用基于RBAC的方法集成遗产系统的访问控制策略

李 寒[1] 郭 禾[2] 王宇新[1] 陆国际[1] 杨元生[1]
(大连理工大学计算机科学与技术学院 大连 116024)[1] (大连理工大学软件学院 大连 116620)[2]

摘 要 访问控制是软件系统的重要安全机制,其目的在于确保系统资源的安全访问。针对多数遗产系统的访问控制不是基于角色的且其实现形式多样,提出了一种基于RBAC的访问控制策略集成方法。该方法将遗产系统中的权限映射为集成系统中的任务,能够在任务树和策略转换规则的基础上使用统一的形式重组访问控制策略。此外,该方法给出了一组用于实现后续授权操作的管理规则。案例分析表明,提出的方法是可行的,能够有效地集成遗产系统的访问控制策略,并将RBAC引入遗产系统的访问控制。

关键词 基于角色的访问控制,访问控制策略,遗产系统,集成,任务
中图法分类号 TP311.5 **文献标识码** A

Using RBAC-based Approach to Integrate Access Control Policies in Legacy Systems

LI Han[1] GUO He[2] WANG Yu-xin[1] LU Guo-ji[1] YANG Yuan-sheng[1]
(School of Computer Science and Technology, Dalian University of Technology, Dalian 116024, China)[1]
(School of Software, Dalian University of Technology, Dalian 116620, China)[2]

Abstract Access control whose objective is to ensure the security of accessing to resources in software systems is an essential part for software systems. As access control policies in legacy systems seldom based on roles are represented in various forms, an RBAC-based approach was proposed to integrate these access control policies. The approach maps permission of legacy systems to tasks of integrated system. Based on task trees and transformation rules of access control policy, various access control policies were reorganized in a unified form. Moreover, management rules were provided to achieve further authorization. A case study is demonstrated to depict the proposed approach is a feasible solution to integrate legacy access control policies and introduce RBAC into legacy systems.

Keywords Role based access control, Access control policy, Legacy system, Integration, Task

1 引言

访问控制指在保障授权用户获取所需资源的同时拒绝非授权用户的非法访问。目前大多数软件系统中都存有大量私密数据,因而访问控制成为必备的安全机制,被应用于近乎所有的软件系统。

遗产系统指由于替换或重新开发成本太高而仍在被使用的软件系统或应用程序[1]。随着实际需求的变化,除需维护和演化单一的遗产系统外,还需集成多个遗产系统。不同的遗产系统往往采用不同的访问控制策略。为减少集成后系统的使用和管理代价,应以统一的标准重组不同形式的访问控制策略。

访问控制策略主要包含强制访问控制(MAC)、自由访问控制(DAC)和基于角色的访问控制(RBAC)。相对于DAC和MAC,RBAC是解决软件系统中存在的资源控制问题的有效方法。使用RBAC可以有效减少50%~90%的管理操作[2],该方法且能大大降低系统的使用和维护开销[3]。RBAC虽被广泛应用于近年开发的软件系统,却未被遗产系统普遍采用。因此,研究基于RBAC访问控制策略集成方法对于遗产系统的演化和集成具有重要的意义。

本文选择RBAC作为访问控制策略的组织标准,提出一种用基于RBAC的方法重组多个遗产系统的访问控制策略。该方法将RBAC引入遗产系统的安全机制,并使用相同的标准组织不同形式的访问控制策略,从而实现了对不同遗产系统的访问控制策略的集成。

本文第2节介绍相关工作;第3节详述提出的方法,包括基本结构、任务提取、策略集成和授权管理;第4节用案例验证方法的可行性;最后给出结论。

2 相关工作

近年来,在RBAC与软件系统的集成方面取得了一些研究成果。2007年,Chen等提出了一种基于代理的面向服务方法[2],该方法将RBAC引入未使用角色进行访问控制的软件系统,能为软件系统提供较高层次上的安全保护。2007

到稿日期:2010-08-12 返修日期:2010-12-06

李 寒(1981-),女,博士生,主要研究方向为软件工程、软件安全、软件测试,E-mail:lihan409@gmail.com;**郭 禾**(1955-),男,教授,主要研究方向为计算机体系结构、软件工程;**王宇新**(1973-),男,讲师,主要研究方向为图像处理、软件工程;**陆国际**(1987-),男,硕士生,主要研究方向为软件工程;**杨元生**(1946-),男,教授,主要研究方向为算法分析与设计。

年，王治刚等研究了不同的安全策略定义语言和方法，提出了一种基于本体的访问控制策略定义机制 OntoRBAC[4]，它能够在语义级别集成不同的访问控制策略。2009 年，Memon 提出了一种将 RBAC 与数据库服务器工具进行集成的方法，以提高 Ad-hoc 网络环境中数据访问的安全级别[5]。同年，万宏辉等使用 Web Service 集成了 Web 环境中分散的访问控制，为多 Web 应用系统提供了整体化的安全访问服务[6]。

此外，RBAC 模型的管理也是访问控制策略集成时备受关注的问题之一[7]。2008 年，Dekker 等提出了一种用于分布式系统的 RBAC 管理模型及其相关的管理过程[8]。为了高效管理不同的访问控制模型，李晓峰等给出了一种通用管理模型[9]。通过引入与实际组织结构相对应的管理空间的概念，该模型具有更易于被管理人员理解的优点。

综上所述，目前大多数的研究均采用在软件系统外部增加功能模块的方法集成访问控制，而着眼于软件系统自身的研究却较少。因此，本文运用反向工程和再工程的思想，从软件系统内部对其访问控制策略进行改造。

3 基于 RBAC 的访问控制策略集成方法

图 1 为集成方法的体系结构，它由如下 4 个模块组成。

1）集成系统：所有待集成遗产系统的集合。

2）任务库：从遗产系统中提取的所有任务的集合。

3）策略库：由一组集合组成，用于描述访问控制策略。

4）管理规则库：涵盖用于管理后续授权操作的规则。

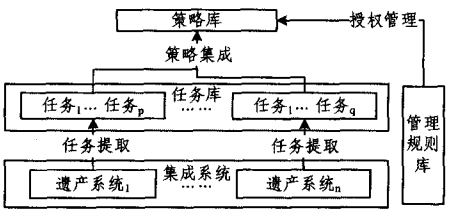

图 1 集成方法的体系结构

除上述模块外，方法还包含联系各模块的 3 个动作。

1）任务提取：负责从遗产系统的源代码中提取任务，并将得到的任务以树形结构进行组织。

2）策略集成：以策略库为目标，重组访问控制策略。

3）授权管理：根据管理规则库中的规则，执行授权操作。

3.1 任务提取

访问控制策略中的权限指操作(action)和对象(object)间的映射(action,object)，在软件系统中常体现为系统功能。在绝大多数软件系统中，系统功能的执行依赖于影响系统执行流程的用户可操作组件的触发，因而可将权限与相应的用户可操作组件及与该组件绑定的事件对应。本文将用户可操作组件定义为交互组件，将这些直接或间接蕴含权限的交互组件及其绑定事件定义为任务。因此，用户与权限的映射关系就转化为用户与任务间的映射。例如，按钮 Btn1 仅具有单击事件，且该事件会触发读取文件 Readme. txt 的操作，则 Btn1 及与 Btn1 绑定的 Click 事件将被视作一个任务，其对应的权限为(Read,Readme. txt)。

一般来说，若软件系统中的某交互组件可用，则与其绑定的所有事件都可用。因此，任务在软件系统中表现为隐含权限的交互组件，任务提取的实质可描述为找出遗产系统中的此类交互组件。需注意的是，由于任务提取依赖于源代码，因此功能相同的任务在不同的开发语言中可能体现为不同的交互组件。以 Button 组件为例，在 C++中表示为 CButton，而在 java 中表示为 JButton。此外，如果待集成的遗产系统是基于网络的，其交互组件大多表现为超级链接。

任务提取包含构造子任务树和合成全局任务树两部分，其步骤如下：

①构造子任务树：负责为每个待集成的遗产系统生成相应的任务树。

(1)生成 UI(User Interface)类图：利用反向工具 EA(Enterprise Architect)为遗产系统生成 UI 类图。

(2)初始化根节点：用遗产系统名作为该遗产系统对应的子任务树的根节点，并定义其为子任务树的第 1 层。

(3)生成第 2 层的节点：令 C_{first} 为用户使用系统时访问的第一个 UI 类，则 C_{first} 所包含的交互组件将作为任务树的第 2 层节点添至任务树。同时，将 C_{first} 加入 UI 类集 $C_{current}$。

(4)生成其余层次的节点：

1)对 $C_{current}$ 中的每个 UI 类 C，根据 UI 类图找到与 C 有单向关联的所有 UI 类，并将这些 UI 类加入 UI 类集 $C_{successor}$。

2)对 $C_{successor}$ 中的每个 UI 类 C'，在 C 中搜索将 C' 实例化的交互组件 Com，并将 C' 所包含的交互组件作为 Com 的子节点添至任务树。

3)将 $C_{current}$ 更新为所有 $C_{successor}$ 的并集。

4)$C_{current}$ 为空，则执行步骤 2。否则，返回步骤 2)继续生成子任务树中的其余节点。

②合成全局任务树：以 IS 为根节点，将每个遗产系统对应的子任务树作为 IS 的孩子，生成集成系统对应的全局任务树。在全局任务树中，IS 为第 0 层。

全局任务树是实现集成系统权限控制的基础。在全局任务树中，对用户权限的控制体现为用户是否有权利使用权限所对应的交互组件。也就是说，具有不同权限的用户将对应全局任务树中不同的子任务树。

3.2 策略集成

策略集成意在将遗产系统中以不同形式存在的访问控制策略用统一的结构进行组织。

3.2.1 策略库

策略库的作用是用统一的形式组织不同类别的访问控制策略。策略库以 RBAC 模型为基础，共设计了 7 个主集合和 3 个辅集合，用以涵盖描述和转换访问控制策略涉及到的元素。主集合包括 LSS 用于保存待集成的遗产系统的信息；U 存储的是用户信息；AU 包含管理员信息，其中的管理员是负责维护系统权限的管理员的简称；R 用来保存集成系统中的角色信息；T 包含任务的信息；UR 中存储的是用户和角色间的映射关系；RT 则保存了角色和任务之间的对应关系。辅集合包括：UG 是用户组的集合；UGT 为用户组和任务间映射关系的集合；UUG 为用户和用户组间映射关系的集合。

此外，对于集成系统 LSS 中的某个遗产系统 ls 来说，ls 对应的集合可表示为 ls. U，ls. AU，ls. R，ls. T，ls. UR，ls. RT，ls. UG，ls. UGT 和 ls. UUG。

为操作上述集合，定义了6个基本操作，其描述如表1所列，其中，$u\in U$，$a\in AU$，$t\in T$，$r\in R$。

表1 基本操作及其描述

基本操作	描述
CreateRole(a,r)	管理员a创建角色r
DeleteRole(a,r)	管理员a删除角色r
AddTask(a,t,r)	管理员a关联任务t与角色r
RemoveTask(a,t,r)	管理员a移除任务t与角色r的关联
AddRole(a,u,r)	管理员a添加角色r给用户u
RemoveRole(a,u,r)	管理员a从用户u的角色中移除角色r

3.2.2 访问控制策略的转换

针对遗产系统中常用的3种访问控制实现方式，给出了相应的转换规则，用来将不同形式的访问控制策略重组为以RBAC为基础的统一结构。

转换规则1 软件系统使用一组预定义的用户名/密码实现访问控制。在这类系统中，用户通过输入设定好的用户名和密码访问遗产系统。针对这种策略，可通过增加单一的角色来解决。该转换规则的形式化描述如下：

CreateRole(a,r)；

$\forall t\in$ ls. T, AddTask(a,t,r)；

$\forall u\in$ ls. U, AddRole(a,u,r)。

转换规则2 软件系统通过预定义用户组实现访问控制。系统包含多个用户组，不同用户组可以具有不同的访问控制权限，只有属于至少一个用户组的用户才能够对系统进行访问。这种策略的转换可通过设置多个角色来实现。其形式化描述如下：

$\forall ug\in$ ls. UG, CreateRole(a,r)；

(ug,t)$\in$ ls. UGT, AddTask(a,t,r)；

(u,ug)$\in$ ls. UUG, AddRole(a,u,r)。

转换规则3 软件系统使用RBAC实现访问控制。针对这种情况，需要用任务表示权限，将遗产访问控制策略用策略库中的标准形式重新组织。

3.3 授权管理

授权管理负责执行集成系统中的授权操作。本节定义了6条授权管理规则，用以处理不同的授权操作，其中谓词Enable(operation)表示允许执行操作operation。

管理规则1 管理员可向其所在遗产系统添加新角色。

$a\in$ ls. AU$\Rightarrow$Enable(CreateRole(a,r))

管理规则2 管理员可删除其所在遗产系统中已有的角色。

$a\in$ ls. AU$\wedge r\in$ ls. R$\Rightarrow$Enable(DeleteRole(a,r))

管理规则3 管理员可为其所在系统中已有的角色添加任务。

$a\in$ ls. AU$\wedge r\in$ ls. R$\wedge t\in$ ls. T$\Rightarrow$Enable(AddTask(a,t,r))

管理规则4 管理员可从其所在系统已有的角色中删除已有任务。

$a\in$ ls. AU$\wedge r\in$ ls. R$\wedge t\in$ ls. T$\wedge$(r,t)$\in$ ls. RT$\Rightarrow$Enable(RemoveTask(a,t,r))

管理规则5 管理员可为其所在系统中的用户添加角色。

$a\in$ ls. AU$\wedge r\in$ ls. R$\wedge u\in$ ls. U$\Rightarrow$Enable(AddRole(a,u,r))

管理规则6 管理员可回收其所在系统中的用户的已有角色。

$a\in$ ls. AU$\wedge r\in$ ls. R$\wedge u\in$ ls. U$\wedge$(u,r)$\in$ ls. UR$\Rightarrow$Enable(RemoveRole(a,u,r))

至此，通过任务提取，集成系统中的任务以树形结构构成任务库。基于该任务库和策略转换规则，不同遗产系统的访问控制策略经策略集成被转换为统一的结构，并保存在策略库中。另外，依据管理规则库中的管理规则，可以通过操作策略库中的集合实现后续的授权管理。

4 案例分析

本节使用本文方法集成某银行两个遗产系统的访问控制策略，以验证方法的可行性：1)资产管理系统(Asset Management System，AMS)，使用VB6和Oracle数据库开发，采用预定义一组用户名和密码的方法实现访问控制；2)消费积分兑换系统(Consumption Point Exchange System，CPES)，由VB.net和Access数据库开发，使用用户组的方式实现系统的访问控制。上述两个系统的功能不同，但由于同属于一家银行，因而用户信息存在重叠，即一个用户对两个系统都具有访问权限。

4.1 提取遗产系统中的任务

首先使用EA工具为两个遗产系统生成各自的UI类图，并以此为基础，分别为CPES和AMS建立对应的子任务树。

图2为CPES系统的UI类图。以CPES为例，初始化子任务树时，系统名CPES被设定为根节点，并标识为子任务树的第1层。由于UI类_Default是系统入口，因而将_Default设为C_{first}。包含在_Default中的交互组件btnSubmit成为任务树的第2层，并将_Default加入至$C_{current}=\{\text{_Default}\}$。此时，$C_{current}$中只有唯一的$C=$_Default。在UI类图中，与$C$具有单向关联的UI类只有Menu，因此$C_{successor}=\{\text{Menu}\}$。因为UI类Menu是在交互组件btnSubmit的Click事件中被实例化的，所以Menu中的所有交互组件将作为btnSubmit的子节点加入到子任务树中，成为子任务树中的第3层节点。与此同时，更新$C_{current}=C_{successor}=\{\text{Menu}\}$。$C_{current}$不为空，算法继续。最终生成的子任务树共有4层，包含25个节点。将任务提取算法应用于AMS系统，得到一个深度为6、内含66个节点的子任务树。合并2棵子任务树后，得到集成系统的全局任务树，如图3所示。

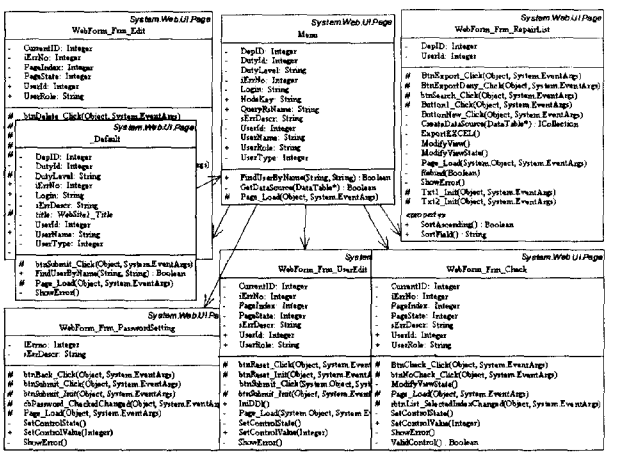

图2 CPES系统的UI类图

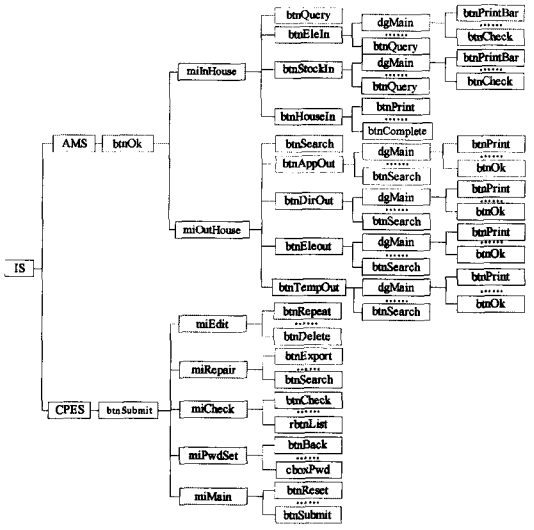

图 3　集成系统的全局任务树

该全局任务树的最大深度为 7，根节点 IS 代表集成系统，它是全局树的第 0 层，其余的节点代表系统任务。任务树中的节点代表任务，它们与权限存在着对应关系。表 2 列出了图 3 中的部分节点（即任务）及其对应的权限。

为了区分同名的不同任务，用（2 级前驱节点名：1 级前驱节点名：节点名）的形式表示任务。例如，IS：AMS：btnOk 表示为 IS 下 AMS 中的任务 btnOk。

表 2　部分任务及对应的权限

任务	权限
IS：AMS：btnOk	Dim frmMainForm As New functionFrm
AMS：btnOk：miInHouse	Dim frmInHouseList As New Frm_inHouseList
AMS：btnOk：miOutHouse	Dim frmOutHouseList As New Frm_outHouseList
btnOk：miInHouse：btnQuery	Select * from 入库单
miInHouse：btnEleIn：btnQuery	Select * from 电子设备入库表

4.2　集成访问控制策略

为了验证访问控制策略的集成，从两个系统中共选取了 7 个用户，包括 2 名系统管理员（假设用户名为 Andy 和 Bob）和 5 名普通用户（假设用户名为 Jack，Joe，Paul，Roy 和 Tom），其中 Bob 既是 AMS 系统的管理员又是 CPES 系统的普通用户；Andy 对 AMS 和 CPES 系统均拥有管理权限；Jack，Paul 和 Roy 是 AMS 系统的普通用户，Joe，Roy 和 Tom 是 CPES 系统的普通用户。

由于 AMS 系统采用预定义一组用户名/密码的方式实现访问策略，即用户只要输入正确的用户名和密码就可以访问整个系统，因此需要创建唯一的角色 r1，并将所有的用户和权限赋予该角色。令 children(N)表示节点 N 的所有后继节点，parents(N)表示节点 N 的所有前驱节点。根据转换规则 1，转换后的策略的形式化描述如下：

LSS＝{AMS}；AMS. AU＝{Andy，Bob}；AMS. R＝{r1}；

AMS. U＝{Jack，Paul，Roy}；

AMS. T＝{btnOk，children(btnOk)}；

AMS. UR＝{(Jack，r1)，(Paul，r1)，(Roy，r1)}；

AMS. RT＝{(r1，btnOk，children(btnOk))}。

CPES 系统采用 3 个用户组（订单录入组 OrderEdit、订单审核组 OrderCheck 和游客组 Guest）实现访问控制。依据转换规则 2，将新增 3 个角色（r2，r3 和 r4）分别对应用户组 OrderEdit，OrderCheck 和 Guest，转换后的策略的形式化描述如下：

LSS＝{AMS，CPES}；CPES. AU＝{Andy}；

CPES. R＝{r2，r3，r4}；CPES. U＝{Bob，Joe，Roy，Tom}；

CPES. T＝{btnSubmit，children(btnSubmit)}；

CPES. UR＝{(Joe，r2)，(Bob，Joe，r3)，(Roy，Tom，r4)}；

CPES. RT＝{(r2，btnSubmit，miEdit，miMain，miRepair，miPwdSet，miUserEdit，children(miPwdSet)，children(miRepair)，children(miEdit)，children(miUserEdit))，(r3，btnSubmit，miCheck，miMain，miRepair，miPwdSet，miUserEdit，children(miCheck)，children(miRepair)，children(miPwdSet)，children(miUserEdit))，(r4，btnSubmit)}。

4.3　管理后续的授权操作

授权操作是访问控制的重要组成部分，因此需要依据3. 3 节定义的管理规则对后续的授权操作进行管理。以在 AMS 系统中增加新角色用以管理电子设备入库出库为例，该角色应拥有对任务 btnEleIn 和 btnEleout 的使用权，应用不同的管理规则，其授权过程如下：

首先，根据管理规则 1 创建新角色 r5：

Andy∈AMS. AU⇒Enable(CreateRole(Andy，r5))

其次，应用管理规则 3 为 r5 关联相应的任务：

$\forall t \in$ {btnEleIn，btnEleout} ∪ parents(btnEleIn) ∪ children(btnEleIn) ∪ parents(btnEleout) ∪ children(btnEleout))，Andy∈AMS. AU∧r1∈AMS. R⇒Enable(AddTask(Andy，t，r5))

授权结束后，AMS. R 和 AMS. RT 被更新，其结果如下：

AMS. R＝{r1，r5}；

AMS. RT＝{(r1，btnOk，children(btnOk))，(r5，btnOk，miInHouse，miOutHouse，btnEleIn，btnEleOut，children(btnEleIn)，children(btnEleOut)}

经过策略集成和授权管理，对用户的权限控制转换为对任务的控制。依据策略库中的信息和任务在软件系统中的表现形式，可以通过动态设置交互组件可用性或可视性的方法实现用户对任务的访问控制。总之，提出的方法是可行的，它能够有效地集成遗产系统的访问控制策略。此外，它还可以作为新系统访问控制机制的设计依据。

结束语　访问控制策略的集成是遗产系统集成的关键问题之一。针对遗产系统中多种形式的访问控制策略，本文提出了一种基于 RBAC 的访问控制策略集成方法。该方法研究了遗产系统中的权限与集成系统中的任务之间的对应关系，给出了一种统一结构用以重组各种访问控制策略，并定义了 3 条访问控制策略转换规则和 6 条用于授权操作的管理规则。本文案例以某银行的子系统集成为背景，结果表明提出的方法是有效且可行的，它不仅能实现策略集成，还通过 RBAC 的引入改进了遗产系统访问控制机制。

参 考 文 献

[1]　Bennett K H. Legacy system：coping with success[J]. IEEE Software，1995，12(1)：19-23

（下转第 138 页）

(Ⅱ) $I(P)=\{I_1\}$, $\bar{I}(P)=\{I_2\}$

仅构件集$\{C_9, C_2, C_3\}$被激活执行，产生输出结果O_1，因此软件可靠性为：

$R(P)=R(C_1)\cdot R(C_2)\cdot R(C_3)=0.857$

从计算结果可知，在不同的数据输入接口集上加载有效数据，软件执行的功能和数据流激活的构件集不同，可使软件运行时的可靠性出现较大差异。

4.2 讨论、分析

软件运行时的实际可靠性与各构件输入域划分、操作剖面传递及有效输入接口集合存在密切关系。

(1)输入子域设置的影响

输入子域数设置太少，难以反映软件的实际可靠性。当缩减到单个输入子域时，实际可靠性退化为测试可靠性。输入子域划分越多、越细，操作剖面就越能反映实际运行的输入数据分布，实际可靠性计算结果就越准确。但划分太多子域，会增加开发者构件测试工作量及操作剖面传递计算的难度，因此需要对输入子域的划分粒度进行折中。

(2)操作剖面传递精确性的影响

要有效计算软件运行时的实际可靠性，应对构件操作剖面进行准确传递。精确地计算构件的输出剖面，不但需要获得构件功能函数的单调性、连续性、极大极小值等特性，还需要实际运行构件，计算以子域边界点作为输入的输出。划分过细的输入子域，可能给输出剖面的计算带来困难。应适当控制输入剖面子域数量，并采用近似简便方法估算构件输出剖面，以利于控制问题的复杂度。

(3)有效输入接口集合的影响

输入接口上的有效数据驱动构件的执行，加载有效数据的输入接口的不同可对软件可靠性造成影响，因此准确分析构件间的数据流关系及构件执行频率，是影响模型精确性的关键因素之一。

结束语 本文将基于频率统计和操作剖面的方法相结合，提出评估基于构件数据流软件实际可靠性的模型。将基于构件的数据流结构划分为由基本结构组合节点构成的多层嵌套结构。在每个层次上，根据构件间的数据流及控制流关系，估算实际运行中被激活的构件集合，并统计其执行频率。利用基于深度遍历的递归算法思想，逐层计算各级组合节点的可靠性，能反映软件实际运行中输入接口的有效数据就绪状态及由此引起的数据流关系的动态变化。在递归算法中融入操作剖面传递过程，计算各层次构件的输入剖面，并与按子域表示的构件可靠性相结合估算构件、基本结构、组合节点和整个程序的实际可靠性，使之反映软件实际运行时输入数据的分布特性。

软件可靠性计算结果对操作剖面的反映程度取决于输入子域划分的精细度与操作剖面传递的精确性，而这些都与具体应用逻辑密切相关。细分的输入子域和精确的剖面传递要求可使操作剖面计算及传递变得复杂，因而应根据具体构件逻辑及可靠性要求进行折中。

参 考 文 献

[1] Lyu M R. Software Reliability Engineering: A Roadmap [C] // Proc of Future of Software Engineering. Washington: IEEE Computer Society, 2007: 153-170

[2] 谢景燕，安金霞，朱纪洪. 考虑不完美排错情况的 NHPP 类软件可靠性增长模型[J]. 软件学报，2010，21(5)：942-949

[3] 雷航，马成功. Markov 模型的软件可靠性测试充分性问题的研究[J]. 电子科技大学学报，2010，39(1)：101-105

[4] Wang Wen-li, Pan Dai, Chen Meihwa. Architecture-based software reliability modeling [J]. Journal of Systems and Software, 2006, 79(1): 132-146

[5] 詹涛，周兴社，符宁，等. 软件能力可信研究综述[J]. 小型微型计算机系统，2008，29(5)：786-792

[6] Katerian G P, Trivedi K S. Architecture-based approaches to software reliability prediction[J]. Computers and Mathematics with Applications, 2003, 46: 1023-1036

[7] 毛晓光，邓勇进. 基于构件软件的可靠性通用模型[J]. 软件学报，2004，15(1)：27-32

[8] 万晓民，张德平，聂长海，等. 统计测试中操作剖面的一种优化设计方法[J]. 东南大学学报：自然科学版，2008，138(12)：233-238

[9] Johnston W M, Hanna J R P, Millar R J. Advances in Dataflow Programming Languages[J]. ACM Computing Surveys, 2004, 36(1): 1-34

[10] LABVIEW. 2000. LabView User Manual[OL]. ustin, TX: National Instruments, A2000

[11] Wesley M, Hanna P J R, Richard M J. Advances in dataflow programming languages[J]. Advances in Dataflow Programming Languages, 2004, 36(1): 1-34

[12] Jurij S, Borut R, Theo U. Asynchrony in parallel computing: From dataflow to multithreading[J]. Journal of Parallel and Distributed Computing Practices, 1998, 1: 1-33

(上接第 129 页)

[2] Chen F, Li S, Yang H. Enforcing role-based access controls in software systems with an agent based service oriented approach [C] // Proceedings of 2007 IEEE International Conference on Networking, Sensing and Control. London, UK: IEEE Computer Society, 2007: 15-17

[3] Bertino E. RBAC models-concepts and trends[J]. Computers & Security, 2003, 22(6): 511-514

[4] 王治刚，王小刚，卢正鼎，等. OntoRBAC：基于本体的 RBAC 策略描述与集成[J]. 计算机科学，2007，34(2)：82-85

[5] Memon Q A. Implementing role based access in healthcare Adhoc networds[J]. Journal of Networks, 2009, 4(3): 192-199

[6] 万宏辉，朱更明. 基于 Web Service 的多 Web 应用系统访问控制集成[J]. 计算机应用与软件，2009，26(7)：28-30

[7] Li N, Mao Z. Administration in role-based access control[C] // Proceedings of 2nd ACM Symposium on Information, Computer and Communications Security. Singapore: ACM New York, 2007: 127-138

[8] Dekker M A C, Crampton J, Etalle S. RBAC administration in distributed systems[C] // Proceedings of 13th ACM Symposium on Access Control Models and Technologies. Estes Park, United States: ACM New York, 2008: 93-101

[9] 李晓峰，冯登国，徐震. 一种通用访问控制管理模型[J]. 计算机研究与发展，2007，44(6)：947-957