任务状态敏感的访问控制模型及其有色网仿真

翟治年[1,2]，奚建清[2]，卢亚辉[3]，郭玉彬[4]
（1. 浙江科技学院信息学院，310023，杭州；2. 华南理工大学计算机科学与工程学院，510006，广州；3. 深圳大学计算机与软件学院，518062，广东深圳；4. 华南农业大学信息学院，510640，广州）

摘要：针对现有主动访问控制模型中授权流与工作流同步粒度不够精细的问题，提出一种任务状态敏感的访问控制模型.根据任务实例的不同状态对多个执行角色进行差异化授权，施加相应的职责分离约束，并给出一种有色网仿真分析方法.概念模型可以优化工作流中的数据利用，充分增强业务中的角色协作.仿真分析方法可跟踪工作流执行时的安全状态，并发现潜在的死锁问题.通过一个软件工作流验证了其协作与访问控制概念的可行性及其分析方法的有效性.

关键词：访问控制；工作流；任务状态；角色；有色 Petri 网
中图分类号：TP309 **文献标志码**：A **文章编号**：0253-987X(2012)12-0085-07

An Access Control Model with Task-State Sensitivity and Its CPN Simulation

ZHAI Zhinian[1], XI Jianqing[2], LU Yahui[3], GUO Yubin[4]
(1. School of Information, Zhejiang University of Science and Technology, Hangzhou 310023, China;
2. School of Computer Science and Engineering, South China University of Technology, Guangzhou 510006, China;
3. School of Computer and Software, Shenzhen University, Shenzhen, Guangdong 518062, China;
4. College of Informatics, South China Agriculture University, Guangzhou 510640, China)

Abstract: An access control model with task-state sensitivity is proposed to address the issue that the synchronizing granularity of authorization-flow and workflow is not fine enough in existing active access control models. Differentiated permissions are authorized to multiple roles under different states of a task instance, and a duty separation constraint is enforced. A colored Petri net simulation technique is provided and used as an analysis method. A concept model is used to optimize the data utilization in workflows; and the roles collaboration in business is fully enhanced. The simulation technique is utilized to track run-time safety states of a workflow, and to find out potential deadlocks. Feasibility of the collaboration and access control concept and the effectiveness of the analysis method are verified via a software workflow.

Keywords: access control; workflow; task-state; role; colored Petri net

主动访问控制(AAC, Active Access Control)是面向业务过程的权限管理范型，能够将授权流与工作流同步[1-2]，在业务执行过程中贯彻最小特权原则.现有 AAC 模型可归结为两类：直接对任务授权，再将其分配给用户/角色；根据任务与用户/角色的某种组合进行授权[3].然而，两者的授权流与工作流同步均停留在较粗粒度上：根据实例是/否运行对执行者授予/撤销相关权限.在实例生命周期中特别是运行阶段，仍可能包括多种状态[4]，权限要求未必相同.为了给访问控制提供清晰的依据，应当对不同状态下的任务权限给出精确说明.同时，还应考虑状态之间的角色差异.由此将带来如下特性.

收稿日期：2011-04-16. 作者简介：翟治年(1977—)，男，博士生；奚建清(通信作者)，男，教授，博士生导师. 基金项目：国家自然科学基金资助项目(60903114，60973100，61103038)；粤港关键领域重点突破资助项目(2006B80407001)；广东省教育部产学研结合计划资助项目(2008B090500193)；深圳市科技计划基础研究资助项目(JC201005280402A，JC200903120046A).
网络出版时间：2012-09-20 网络出版地址：http://www.cnki.net/kcms/detail/61.1069.T.20120920.1204.008.html

(1) 在任务生命周期中充分利用数据资源,优化业务实现. 由于控制流和数据流并非精确一致,工作流中经常有一些同步开销,常见情形是实例因等待某些数据暂未就绪,而已到达数据不得不被闲置. 例如,某软件过程包括分析、设计和编程等任务,程序员在需求规格和设计图到齐后才能编程. 故需求规格到达后和设计图到达前,编程任务将长时间不能就绪,但是此时应当允许程序员阅读需求,为编码做准备. 当编程任务就绪并执行后,再授予其更多权限. 也可设置一个与设计并行的任务供程序员熟悉需求,但这种业务建模方式耦合了后期的访问控制因素,给建模人员带来了额外负担. 本质上,该问题应通过增强 AAC 的业务适应性来解决.

(2) 满足任务中不同角色在不同状态下的差异化权限需求. 例如,主持人在会议任务就绪期间即可访问所有类型的文档,而普通参与者在会议开始后才能访问特定密级的文档. 该任务的授权需求因状态和角色而异,在现有 AAC 中是无法表达的.

(3) 支持轻量级的业务建模. 工作流任务有多个状态需要监督或处理,然而状态本身没有进一步的开销,并且某些情况下可以提供同等的建模能力. 例如,需求规格由分析员编写,并由项目经理审核. 通常将分析和评审表示成两个任务,但它们也可用同一任务的执行和挂起状态来表示. 分析员在执行状态下编写和修改需求,项目经理在挂起状态下审阅和确认,任务将在两个状态间来回切换,直到需求最终得以确认.

本文据此提出任务状态敏感的访问控制模型(TSSAC,Task-State Sensitive Access Control),基于任务状态对相应执行者授权,而每个任务或状态可由多个角色协作处理,从而支持前述几种新特性. 由于仿真执行是最基本的工作流分析手段,随后给出 TSSAC 的有色网(CPN,Colored Petri Net)仿真方法,以便验证授权方案在运行阶段的安全性.

1 相关工作

文献[5]提出一种支持同一任务在不同状态下差异化授权的工作流授权模型,并给出其 Petri 网表示,但它仅仅给出了形式化授权机制,未能指出其意义和优势所在,也没有说明其应用. 此外,该文献未考虑角色概念. 就任务状态敏感的访问控制来说,这一工作只是做了非常初步的尝试.

文献[1-2]给出了 WAM 的两种时态有色网(CTPN,Colored and Timed Petri Net)表示. 前者用两个变迁表示任务的开始和结束,一个库所表示任务处于运行状态,一个库所表示任务的执行用户,而对象-权限对作为着色托肯在网结构中流动. 它不能对角色建模. 后者将角色概念引入 WAM,并调整了 CTPN 表示方法:角色和对象类型均表示成库所,权限或任务表示成一个库所和两个变迁,而用户-对象二元组作为着色托肯在网中流动. 文献[6]在文献[2]的基础上提出了角色及其层次的有色网表示方法,但是,它们不能区分执行某任务这一权限与执行该任务时所需的数据访问权限,难以简洁地表示任务内的多种权限. 文献[7]结合工作流的 Petri 网表示,根据访问控制矩阵分析基于任务授权的安全性,但是没有对授权自身进行仿真. 文献[5]给出一种工作流授权模型的 Petri 网表示,支持同一任务在不同状态下的不同授权,但是不能对角色进行建模. 文献[8]针对工作流中的 SoD 约束给出了一种不考虑互斥任务执行顺序的 CPN 表示方法,并采用可达树来分析资源死锁问题,但是它没有考虑任务内部的授权,以及授权流与工作流同步的仿真问题.

2 TSSAC 模型

TSSAC 概念模型是一个 12 元组$\{I, S, R, U, T, A, C_{TA}, C_{UA}, C_{RA}, P, C_{Au}, C_{SoD}\}$,其组件定义如下:

定义 1 实例集 I. 实例是工作流定义[4]的一次执行,每个定义可以有多个实例.

定义 2 步骤集 S. 步骤是一个任务(即工作流活动[4])或某个任务的一个执行状态. 任务执行状态有多种定义,但 TSSAC 不依赖于具体定义方式.

定义 3 角色集 R. 角色是职责与能力的统一,通常源于组织中的工作岗位或过程中的人员分工.

定义 4 用户集 U. 用户是能够在某个实例的某个步骤中承担部分或全部工作的(人力)资源.

定义 5 类型集 T. 类型是一类数据对象的集合,例如源代码、工作报告,等.

定义 6 动作集 A. 动作是对某类型数据的某种操作模式,例如读、写、标注和执行.

定义 7 类型指派 $C_{TA}\subseteq I\times T(i,t)\in C_{TA}$ 表示在实例 i 中将会处理类型 t 的对象,也可表示实例 i 中所有 t 类型的对象.

定义 8 用户指派 $C_{UA}\subseteq I\times U\times R(i,u,r)\in C_{UA}$ 表示指派用户 u 在实例 i 中担任角色 r. 根据工作流定义可以确定其需要哪些角色,而每个角色都包含一组用户,从而为工作流执行提供了资源,但是,在同

一工作流的不同实例中,同一角色往往由不同的用户来担任,因此还要将用户进一步分配到实例.

定义 9　资源分配 $C_{RA}\subseteq C_{UA}\times S((i,u,r),s)\in C_{RA}$ 表示在实例 i 中分配用户 u 以角色 r 来执行步骤 s. 工作流步骤的资源分配必须具体到实例和用户,由于一个步骤可能由多个角色协作完成,还需指定用户在实例中担任的角色.

定义 10　权限集 $P\subseteq I\times U\times T\times A$ $(i,u,t,a)\in P$ 表示用户 u 在实例 i 中对类型 t 的对象执行动作 a 的许可.

定义 11　授权 $C_{Au}\subseteq P\times S((i,u,t,a),s)\in C_{Au}$ 表示在实例 i 中步骤 s 执行期间,用户 u 被许可对类型 t 的对象执行动作 a. 如果 s 表示任务而非状态,该授权需要进一步解释. 为此管理员应当预先配置任务授权在不同状态下的作用方式,例如任务挂起时禁用任何授权,而其他运行状态启用所有授权.

C_{Au} 侧重描述授权结果,直接指定它比较烦琐,因为要逐实例逐用户指定权限. 可以通过间接方式生成 C_{Au},指定每个步骤中的每个角色能够对何种类型执行什么动作,再结合 C_{RA} 就可导出 C_{Au}.

定义 12　职责分离 $C_{SoD}\subseteq(S\times R)\times(S\times R)((s,r),(s',r'))\in C_{SoD}$ 表示在任何实例中,若某用户担任 r 角色的步骤 s 已经开始或结束,则由该用户担任角色 r' 的步骤 s' 不能开始,反之亦然.

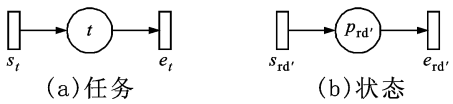
在工作流访问控制的系统结构[9]中,工作流本身分为过程定义和工作流执行服务两部分,而访问控制分为访问控制规则和访问检查两部分. TSSAC 系统仍然由上述几部分构成,不同之处在于,过程定义时应当允许基于任务、状态或其混合的业务建模;工作流执行期间根据精确的授权或各状态的默认设置来执行任务的每一个状态. 在向用户工作列表中添加工作项[4]时应当指出实例、步骤和角色,避免给同一用户生成互斥的工作项;访问控制规则由前述 12 元组描述,但可采用更简洁的间接描述;在访问检查阶段,可由用户提供工作项信息作为凭证,系统根据 C_{RA} 和 C_{Au} 为其开放一组权限,并控制其时效(仅在步骤执行期间有效),然后根据有效权限许可/拒绝用户的数据访问.

3　CPN 仿真

为建立 TSSAC 的仿真分析方法,我们基于标准 CPN 给出模型各组件的表示.

(1)开始与结束. 分别用一个输入库所 In 和一个输出库所 Out 来表示工作流的开始和结束.

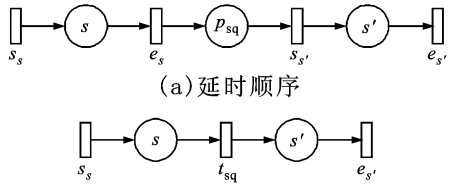
(2)步骤集. 每个步骤由一个开始变迁、一个结束变迁和一个状态库所表示. 开始/结束变迁表示该步骤的开始/结束,而状态库所表示该步骤正在运行. 如图 1a 所示,任务 t 用状态库所 t、开始变迁 s_t 和结束变迁 e_t 来表示. 类似地,在图 1b 中,任务 t' 的就绪状态用状态库所 $p_{rd'}$、开始变迁 $s_{rd'}$ 和结束变迁 $e_{rd'}$ 来表示.



(a)任务　　(b)状态

图 1　步骤的 CPN 表示

(3)步骤间的控制流. ①顺序依赖. 如图 2a 所示,为表示从步骤 s 到 s' 的顺序依赖关系,在 s 的结束变迁 e_s 和 s' 的开始变迁 $s_{s'}$ 之间添加一个状态库所 p_{sq}. 状态库所 p_{sq} 表示步骤 s 已结束而 s' 未开始的状态. 若 s' 在 s 结束后立即开始,可以采用另一种表示方式,如图 2b 所示,变迁 e_s 和 $s_{s'}$ 被合并为单个变迁 t_{sq}. ②与分支. 如图 3a 所示,为表示步骤 s 结束后两个步骤 s_1 和 s_2 并行运行,在 s 的结束变迁和 s_1、s_2 的开始变迁之间添加两个状态库所 p_{as1}、p_{as2}. 状态库所 p_{as1} 表示 s 已经结束,而 s_1 即将开始;p_{as2} 的含义类似. 若 s_1 和 s_2 将在 s 结束后立即开始,可以采用另一种表示方法,如图 3b 所示,将 3 个变迁 e_s、s_{s1} 和 s_{s2} 合并为 1 个变迁 t_{as}. ③与汇合. 如图 4a 所示,为表示步骤 s' 在 s_1 和 s_2 全部结束后开始,在 s_1、s_2 的结束变迁和 s' 的开始变迁之间添加两个状态库所 p_{aj1}、p_{aj2}. p_{aj1} 表示 s_1 结束后 s' 可能开始,p_{aj2} 的含义类似. 若 s_1 和 s_2 同时结束之后 s' 立即开始,可采用另一种表示,如图 4b 所示,将变迁 e_{s1}、e_{s2} 和 $s_{s'}$ 合并为一个变迁 t_{aj}. ④或分支. 如图 5a 所示,为表示步骤 s 结束后,若条件 $c(x)$ 为真则步骤 s_1 开始,否则步骤 s_2 开始,在 s 的结束变迁和 s_1、s_2 的开始变迁之间添加一个状态库所 p_{os},而将 $c(x)$、not $c(x)$ 分别设置为 s_{s1} 和 s_{s2} 的卫函数,其中 p_{os} 表示 s 已结束,而 s_1 或 s_2 未开始的状态. 若 s_1 或 s_2 在 s 结束后立即开始,可采用另一种表示方法,如图 5b

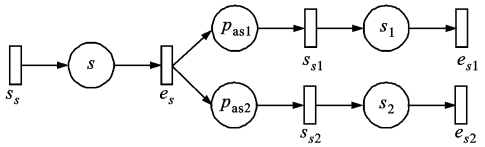


(a)延时顺序

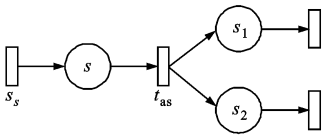
(b)即时顺序

图 2　顺序依赖的 CPN 表示

所示，将三个变迁 e_s、s_{s1} 和 s_{s2} 合并为一个变迁 t_{os}，而 s_{s1} 和 s_{s2} 的卫函数转换为 s_1 和 s_2 的输入弧函数. 由于 $c(x)$ 和 not $c(x)$ 不能同时满足，当 t_{os} 触发后，库所 s_1 和 s_2 中只有一个能够获得托肯，从而 e_{s1} 和 e_{s2} 只能有一个就绪. ⑤或汇合. 如图 6a 所示，为表示步骤 s' 将在步骤 s_1 或 s_2 结束之后开始，在 s_1、s_2 的结束变迁 e_{s1}、e_{s2} 和 s' 的开始变迁 $s_{s'}$ 之间添加一个状态库所 p_{oj}，用来表示 s_1 或 s_2 结束而 $s1$ 尚未开始的状态. 若 s_1 或 s_2 结束后 s' 立即开始，则可采用另一种表示方式，如图 6b 所示，将三个变迁 e_{s1}、e_{s2} 和 $s_{s'}$ 合并为两个变迁 t_{oj1} 和 t_{oj2}.

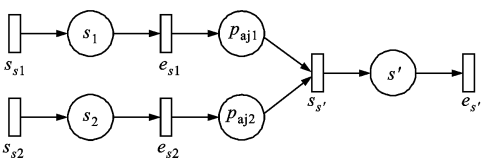


(a)延时顺序

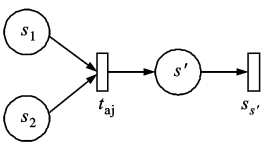


(b)即时顺序

图 3 与分支的 CPN 表示



(a)延时与汇合



(b)即时与汇合

图 4 与汇合的 CPN 表示

(4) 实例集. 用颜色集 I=INT 来表示.

(5) 类型集. 用颜色集 T=STRING 来表示.

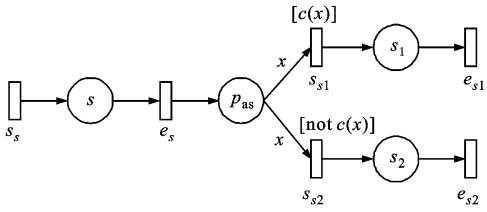
(6) 角色集. 用颜色集 R=STRING 来表示.

(7) 用户集. 用颜色集 U=INT 来表示. 每个用户用一个整数来标识.

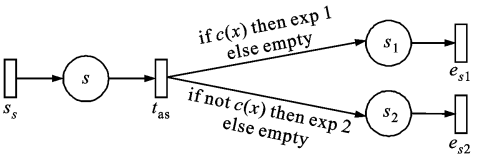
(8) 动作集. 用颜色集 A=STRING 来表示.

(9) 类型指派. 类型到实例的指派用颜色集 T_A=product $I\times T$ 来表示.

(10) 用户指派. 用户在实例中承担角色这一关系用一个库所 U_a 来表示，其标记颜色集为 U_A=product $I\times U\times R$.

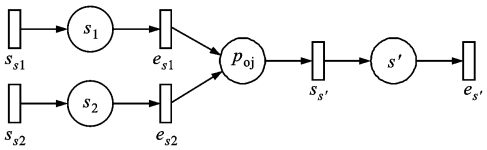


(a)延时或分支

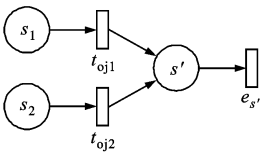


(b)即时顺序

图 5 或分支的 CPN 表示



(a)延时或分支



(b)即时顺序

图 6 或汇合的 CPN 表示

(11) 权限集. 用颜色集 $P_r=I\times U\times T\times A$ 来表示.

(12) 资源分配. 步骤 s 的资源分配用一个从 U_a 到 s_s 的输入弧来表示. 弧函数表达式是 U_A 上的多集，它决定了哪些角色负责步骤 s 以及执行 s 的实例时，每个角色各需几个用户. 在该多集中，各个元素的实例分量必须相同. 例如，图 7 中，i 是颜色集 I 的变量，u_1 和 u_2 是颜色集 U 的变量，弧函数表达式 $1'(i,u_1,\text{“}r_1\text{”})+1'(i,u_2,\text{“}r_2\text{”})$ 表示存在两个用户在同一实例 i 中分别担任角色“r_1”和“r_2”时，该实例中的步骤 s' 才能开始.

(13) 授权. ①权限授予. 对步骤 s 的权限授予用从 s 的开始变迁到其状态库所的弧来表示. 弧函数表达式是 P_r 上的多集，它决定了哪些权限将被授予 s 的实例. 例如，图 7 中，从 $s_{s'}$ 到 s' 的弧函数 $1'(i,u_1,\text{“ty”},\text{“w”})+1'(i,u_2,\text{“ty”},\text{“r”})$ 表示当实例 i 中步骤 s' 开始执行时，用户 u_1 将可以写入“ty”类

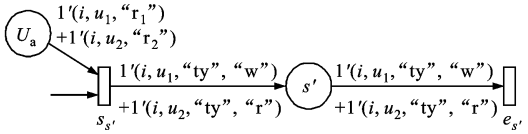


图 7　资源分配和授权的 CPN 表示

型的对象，而用户 u_2 可以读同类型对象. ②权限撤销. 步骤 s 的权限撤销用 从 s 的状态库所到其结束变迁的弧来表示. 弧函数表达式是 P_r 上的多集，它决定了哪些权限将从 s 的实例撤销. 如图 7 所示，从 s' 到 $e_{s'}$ 的弧函数 $1'(i,u_1,\text{“ty”},\text{“w”})+1'(i,u_2,\text{“ty”},\text{“r”})$ 表示当实例 i 中步骤 s' 结束时，用户 u_1/u_2 将被禁止写入/读取“ty”类型的对象.

此外，为表示数据对象/执行者的到达和离开，可以将库所 s 的标识颜色集扩展至包括 T_A 和 U_A. 这需要定义一个抽象颜色集：$E=$ union ob：T_A+ xr：U_A+pr：P_r.

(14)职责分离. 我们通过例子来说明职责分离的 CPN 表示. 如图 8 所示，在任何实例 i 中，步骤 s 和 s'均需分配一个角色为“r′”的用户 u，而步骤 s'需分配一个角色为“r′”的用户 u'. 安全策略要求(s，“r”)和(s'，“r′”)互斥，也就是说，若一个用户在某实例中担任过 s 中的“r”角色，则其不能在该实例中担任 s'中的“r′”角色，反之亦然. c 为表示这种互斥关系，添加一个约束库所 c，并从 c 到 s_s 添加一条弧，其弧函数是变量形式的用户指派 3 元组，其实例分量与 U_a 到 s_s 的弧函数保持一致，这样可以控制约束在同一实例内起作用. 对称地，添加从 c 到 $s_{s'}$的弧. 然后从 s 的开始变迁到库所 c 添加一条弧，其弧函数与从 U_a 到 s_s 的弧相同，这样当 s 开始时其资源消耗将被记录在 c 中. 对称地，为 s' 添加类似的弧. 现在，当 s（或 s'）开始时将从 c 中取出一个托肯，再放入一个托肯. 由于后一托肯记录了 s（s'）的资源消耗，我们可以将其与 s'（s）的资源要求相比较，据此设置开始变迁的触发条件，从而限制随后 s'（s）的执行. 为说明上述构造如何运行，作为初始标记，在 U_a 中放入托肯(1，1，“r”)、(1，2，“r”)和(1，2，“r′”)，表示在实例 1 中，用户 1 担任“r”角色，用户 2 既担任“r”角色，也担任“r′”角色. 同时在 c 中放入一个特殊的托肯(1，0，“0”)（其中 0 表示空值），此时对变迁 s_s，绑定关系为：$i=1$，$u_1=0$，$r_1=$“0”，$u=1$ 或 $u=2$，无论 u 取值如何，均满足 s_s 的卫函数 $u<>u_1$，于是 s_s 被使能. 类似地，$s_{s'}$也被使能. 若 s_s 以 $u=2$ 的绑定方式触发，则其将从 U_a 中移除(1，2，“r”)，表示用户 2 在实例 1 的步骤 s 中担任了“r”角

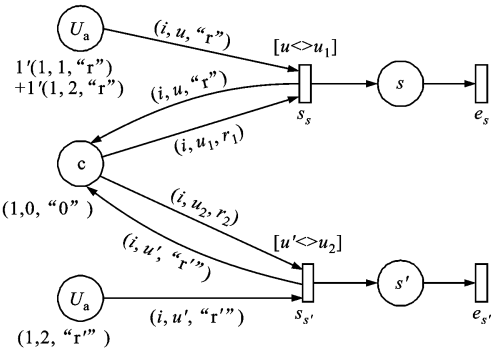


图 8　职责分离约束的 CPN 表示

色. 同时 s_s 从 c 中移除(1，0，“0”)，并代之以(1，2，“r”). 此时对变迁 $s_{s'}$，若 i 仍然绑定为 1，则 u_2 只能绑定为 2. 由于没有其他用户在实例 1 中担任“r′”角色，u'也只能绑定为 2. 这就违反了变迁 $s_{s'}$的卫函数 $u'<>u_2$，从而该变迁不能触发. 倘若步骤 s 还需角色“r″”的参与，并且用户 2 也在实例 1 中担任这一角色，那么图 8 中的 U_a 应增加一个托肯(1，2，“r”)，而从 U_a 到 s_s 的弧函数将变为 $1'(i,u,\text{“r”})+1'(i,u,\text{“r″”})$. 假设实例 1 的步骤 s 由用户 1 担任“r”角色，用户 2 担任“r″”角色，那么 s_s 触发时将从 U_a 中移出 $1'(1,1,\text{“r”})+1'(1,2,\text{“r″”})$. 由于 c 中的(1，0，“0”)将被(1，1，“r”)所代替，随后 $s_{s'}$仍然可以触发. 这说明，前述 CPN 构造能够精确模拟(s，“r”)和(s'，“r′”)之间的互斥，而不会对(s，“r”)和(s'，“r′”)形成约束.

4　案例验证

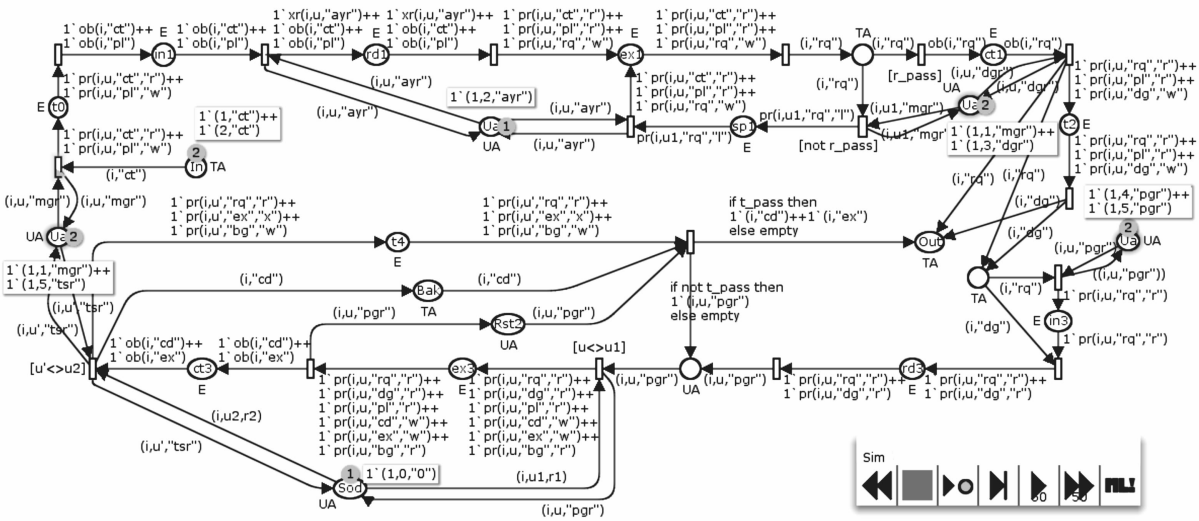
一家公司根据客户要求进行软件定制开发，建立了相应的工作流程，其中包括计划、分析、设计、编程和测试等任务. 任务可能经历初始(in)、就绪(rd)、执行(ex)、挂起(sp) 和提交(ct)等状态. 每个步骤的业务功能和负责角色如表 1 所示. 有关的对象类型包括项目合同(ct)、项目计划(pl)、需求规格(rq)、设计图(dg)、源代码(cd)、可执行代码(ex)和 bug 报告(bg). 相应的动作包括读(r)、写(w)和标注(l). 写包括标注的功能，标注包括读的功能.

上述工作流在 cpntools 3.2.2 中的仿真结果如图 9 所示. 图 9a 给出了初始标记，其中有两个项目实例 1 和 2 需要处理，其项目合同分别用开始库所 In 中的托肯 (1，“ct”)和(2，“ct”) 来表示. 比较 U_a 的初始标记和 U_a 的射出弧函数可知，实例 1 的资源要求全部可以满足：由用户 1 担任项目经理角色，用户 2 担任分析员角色，用户 3 担任设计师角色，用

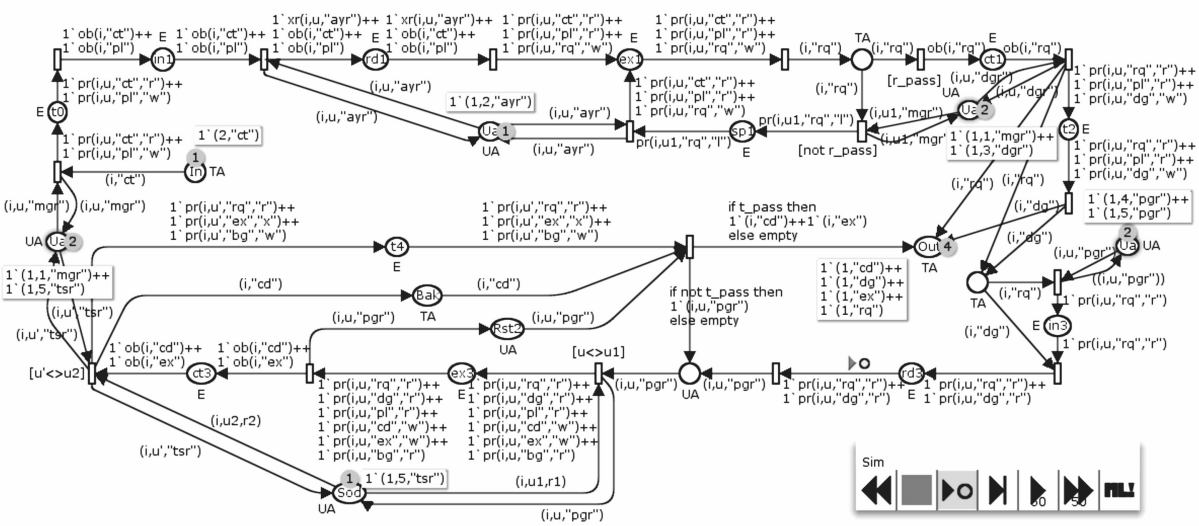
表 1 软件工作流中的业务功能和角色职责

步骤	角色	职责
计划（t0）	项目经理（mgr）	根据项目合同制订开发计划.
分析-初始(in1)	/	等待分析员就位.
分析-就绪(rd1)	分析员	在适当的时候启动分析任务的执行.
分析-执行(ex1)	分析员	根据项目合同按计划进度完成需求规格说明;若其未能通过审核,则相应进行修改.
分析-挂起(sp1)	项目经理	审核需求规格,符合要求时签字确认,否则批注修改意见.
分析-提交(ts1)	/	/
设计（t2）	设计师（dgr）	根据需求规格,按计划进度完成设计图绘制.
编程-初始（in3）	程序员（pgr）	熟悉需求.
编程-就绪（rd3）	程序员	熟悉需求和设计.
编程-执行（ex3）	程序员	根据需求规格和设计图,按计划进度编写代码;若不能通过测试,则修改相应的 bug.
编程-挂起（sp3）	程序员	等待恢复执行.
编程-提交（ct3）	/	/
测试（t4）	测试员（tsr）	根据需求规格说明对可执行代码进行测试,如果有 bug,给出测试报告,否则提交当前版本的可执行代码和源代码.

职责分离:由于测试员人手不足,一些程序员可能被分配担任测试员角色,但是必须禁止用户对自己参与开发的软件进行测试. 因此在同一实例中,若某用户在 ex3 中担任过 pgr,则应禁止其在 t4 中担任 tsr.



(a)初始标记



(b)结束标记

注:TA 为类型指派;UA 为用户指派

图 9 软件开发过程的 CPN 仿真

户 4 担任程序员角色,用户 5 担任程序员和测试员角色.将托肯(1,0,“0”) 放入约束库所 Sod 中,以便启动实例执行. t0 的开始变迁已经被使能,因为用户 1 可以在实例 1 中担任项目经理来处理项目合同(1,“ct”).当 t0 触发时,(1,“ct”) 将从 In 中移除,而 (1,1,“ct”,“r”) 和 (1,1,“pl”,“w”) 将被放入 t0,于是用户 1 可以读取项目 1 的合同并在项目计划中写入内容.与此同时,t0 的结束变迁也被使能.一旦其触发,将会从 t0 中移除(1,1,“ct”,“r”) 和 (1,1,“pl”,“w”),从而撤销用户 1 的上述权限.如此执行下去,直至如图 9b 所示的情形,结束库所 Out 中出现 4 个托肯(1,“rq”)、(1,“dg”)、(1,“cd”) 和 (1,“ex”),表示项目实例 1 的所有产出,实例 1 处理结束.值得注意的是,若 in3 的开始变迁从 Ua 中取出(1,5,“pgr”),则 ex3 执行时只能由用户 5 担任程序员.由于 Sod 的约束作用,t4 执行时不能由用户 5 担任测试员.又由于没有其他用户在实例 1 中担任测试员,工作流实例将无法执行下去(即死锁),这是职责分离约束与资源分配方案的冲突导致的.通过 CPN 仿真执行提前发现这类情况,可避免因调度不当,在工作流执行时出现难以弥补的错误.

5　结　论

本文提出一种任务状态敏感的访问控制模型 TSSAC,支持在同一任务的不同状态下对多个角色的不同授权.随后给出 TSSAC 的 CPN 仿真方法,可通过模拟运行观察工作流执行期间的安全状态.仿真分析需要反复做 CPN 执行,比较烦琐耗时.下一步将对 TSSAC 的其他分析技术进行研究.

参考文献:

[1] ATLURI V, HUANG Weikuang. An authorization model for workflows [C]// Proceedings of the 5th European Symposium on Research in Computer Security. Berlin, Germany: Springer-Verlag, 1996:44-64.

[2] ATLURI V, HUANG Weikuang. Petri net based safety analysis of workflow authorization models [J]. Journal of Computer Security, 2000, 8(2/3):209-240.

[3] 翟治年,奚建清,卢亚辉,等. 主动授权管理中的关联继承机制[J]. 西安交通大学学报, 2012, 46(4):24-31.
ZHAI Zhinian, XI Jianqing, LU Yahui, et al. Association inheritance mechanism in active authorization management [J]. Journal of Xi'an Jiaotong University, 2012, 46(4): 24-31.

[4] WFMC-TC00-1003 Workflow reference model [S]. Cohasset, MA, USA: Workflow Management Coalition, 1995.

[5] DONG Xin, CHEN Gang, YIN Jianwei, et al. Petri-net-based context-related access control in workflow environment [C] // Proceedings of the International Conference on Computer Supported Cooperative Work. Piscataway, NJ, USA: IEEE CS, 2002:381-384.

[6] ZHANG Yi, ZHANG Yong, WANG Weinong. Modeling and analyzing of workflow authorization management [J]. Journal of Network System Management, 2004, 12(4): 507-535.

[7] KNORR K. Dynamic access control through Petri net workflows [C]// Proceedings of the 16th Annual Computer Security Applications Conference. Piscataway, NJ, USA: IEEE CS, 2000:159-167.

[8] LU Yahui, ZHANG Li, SUN Jiaguang. Using colored Petri nets to model and analyze workflow with separation of duty constraints [J]. International Journal of Advanced Manufacturing Technology, 2009, 40(1): 179-192.

[9] 卢亚辉. 工作流系统的访问控制模型及其安全性分析的研究[D]. 北京: 清华大学, 2008.

(编辑　武红江)