第33卷 第14期 计 算 机 工 程 2007年7月
Vol.33 No.14 Computer Engineering July 2007
·软件技术与数据库· 文章编号：1000—3428(2007)14—0073—03 文献标识码：A 中图分类号：TP311

支持软件演化的第2代SDDM软件过程模型

赵 娜，赵锦新，李 彤
(云南大学信息软件学院，昆明 650091)

摘 要：软件演化已成为软件生存周期中最重要的形态之一。该文对支持软件并行工程的 SDDM 过程模型在标准化、角色模型、资源模型、层次化4个方面进行了扩展，得到了一个支持标准化的可配置、可重用、支持软件演化过程的第2代SDDM软件过程模型。
关键词：软件演化；软件过程模型；Petri网；SDDM模型

Software Process Model Supporting Software Evolution——SDDM Ⅱ

ZHAO Na, ZHAO Jinxin, LI Tong
(College of Information and Software, Yunnan University, Kunming 650091)

【Abstract】 Software evolution becomes an important characteristic in software life cycle. System dynamic development model, which supports software concurrent engineering, is extended with standardization, role model, resource model and hierarchy. It produces the second generation SDDM, which is configurable, reusable, supporting the standardization and software evolution model.
【Key words】 software evolution; software process model; Petri net; SDDM model

软件的提交形态已从产品形态向服务形态转换，软件需要根据用户的需求、技术的发展作出改变，推动软件从低级走向高级、从幼稚走向成熟，可用“演化”来描述这种改变[1,2]，并将演化看成是持续不断的“软件再工程”[1]。

随着越来越多的成功软件系统变成了遗产系统(legacy system)，软件演化的重要性和普及性越来越强。软件演化已成为软件生存周期中最重要的形态之一。软件过程在提高软件演化的效率和质量方面发挥着重要的作用。“软件过程”和“软件演化的多学科交叉”已成为软件工程中的一个重要领域。一个良好的软件过程能够有效地促进软件演化的成功实施，反之，将导致软件演化的失败。

软件演化具有以下5个重要特征：(1)迭代性；(2)并发性；(3)持续和间断的改变是交错进行的；(4)反馈驱动系统；(5)模型体系结构是多层的。

为了更好地支持软件演化特性，本文对基于Petri网、具有面向对象特征的软件过程模型(system dynamic development model, SDDM)[3]进行了改进，在标准化、角色模型、资源模型、层次化4个方面进行扩展，得到一个支持标准化的、可配置、可重用、与层次化角色模型一体化、支持软件演化的第2代SDDM软件过程模型。

1 第2代SDDM软件过程模型的基本定义

文献[6,7]指出：“软件过程指软件生存周期中所涉及的一系列相关过程。过程是活动的集合，活动是任务的集合，任务是把输入转换为输出的操作”。在第2代SDDM建模方法中，本文把活动分为3类：(1)自动活动，活动体是任务序列，由系统自动完成；(2)个体活动，活动体是活动序列，主要由人工完成；(3)过程活动，活动体是过程。

定义1 活动或者是一个顺序执行的任务序列$T_1,T_2,...,T_n$；或者是一个软件过程[3]。活动是与角色对应的一个执行单位，它是一个八元组，即

$$A=<N,T,S,E,R,AT,SC_1,SC_2>$$

(1)N是变量名集合，T是变量类型集合。

(2)局部变量集，即

$$S=\{<n,t>|n\in N,t\in T\}$$

其中，$<n,t>$刻画了活动的一个局部变量。

(3)E是活动体，是一个任务序列，也可是一个过程。

(4)R为执行该活动的角色集。

(5)活动类型为

$$AT:A\rightarrow\{auto,individual,process\}$$

其中，*auto*表示自动活动，*individual*表示个体活动，*process*表示过程活动。

(6)$SC1$为附加于活动上的点火约束函数集，即

$$\{C_1,C_2,...,C_n\}$$

点火约束是指活动执行要满足的条件和所受到的限制。令L为谓词或者断言的一个有限集，则约束函数C_i可以形式化地表示为

$$l_1\wedge l_2\wedge...\wedge l_p\rightarrow\{0,1\},l_i\in L$$

只有当所有点火约束函数都满足时活动才可能点火。

(7)$SC2$为附加于活动上的提交约束函数集，只有当所有提交约束函数都满足时，活动才能正常提交，从而完成点火。

定义2 软件过程是一个七元组，即

$$P=<C,A,F,R,N,T,V>$$

(1)C为条件的集合，C中的条件是一个二元组，即

$$<\{Q\},AN>$$

其中，$\{Q\}$是一个谓词，表示条件；AN是一个广义的数据结构，是活动操作的对象；C中有两个特殊的条件I_0和O_0；$I_0=\phi$，

基金项目：国家自然科学基金资助项目“软件演化过程研究”(60463002)；云南省教育厅科研基金资助重点项目“软件演化过程研究”(04Z229D)
作者简介：赵 娜(1982－)，女，硕士研究生，主研方向：软件工程；赵锦新，硕士研究生；李 彤，教授、博士生导师
收稿日期：2006-09-20 **E-mail：**zhaonayx@126.com

$O_0^{'}=\phi$分别称为起始条件和终止条件，且$I_0^{'}$和$O_0^{'}$都只有一个元素。

(2)A 是活动的集合，即

$$A=A_a\cup A_i\cup A_p$$

其中，A_a为自动活动集；A_i为个体活动集；A_p为过程活动集。A中的活动a在其前提条件具备的情况下可以执行(称为点火)，但a能否顺利进行，还要看其内部的约束函数是否满足，且$C\cap A=\phi$。

(3)流关系为 $F\subseteq(C\times A)\cup(A\times C)$。

(4)R 为该软件过程所使用的角色集，其中的角色是活动集 A 中活动的候选角色。

(5)N 是变量名集合，T 是变量类型集合。

(6)软件过程变量集为

$$V=\{<n,t>|\ n\in N,t\in T\}$$

其中，$<n,t>$刻画了一个软件过程变量，软件过程变量用于信息共享和路由选择。

P 的点火规则遵从 Petri 网的点火规则。

定义 3 动态系统开发模型是一个八元组，即

$$SDDM=<I,O,A,F,R,N,T,V>$$

其中，I 为起始条件；O 为终止条件；A 是活动的集合；F 为流关系；R 为模型角色集；N 是变量名集合；T 是变量类型集合。模型变量集为

$$V=\{<n,t>|\ n\in N,t\in T\}$$

其中，$<n,t>$刻画了一个模型变量。

2 软件演化过程的标准化

把活动作为任务序列、活动、过程三者的统一体，活动可大可小，一个任务序列可看作一个活动，一个过程也可以看作一个活动。同一个活动可能因为规模、角色集等方面的不同而拥有不同的活动体，这些活动体都是同一个活动的不同实施方式，任务序列和过程都以活动体的形式存在。可以把它们归在一起，提出标准化活动、标准化过程活动体、标准化任务序列活动体的概念。

定义 4 标准化活动可以包含实现该活动的多种形式的任务序列；或者多个软件过程。它是一个七元组，即

$$SA=<EL,N,T,S,R,AT,SC_1,SC_2>$$

(1)EL 是活动体集合，对于标准化个体活动和自动活动而言，它是一个任务序列活动体集合；对于标准化过程活动而言，它是一个过程活动体集合。

(2)N 是变量名集合，T 是变量类型集合。

(3) $S=\{<n,t>|\ n\in N,t\in T\}$ 是标准化活动的局部变量集，$<n,t>$刻画了标准化活动的一个局部变量。

(4)R 为标准化活动所使用的角色集。

(5)活动类型可表示为

$$AT:A\rightarrow\{auto,individual,process\}$$

其中，$auto$ 表示标准化自动活动；$individual$ 表示标准化个体活动；$process$ 表示标准化过程活动。

(6)SC1 为附加于活动上的点火约束函数集$\{C_1,C_2,...,C_n\}$。点火约束是指活动执行要满足的条件和所受到的限制。令L为谓词或者断言的一个有限集，则约束函数C_i可以形式化地表示为

$$l_1\wedge l_2\wedge...\wedge l_p\rightarrow\{0,1\},l_i\in L$$

只有当所有点火约束函数都满足时，活动才可能点火。

(7)SC_2为附加于活动上的提交约束函数集，只有当所有提交约束函数都满足时，活动才能正常提交，完成点火。

定义 5 标准化软件过程是一个七元组，即

$$SP=<SC,SA,SF,SR,SN,ST,SV>$$

(1)SC为条件的集合，SC中的条件是二元组$<\{Q\},AN>$的集合，$\{Q\}$是一个谓词，表示条件；AN是一个广义的数据结构，是活动操作的对象，SC中有两个特殊的条件I_0和O_0，$I_0^{'}=\phi$，$O_0^{'}=\phi$分别称为起始条件和终止条件，且$I_0^{'}$和$O_0^{'}$都只有一个元素；

(2)SA 是活动的集合，即

$$SA=SA_a\cup SA_i\cup SA_p$$

其中，SA_a为标准化自动活动集；SA_i为标准化个体活动集；SA_p为标准化过程活动集。SA中的活动a在其前提条件具备的情况下可以执行(称为点火)，但a能否顺利进行，还要看其内部的约束函数是否满足，且$SC\cap SA=\phi$。

(3)流关系为

$$SF\subseteq(SC\times SA)\cup(SA\times SC)$$

(4)SR 为标准化软件过程所使用的角色集，其中的角色是活动集 A 中活动的候选角色。

(5)SN 是变量名集合，ST 是变量类型集合。

(6)软件过程变量集为

$$SV=\{<n,t>|\ n\in SN,t\in ST\}$$

其中，$<n,t>$刻画了一个软件过程变量，软件过程变量用于信息共享和路由选择。

每个活动都应该对应一个、且只能对应一个标准化活动。如果活动体是任务序列，则在新活动中生成一个任务序列，每一个任务都对应一个标准化任务；如果活动体是过程，则在新活动中生成一个过程，其中每个活动都分别与标准化过程中的标准化活动一一对应。活动标准化示例见图 1。

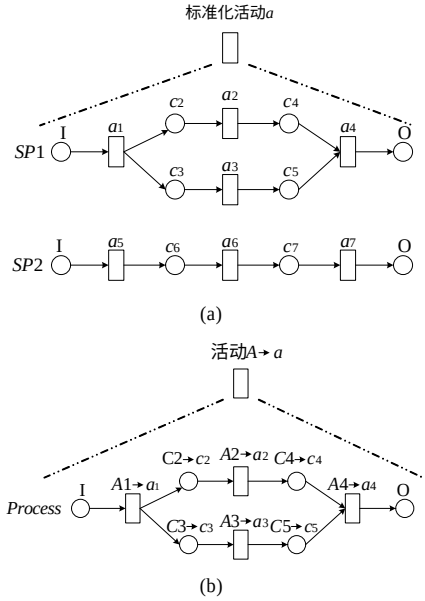

图 1 活动标准化示例

本文根据标准化活动 a 创建活动 A，标准化活动 a 有 2 个活动体可选(见图 1(a))，假设本文选择根据标准化活动体 $SP1$ 为活动 A 生成活动体，则得到如图 1(b)所示的过程活动体 $Process$，其中的条件和活动都是根据 $SP1$ 中的标准化条件和活动产生，接下来继续为 $Process$ 中的每个活动配置合适的活动体。

也可采用与图 1 中反向的方式，根据活动的活动体为其标准化活动，生成对应的标准化活动体，提交后的标准化活动体在标准化工程师审核后，可被其它活动选择配置和重用。

标准化活动及活动体都存放在组织的标准化过程库中，供过程工程师建模使用。

通过活动及活动体的标准化，不仅可以实现业务的规范化和标准化管理，而且为业务过程的重用提供了一种新的机制，它重用的粒度比过程模型重用的粒度更小，重用率更高。通过它使得建模过程变得更加简便快捷，实现了配置式的软件过程建模方法。

3 软件演化过程的角色模型

本文的建模方法是以角色为中心的，每个过程都对应一组角色集，其中每个角色对应一个活动。对于过程活动而言，又会有下一层的角色集，这样形成一个层次化的角色模型。通常同一个项目成员可能会在项目中扮演多个角色，角色模型中的角色一般会比实际的项目成员多。

定义 5 角色模型是一个序偶集，其中的每个元素都是二元组，即

$$R =< Role, Role\_parent >$$

其中，$Role\_parent$ 是 $Role$ 的上级角色；$Role$、$Role\_parent$ 是项目中的角色。

角色模型与项目的组织结构不同，角色模型中角色关系反映的只是一种临时的工作关系，是一种动态的关系，随着活动的完成而结束。项目人员按功能职责进行分类，形成矩阵型的组织结构，利用角色模型动态形成具体工作单元，这样有利于资源的充分利用。

4 软件演化过程的资源模型

在 SDDM Ⅱ 中，资源分配主要是根据角色模型来进行分配。资源分配完成后，资源就会成为软件过程模型中新的约束条件，可以把它们加入到软件过程模型中。本文也可以用单独的资源模型来显示，以便更好地反映资源的分配与争用情况。

定义 6 资源模型是一个五元组，即

$$< P, A, \mathrm{Pr}, T, E >$$

其中，P 为资源集；A 为活动集；Pr 为优先级集合；T 为延迟时间的集合。执行者集合为

$$E = \{< p, a, pr, t >| \ p \in P, a \in A, pr \in \mathrm{Pr}, t \in T\}$$

每个执行者记录 1 个资源与活动间的关联，以及该关联的优先级和延迟时间。

资源模型是在资源分配和调整的过程中动态构建的：(1)对需要分配资源的所有活动按一定的顺序依次寻找其最合适的资源；(2)为每个分配资源的活动生成一个执行者，记录每个活动与资源的关联；(3)根据资源模型中资源的共享和争用情况，在执行者中设置合适优先级和延迟时间。

5 软件演化过程模型的层次化扩展

软件演化过程模型的体系结构必须是多层的。本文的软件演化过程模型是以角色为中心建立的，而角色模型是有层次的，为了使软件过程模型与角色模型一体化，必须使过程模型的建立具有层次结构，且与角色模型的结构相一致。在模型优化时，可能需要进行层次的压缩和并行性延拓。本文将过程进行规范化，以便在层次压缩时父网和子网相一致，为此规定：

定义7 每个过程都必须只有一个起始点(Start)和一个终止点(End)，且起始点只有一个输出弧，终止点只有一个输入弧。符合该规定的过程，我们称之为规范化过程。

定义 8 I_A, O_A分别为软件演化过程模型父层中过程活动 A的输入和输出集，Start和End 分别为子层中活动体A的起始点和终止点，在进行层次压缩后，父层中过程活动A的输入集I_A和输出集O_A分别成为子层中A的活动体的输入集和输出集，且原先父层当中由I_A指向过程活动A的输出弧现在指向原子层中A的活动体的起始点start所指向的过程活动;同理，子层A的活动体中指向终止点end的输出弧现指向原父层当中的过程活动A的输出集O_A。

在层次压缩时，角色模型的层次关系不会发生变化，仍然保持着角色之间的工作关系，并且角色与活动之间是一一对应的关系，实现工作任务的分配。

规范化过程层次压缩前后 Petri 网的一致性证明：

有如下假设：

(1)活动 A 为层次 Petri 网中的一个过程活动；

(2)活动A的过程活动体是一个规范化过程P，其起始点为Start，终止点为End，且活动a_1、a_2满足

$$Start^{\cdot} = \{a_1\}, {}^{\cdot}End = \{a_2\}, {}^{\cdot}a_1 = \{Start\}, a_2^{\cdot} = \{End\}$$

(3)C_1是活动A的前提条件集，C_2是活动A的后提条件集；

(4)库所的容量为 1。

压缩前的点火规则如下：

(1)当活动A的前提条件集C_1中的每个条件都含有一个Token，且点火约束函数集$SC1$ 中其它条件都满足时，活动A点火，条件集C_1中的每个条件都移出一个$Token$，活动A处于执行状态，过程活动体P启动，在其起始点Start中添加一个$Token$。

(2)Start中有$Token$，且a_1的其它点火约束函数集$SC1$ 都满足，则a_1点火，从Start中移出一个Token，并在其所有后继条件中都添加一个$Token$。

(3)当过程活动体P执行到使活动a_2的点火约束函数集$SC1$ 都满足时，a_2点火，从a_2的每个前提条件中移出一个Token，同时在其终止点End中产生一个$Token$。

(4)处于执行态的过程活动A扫描到其过程活动体的终止点End中有$Token$，结束点火，移出其终止点中的$Token$，并且过程活动A的后提条件集C_2中的每个条件添加一个$Token$。

压缩后的点火规则如下：

(1)当活动a_1的前提条件集C_1中的每个条件都含有一个Token，且点火约束函数集$SC1$ 中其它条件都满足时，活动a_1点火，条件集C_1中的每个条件都移出一个$Token$，并在其所有后继条件中都添加一个$Token$。

(2)当过程执行到使活动a_2的点火约束函数集$SC1$ 中其它条件都满足时，a_2点火，从a_2的每个前提条件中移出一个Token，“ 后提条件集 ” C_2中的每个条件添加一个$Token$。

压缩前后，过程P的输入接口和输出接口没有改变，仍然是条件集C_1和条件集C_2，当条件集C_1中的每个条件至少含有一个$Token$，且点火约束函数集$SC1$ 中其它条件都满足时，过程启动，在过程结束时为条件集C_2中的每个条件添加一个$Token$。压缩前后Petri网的功能是一致的。

在进行层次压缩时：角色模型是在层次压缩前确定的，层次压缩及重构时活动的执行角色不变，若添加活动，则应该先确定使用角色模型中的哪个角色或在哪个角色下级添加角色，然后才能添加活动，活动会根据角色的层次自动加入到相应的过程活动体中。这就是以角色为中心的建模方法其他方法的不同，角色始终是中心。

从图 2 可以看出，规范化过程与非规范化过程的差别。

(下转第 88 页)