1={final}，MAo_a2={init}。

由于 a2是 a1的后继，因此有 MPe_a1=MPo_a2。又因为如前所述：若 $a \in A$ 为伪活动，则：INp(a)=OUTp(a)=ϕ，即伪活动不影响产品及产品的状态，所以活动 a12执行结束时的产品集即为 MPe_a1，执行活动 a21需要的产品集即为 MPo_a2。由此得出结论：活动 a21可以作为活动 a12的后继活动。

综上所述，组合 Pat_a1和 Pat_a2即组合 Mdv_a1和 Mdv_a2，需将 Mdv_a1中的伪活动 final 和 Mdv_a2中的伪活动 init 去掉，将 a12和 a21用一个组合变迁 t12相连，形成新的活动变迁网和新的动态视图。如图6所示。

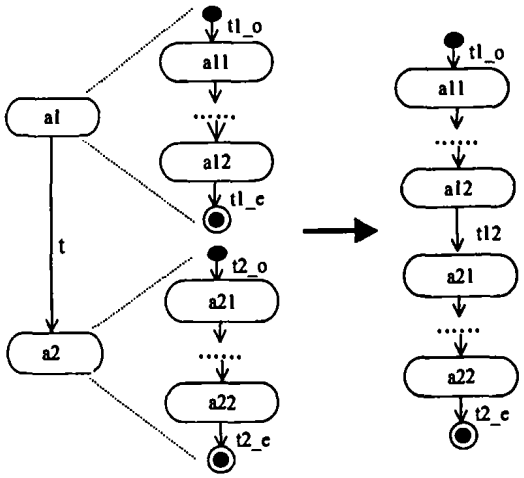

图6 过程模式组合示意图

下面给出可以组合两个过程模式的条件(规则)并定义组合变迁 t12、定义组合后的新的活动变迁网和新的动态视图。

规则7 组合两个过程模式 Pat_a1和 Pat_a2需满足以下过程模式组合条件：

(1) 过程模式 Pat_a1和 Pat_a2所描述的两个活动 a1和 a2为复合活动 a 的两个子活动，对复合活动 a 存在动态视图 Mdv_a=(A_a,T_a;F_a,P_a,Mo_a,Me_a)，那么一定存在变迁 $t \in T\_a$，且 $(a1,t) \in F\_a \wedge (t,a2) \in F\_a$；

(2) 过程模式 Pat_a1所描述的动态视图的终止活动集

MAe_a1={final};

(3) 过程模式 Pat_a2所描述的动态视图的初始活动集 MAo_a2={init};

(4) 过程模式 Pat_a1所描述的动态视图的终止产品集 MPe_a1与过程模式 Pat_a2所描述的动态视图的初始产品集 MPo_a2相同,即 MPe_a1=MPo_a2。

定义7 对满足规则7的两个过程模式,组合其中一个过程模式所描述的活动变迁网 NA1=(A1,T1;F1)中的变迁 t1_e 和另一个过程模式所描述的活动变迁网 NA2=(A2,T2;F2)中的变迁 t2_o,形成新的组合变迁 t12和新的活动变迁网 NA= NA1∪NA2=(A,T;F),其中:

(1)(t1_e ,final)∈F1 ∧ (init ,t2_o)∈F2;

(2) source(t12)=source(t1_e);

(3) target(t12)=target(t2_o);

(4) guard(t12)= guard(t1_e) ∧ guard(t2_o);

(5) A= A1∪A2

(6) T= T1∪T2∪{t12}{t1_e ,t2_o}

(7) 令(a12,t1_e)∈F1,(t2_o ,a21)∈F2,

则 F=F1∪F2∪{(a12,t12), (t12,a21)}−{(a12,t1_e), (t1_e ,final) ,(init ,t2_o),(t2_o ,a21)}。

定义7中第(2)(3)(4)条定义了组合变迁 t12,第(5)(6)(7)条定义了新的活动变迁网 NA,由于活动树为严格树型结构,不存在活动 a∈A 使得 a∈·t12∩t12·,因此该网仍为单纯网。

定义8 组合两个满足过程模式组合条件(规则7)的过程模式所描述的动态视图 Mdv1=(NA1,P1,Mo1,Me1)和 Mdv2=(NA2,P2,Mo2,Me2),形成新的动态视图 Mdv=(NA,P,Mo,Me),其中:

(1) NA= NA1∪NA2;

(2) P=P1∪P2;

(3) Mo= Mo1;

(4) Me= Me2。

组合过程模式形成的新的动态视图 Mdv 中变迁发生的条件和结果仍采用定义5和定义6,同时,新的动态视图 Mdv 满足规则3、规则4和规则5。

4.3.2 利用过程模式细化过程模型中的复合活动 在过程模型的使用中经常需要对某一活动进行细化描述,以满足具体工作的需求。如软件测试人员,他们只需了解软件过程中除测试活动以外的其它活动的大致情况,对测试活动则需详细了解,因此他们需要的过程模型对除测试活动以外的其它活动的描述粒度可以较大,对测试活动的描述粒度则应根据需要逐步细化。

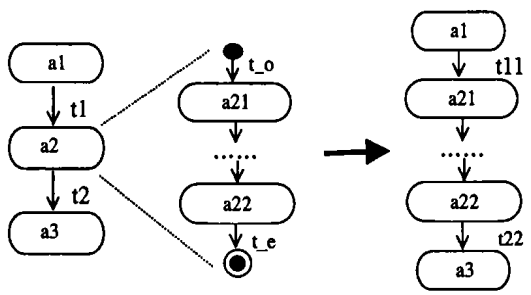

图7 过程模式细化示意图

过程模型中的任何复合活动均可细化。如图7所示,a2为任一复合活动,通过变迁 t1和 t2与 a2相连的活动 a1和 a3可为任意原子活动、复合活动、伪活动。细化复合活动 a2时,需将 a2用其过程模式所描述的子活动替代,将变迁 t1 和 t_o 、t2 和 t_e 分别组合为组合变迁 t11和 t22,形成新的活动变迁网和新的动态视图。

下面定义组合变迁 t12和 t22,定义复合活动细化后的新的活动变迁网和新的动态视图。

定义9 对活动变迁网 NA1=(A1,T1;F1)中的复合活动 a2,用其过程模式中的活动变迁网 NA2=(A2,T2;F2)对其进行细化,需分别对应组合 NA1中的变迁 t1、t2和 NA2中的变迁 t_o、t_e,形成新的组合变迁 t11、t22和新的活动变迁网 NA= NA1 ∪NA2 =(A,T;F),其中:

(1) (t1 ,a2)∈F1 ∧ (a2 ,t2)∈F1;

(2) (init ,t_o)∈F2 ∧ (t_e ,final)∈F2;

(3) source(t11)=source(t1);

(4) target(t11)=target(t_o);

(5) guard(t11)= guard(t1) ∧ guard(t_o);

(6) source(t22)=source(t_e);

(7) target(t22)=target(t2);

(8) guard(t22)= guard(t_e) ∧ guard(t2);

(9) A= A1∪A2−{a2}

(10) T= T1∪T2∪{ t11,t22}−{ t1,t2,t_o,t_e }

(11) 令(t_o ,a21)∈F2,(a22,t_e)∈F2

则 F= F1∪F2∪{ (a1,t11),(t11,a21),(a22,t22),(t22,a3)}−{ (a1,t1),(t1,a2),(a2,t2),(t2,a3),(init,t_o),(t_o ,a21) ,(a22,t_e) (t_e ,final)}

定义9中第(3)(4)(5)条定义了组合变迁 t11,第(6)(7)(8)条定义了组合变迁 t22,第(9)(10)(11)条定义了新的活动变迁网 NA,由于活动树为严格树型结构,不存在活动 a∈A 使得 a∈·t11∩t11·或 a∈·t22∩t22·,因此该网仍为单纯网。

定义10 对动态视图 Mdv1=(NA1,P1,Mo1,Me1)中的复合活动 a2,用其过程模式中的动态视图 Mdv2=(NA2,P2,Mo2,Me2)对其进行细化,形成新的动态视图 Mdv=(NA,P,Mo,Me),其中:

(1) NA= NA1∪NA2;

(2) P = P1∪P2;

(3) Mo= Mo1;

(4) Me= Me1。

细化复合活动形成的新的动态视图 Mdv 中变迁发生的条件和结果仍采用定义5和定义6,同时,新的动态视图 Mdv 满足规则3、规则4和规则5。

4.4 基于过程模式建立的过程模型的正确性分析

定理1 描述一个复合活动中的两个子活动的两个过程模式,若其满足过程模式组合条件(规则7),并且该复合活动过程模型的动态视图和两个过程模式的动态视图均满足正确性条件(规则6),则组合后生成的新的过程模式或过程模型的动态视图同样满足正确性条件。

证明:(下文中直接使用4.3.1中的符号及其意义并参见图6)。

令 M 标识在组合后的新的动态视图 Mdv=(NA,P,Mo,Me)中活动 a12完成时的当前格局。

首先证明在组合后的新的动态视图 Mdv 中,M 下 t12可以发生:

由定义7、两个过程模式的动态视图均满足规则6及活动树的严格树型结构,有:

$'t12=\{a12\} \wedge soutce(t12)=soutce(t1\_e)='t1\_e=\{a12\} \rightarrow 't12=source(t12)$;

$'t12='t1\_e \wedge 't1\_e \subseteq MA \rightarrow 't12 \subseteq MA$;

$t12'=t2\_o'=\{a21\} \wedge a21 \in CHILD(a2) \rightarrow MA \cap t12'=\phi$;

定义5中的第一条被满足。

又由定义7及被组合的过程模式满足规则6有:

$t2\_o'=\{a21\} \wedge M[t2\_o\rangle \rightarrow (\forall p)(p \in INp(a21) \rightarrow p \in MP \wedge INpstate(p) \subseteq state(p))$;

因为 $t12'=t2\_o'=\{a21\}$,定义5中的第二条被满足。

又由定义7及被组合的过程模式满足规则6有:

$(guard(t12)=guard(t1\_e) \wedge guard(t2\_o)) \wedge M[t2\_o\rangle \wedge M[t1\_e\rangle \rightarrow guard(t12)=ture$;

定义5中的第三条被满足。

所以根据定义5,在组合后的新的动态视图 Mdv=(NA,P,Mo,Me)中有:M[t12⟩。

由于组合后的新的动态视图中有 M[t12⟩,且组合变迁 t12在 M 处没有形成新的循环,又由于复合活动过程模型的动态视图及两个过程模式的动态视图均满足正确性条件,可推出组合后生成的新的过程模式或过程模型的动态视图同样满足正确性条件,即可终止、无死锁、无死循环。 证毕。

定理2 一个过程模式描述了一个复合活动中的一个子活动,用该过程模式细化其描述的子活动,形成该复合活动的细化后的新的活动过程模型;若该复合活动的原活动过程模型的动态视图和过程模式的动态视图均满足正确性条件(规则6),则该复合活动的新的活动过程模型的动态视图仍满足正确性条件。

证明:(下文中直接使用4.3.2中的符号及其意义并参见图7)。

令 M 标识在细化后的新的动态视图 Mdv=(NA,P,Mo,Me)中活动 a1完成时的当前格局。

首先证明在组合后的新的动态视图 Mdv 中,M 下 t11可以发生:

由定义9、复合活动的原动态视图和过程模式的动态视图均满足规则6及活动树的严格树型结构,有:

$'t11=\{a1\} \wedge soutce(t11)=soutce(t1)='t1=\{a1\} \rightarrow 't11=source(t11)$;

$'t11='t1 \wedge 't1 \subseteq MA \rightarrow 't11 \subseteq MA$;

$t11'=t\_o'=\{a21\} \wedge a21 \in CHILD(a2) \rightarrow MA \cap t11'=\phi$;

定义5中的第一条被满足。

又由定义9、复合活动的原动态视图和过程模式的动态视图均满足规则6,有:

$t\_o'=\{a21\} \wedge M[t\_o\rangle \rightarrow (\forall p)(p \in INp(a21) \rightarrow p \in MP \wedge INpstate(p) \subseteq state(p))$;

因为 $t11'=t\_o'=\{a21\}$,定义5中的第二条被满足。

又由定义9、复合活动的原动态视图和过程模式的动态视图均满足规则6,有:

$(guard(t11)=guard(t1) \wedge guard(t\_o)) \wedge M[t\_o\rangle \wedge M[t1\rangle \rightarrow guard(t11)=ture$;

定义5中的第三条被满足。

所以根据定义5,在细化后的新的动态视图 Mdv=(NA,P,Mo,Me)中有:M[t11⟩。

同理,令 M′标识在细化后的新的动态视图 Mdv=(NA,P,Mo,Me)中活动 a22完成时的当前格局。可证明:细化后的新的动态视图 Mdv 中,M′下 t22可以发生,即:M′[t22⟩。

由于在细化后的新的动态视图中,有 M[t11⟩和 M′[t22⟩,且组合变迁 t11在 M 处没有形成新的循环,组合变迁 t22在 M′处也没有形成新的循环,又由于复合活动的原动态视图和过程模式的动态视图均满足正确性条件,可推出用过程模式细化复合活动中的子活动后生成的复合活动的新的动态视图同样满足正确性条件,即可终止、无死锁、无死循环。

证毕。

总结 本文在基于过程模式的软件过程框架研究[3]的基础上,提出了该框架下的一种软件过程模型描述的形式化方法,采用该方法描述软件过程模式并说明了基于过程模式建立软件过程模型的必要性与可行性,提出了基于过程模式的软件过程建模方法并证明了方法的正确性。

参考文献

1 McChesney I R. Toward a classification scheme for software process modelling approaches. Information and software Technology,1995,37(7):363~374

2 梁义芝,王延章,赵晓哲,缪旭东. 软件领域中的模式研究. 计算机科学,2003,30(3)

3 梁义芝,王延章,刘云飞,缪旭东. 基于过程模式的软件过程框架. 计算机科学,2003,30(8)

4 Bolton C,Davies J. Activity Graphs and Process. In:Grieskamp W,Santen T,Stoddart B,eds. IFM 2000,LNCS 1945. Germany:Springer-Verlab Berlin Heidelberg,2000. 77~96

5 Borger E,Cavarra A,Riccobene E. An ASM Semantics for UML Activity Diagrams. In:Rus T,ed. AMAST 2000,LNCS 1816. Germany:Springer-Verlab Berlin Heidelberg,2000. 293~380

6 Gnatz M,Marschall F,Popp G,Rausch A,Schwerin W. Towards a Living Software Development Process based on Process Patterns. In:Ambriola V,ed. EWSPT 2001,LNCS 2077. Germany:Springer-Verlab Berlin Heidelberg,2001. 182~202

7 袁崇义. Petri 网原理. 第一版. 北京:电子工业出版社,1998

8 Unified Modeling Language Specification. Version 1.3. Object Managagement Group,Inc,1999

9 Rumbaugh J,Jacobson I,Booch G. The Unified Modeling Language Reference Manual. Boston:Addison-Wesley,1999