UML模型和Java代码之间的一致性检测方法

曾 一[1,2] 李函逾[1] 刘慧君[1,2] 余双双[1] 周 波[1]
(重庆大学计算机学院 重庆 400030)[1] (软件理论与技术重庆市重点实验室 重庆 400030)[2]

摘 要 针对代码与模型之间的不一致性问题,提出了一种基于UML模型和Java代码之间的一致性检测方法。首先,对UML类图和时序图进行形式化描述,并提出时序调用图(SD-CG)这一概念,在此基础上完成类的关联关系到关联属性的转换以及UML时序图到时序调用图SD-CG的转换;其次,通过方法调用图CG来表达类方法之间的调用关系,从而反映代码动态行为,由此通过对Java源代码的词法分析与语法分析,可获得类的信息及方法调用图CG;然后设计了UML模型与Java源代码间一致性检测算法,包括对类间静态信息以及时序调用图SD-CG与方法调用图CG间的一致性检测;最后,通过开发UML模型与Java源代码一致性检测工具,验证了所提出的方法是可行有效的。

关键词 UML模型,Java代码,时序调用图,方法调用图,一致性检测
中图法分类号 TP311.5 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2015.4.030

Consistency Detection Method between UML Model and Java Source Code

ZENG Yi[1,2] LI Han-yu[1] LIU Hui-jun[1,2] YU Shuang-shuang[1] ZHOU Bo[1]
(College of Computer Science, Chongqing University, Chongqing 400030, China)[1]
(Key Laboratory of Software Theory and Technology in Chongqing, Chongqing 400030, China)[2]

Abstract Aiming at the problem of inconsistencies between the code and the model, this paper presented a consistency detection method based on UML model and the JAVA code. Firstly, the UML class diagram and sequence diagram are formalized and a concept of sequence diagram call graph(SD-CG) is put forward. On this basis, the transformation from the associated relationship between classes to the associated attribute of class and the transformation from the sequence diagram to sequence diagram call graph are completed. Thirdly, based on call graph which is made up of class methods being as nodes and the call relation between them being as edges, to analyze the dynamic behavior in the source code, this paper got the information of Java classes and the method call graph (CG) by the lexical analysis and syntax analysis of Java source code. Then the consistency detection algorithm between UML model and Java source code was designed, including the static information consistency detection and the interactive information consistency detection. At last, we developed a prototype tool which can complete information consistency detection to verify the feasibility and effectiveness of the proposed method.

Keywords UML model, Java code, Sequence diagram call graph(SD-CG), Call graph(CG), Consistency detection

"不一致性是普遍存在的"这个观点目前已经得到广泛的共识[1-3]。而在软件开发过程中,软件测试和软件维护占有很大的比例,而模型与代码之间保持一致对软件尤其是大型软件的后期维护是很有帮助的。所以,一致性检测方法可以作为软件测试的一部分,提高对代码分析的效率,便于测试系统的整体性能。另外,目前国内外对于一致性检测的算法研究和代码分析的研究有很多,但对模型与代码之间一致性检测的研究相对较少。因此,将UML模型与代码进行一致性检测具有一定的实用性与创新性。

目前对于不一致的处理策略有很多,如基于经典罗和基于目的的方法等。而本文采用基于模型检验的方法[4-8],对模型和代码之间的一致性方法进行研究。

本文通过解析UML模型、提取源代码的关键信息、确定一致性检测算法,最终完成UML模型和Java代码间的一致性检测,并形成一致性检测工具。在软件开发过程中,它作为代码复查的一个辅助手段,能提高工作效率,便于系统维护。

1 模型与代码一致性检测的基本流程

本文UML模型与Java代码的一致性检测主要基于两个假设:1)UML各个模型(本文重点使用UML类图和时序图)是正确的且彼此之间是一致的;2)当UML模型与Java代码存在不一致信息时,以UML模型为依据输出不一致报告。UML模型与Java代码间一致性的检测研究的具体流程如图1所示。

到稿日期:2014-06-05 返修日期:2014-09-03 本文受国家自然科学基金:自然最近邻居的特征分析与应用研究(61272194)资助。
曾 一(1961-),男,教授,CCF会员,主要研究方向为软件工程理论及应用、软件度量与软件测试、软件模型与软件工具、面向服务计算;**李函逾**(1990-),男,硕士生,主要研究方向为软件测试;**刘慧君**(1975-),男,副教授,主要研究方向为数据挖掘、电子商务、Web技术;**余双双**(1991-),女,硕士生,主要研究方向为软件测试;**周 波**(1990-),男,硕士生,主要研究方向为面向服务计算。

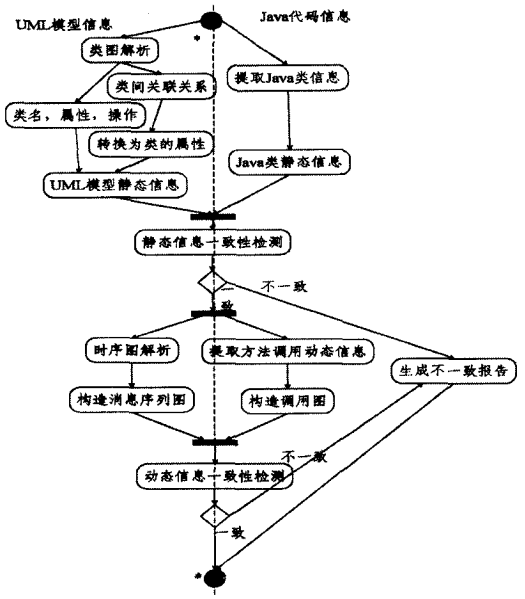

图 1　UML 模型和代码一致性检测的基本流程

2　UML 模型预处理

2.1　UML 模型的形式化定义

UML 模型的半形式化特性使得无法直接对 UML 模型进行信息的提取。因此需要对 UML 模型以数学的方式进行形式化定义描述。

(1)类元组(Class Tuple,CT)[9]是面向对象中类的数学表示形式,类图中一个类就可以表示成一个类元组,类图表示成多个类元组的集合。类元组由类名、类属性以及类方法等基本信息组成。形式化定义如下:

$CT(C) ::= \langle className, \{\langle Attribute\rangle\}, \{\langle Method\rangle\}\rangle$

$Attribute ::= \langle atrrName, Type, Default\ Value\rangle$

$Method ::= \langle methName, \{\langle Parameter\rangle\}, returnType\rangle$

$Parameter ::= \langle paraName, Type, DefaultValue\rangle$

(2)时序图形式化定义一个六元组[10] $SD=\langle O, M, E, \rightarrow, msg, obj\rangle$

①$O=\{O_1, O_2 \cdots O_m\}$是对象的集合。$O_1, O_2 \cdots O_m$ 都是时序图中的对象。

②$M=\{msg_1, msg_2, \cdots, msg_m\}$是消息的集合。消息 msg 定义如下:

$msg ::= \langle msgName, Visibility, \{\langle Parameter\rangle\}, returnType\rangle$

③$E=M\times\{s,r\}$是事件集合。事件是指消息的发送和接收。对于消息 msg,发送事件用$\langle msg, s\rangle$来表示,接收事件用$\langle msg, r\rangle$来表示。

④$\rightarrow$是消息集合 M 上的一个全序关系,表示时序图中的消息在纵向时间轴上的先后关系。

⑤msg 是从 E 到 M 的一个函数关系,$msg(e)\in M$ 表示事件 e 所对应的消息。

⑥obj 是从 E 到 O 的一个函数关系,$obj(e)\in O$ 表示事件 e 所对应的对象。对象 O_i 上所有事件的集合记为 E_i,$E_i=\{e \mid e\in E \wedge obj(e)=O_i\}$。

(3)时序调用图(Sequence Diagram-Call Graph,SD-CG)是一个有向无环图,是通过时序图转换而来的调用树。$SD\text{-}CG ::= \langle M_C, E\rangle$:

①M 是消息的集合。其中,$smsg$ 是指时序调用图的开始节点。

②$cmsg_i ::= \langle msgName, Visibility, \{\langle Parameter\rangle\}, returnType, className\rangle$。

③$E ::= M_C \times M_C$ 表示消息节点间对应的边的集合,消息调用边均为有向边。如$\langle cmsg_1, cmsg_2\rangle$表示调用消息 $cmsg_1$ 对 $cmsg_2$ 有调用关系,且调用边由 $cmsg_1$ 指向 $cmsg_2$。

2.2　UML 类图信息解析

对于类图信息,需要解析类图的 XMI 文件,获取类中类名、属性、方法以及类间关联关系等信息。类图解析的关键在于类间的关联关系到关系属性的转换。类之间的关联关系的模式有 4 种,本文采取的具体转换规则如下:

(1)如果为 1 对 1:类 A 中增加关系属性 B b;类 B 中增加关系属性 A a;

(2)如果为 1 对多:类 A 中增加关系属性 set〈B〉 bs;类 B 中增加关系属性 A a;

(3)如果为多对 1:类 A 中增加关系属性 B b;类 B 中增加关系属性 set〈A〉 as;

(4)如果为多对多:类 A 中增加关系属性 set〈B〉 bs;类 B 中增加关系属性 set〈A〉 as。

2.3　UML 时序图信息解析

对于时序图信息,同样需要对时序图的 XMI 文件进行解析。提取其中的消息、消息时间、发送接收消息对象、对象所属类、消息时间顺序等信息,从而得到时序图 SD,然后再将其转换为时序调用图 SD-CG,具体转换规则如下:

①时序图 SD 中第一个消息("→"关系最左边的消息)为 $smsg$。

②SD 的消息 msg_i 发送对象所属类为 A,接收对象所属的类为 B。根据时序图中消息所属类为接收对象所属的类这一规则,将类 B 的 className 加入 msg_i,将其转换为 SD-CG 中的调用消息 msg_i。

③SD 中 msg_1 的发送对象所属的类与 SD-CG 中 $smsg$ 所属的类相同。规定在 SD-CG 中调用消息节点 $smsg$ 和 $cmsg_1$ 存在调用关系边,且由 $smsg$ 指向 $cmsg_1$。

④SD 消息 msg_i 的接收对象与消息 msg_j 的发送对象是同一对象 O_m,则在 SD-CG 中调用消息节点 $cmsg_i$ 和 $cmsg_j$ 存在调用关系边,且由 $cmsg_i$ 指向 $cmsg_j$。

时序图与时序调用图的对应关系如图 2 所示。

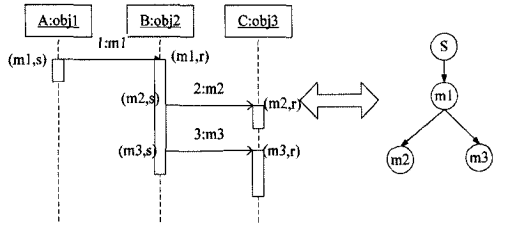

图 2　时序图与时序调用图的对应关系

3　Java 代码信息预处理

本文通过词法分析和语法分析来提取 Java 代码中的关键信息,完成 Java 代码预处理[11,12]。同时设计 Java 类存储

结构和调用图，分别用来存储静态和动态信息。

3.1 代码静态信息预处理

Java 代码的静态结构信息主要是 Java 类的基本信息，如类名、属性、方法等基本信息[13,14]。图 3 给出了 Java 代码中类的数据存储结构。

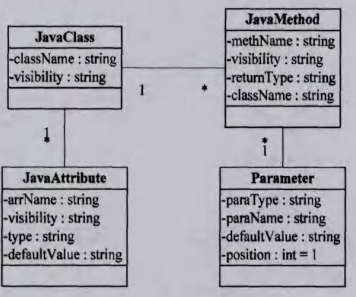

图 3 Java 类的存储结构

Java 代码静态信息提取的主要思想是通过词法分析和语法分析来获取 Java 类的名称、属性序列和方法序列，并将这些信息存储到上面定义的数据存储结构中。

3.2 代码动态信息预处理

对于 Java 代码动态信息提取方法，首先，通过词法分析和语法分析获得调用图节点的基本信息；然后，分析方法间的调用关系；最终构造 Java 调用图 CG。方法调用图 CG 信息存储的数据结构，其本质是一棵调用树，Java 类中的方法作为 CG 的节点，方法间的调用作为 CG 的边，即父节点和子节点之间存在调用关系，父节点调用子节点。下面给出方法调用图 CG 的形式化定义：

$G_c=\langle N,E,r\rangle$，若调用图 CG 为空，用 NULL 来表示，其非空元素定义如下：

(1)$N=\{m_1,\cdots,m_n\}$是节点的集合，即系统中的程序、类或程序单元(如方法、过程等)集。

(2) $m_i=\langle mName, Visibility, \{\langle parameter\rangle\}, returnType, className\rangle$，$m_i$ 是 Java 代码中的某个方法，其中 *className* 是指该方法所属类的名称。

(3)$E\subseteq N\times N$ 是有向边的集合。若$\langle m_i,m_j\rangle\in E$，表示方法 m_i 对方法 m_j 存在调用关系，记做 $m_i\rightarrow m_j$。若方法 m_i 对方法 m_j 有多次调用，无需重复记录，只记录一次即可。

(4)$r\in N$ 代表调用图的第一个节点，即根节点。该节点为空节点，无实际意义。

Java 代码动态信息的提取方法：首先，通过词法分析和语法分析获得 CG 节点的基本信息；然后，分析方法间的调用关系；最终构造 Java 调用图 CG。Java 调用图 CG 的数据存储结构如图 4 所示。

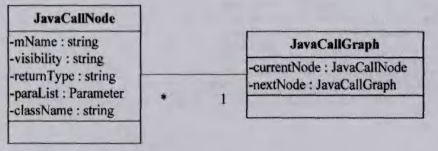

图 4 Java 调用图的存储结构

4 UML 模型与代码一致性检测

4.1 静态信息一致性检测

静态信息包括类名、类的属性集合以及类的方法集合等。表 1 展示了二者之间的映射关系。

表 1 UML 类与 Java 类之间的映射关系

UML 类		Java 类	
类名		类名	
属性		属性	
关联属性			
可见性		可见性	
操作	可见性	可见性	方法
	名称	名称	
	返回值类型	返回值类型	
	参数列表	参数列表	

一致性检测是将 UML 模型预处理后的类图信息与 Java 代码预处理后的 Java 类信息作为静态检测的输入，依次检测类的个数、类名、属性、方法、方法的参数列表的一致性，若有出现不一致的情况，直接输出检测报告，停止检测；对于默认值(defaultValue)不一致的情况只返回警告(Warning)，不终止静态信息的一致性检测，最后输出检测结果的一致性报告。

4.2 动态信息一致性检测

UML 模型与 Java 代码动态信息的一致性检测主要是将时序调用图 SD-CG 与 Java 调用图 CG 之间的信息进行一致性匹配。下面给出动态信息的一致性检测流程。

①提取时序调用图 SD-CG 的消息集合 M_c，构造类的集合 $CT_{SD\text{-}CG}$。

②构造消息与类间的部分关系集合 P_m，如$\langle msg,C\rangle\in P$，表示消息 *msg* 的 *className* 属性值与类 C 的 *className* 属性值相同，即消息 *msg* 属于类 *C*。

③提取时序调用图 SD-CG 的消息调用集 *E*，构造时序调用图 SD-CG 的类间的交互关系集合 $C_{SD\text{-}CG}$。

④提取 Java 调用图 CG 的方法集合 *N*，构造类的集合 CT_{CG}。

⑤构造消息与类间的部分关系集合 P_N。

⑥提取 Java 调用图 CG 的方法调用集 *E*。构造时序调用图 SD-CG 的类间的交互关系集合 C_{CG}。

⑦将由时序调用图 SD-CG 构造的类的交互图 $C_{SD\text{-}CG}$ 与由 Java 调用图 CG 构造的类的交互图 C_{CG} 进行一致性匹配。

⑧将消息节点集合 *Mc* 与方法节点集合 *E* 进行一致性匹配。

⑨将 SD-CG 与 Java 调用图 CG 进行一致性匹配，进行一致性分析，输出一致性检测报告。

接下来将详细介绍动态检测的重点——调用图一致性检测的基本算法：先检测消息(方法)节点的一致性，再通过深度优先算法依次遍历节点间的调用关系，检测调用图之间的一致性，并输出一致性检测报告。

调用图一致性检测算法 AnalysisDynamicM(SD-CG G_S, CG G_C)

输入：时序调用图 G_S 和 Java 调用图 G_C

输出：一致性检测结果 message

将 G_S 中的消息集合 Mc 按照消息名称进行排序，c1=Mc [0]；

将 G_C 中的方法集合 N 按照方法名进行排序，c2=N [0]；

while (c1 != null && c2 !=null)

do

if (c1 与 c2 or c1 的 className 属性值与 c2 的不一致)

message="Error：时序调用图 SD-CG 与调用图 CG 中对应的方法不一致"；

return message;

c1=Mc [i++];

c2=N [i++];

done

c1=时序调用图 G_S 的开始节点;

c2=Java 调用图 G_C 的开始节点;

建立堆栈 t1 和 t2 分别用于存储 SD-CG 和 CG 中的消息或方法;

while (c1 != null && c2 !=null)

cc1=G_S 关系 E 中 c1 指向的下一个消息;

cc2=G_C 关系 E 中 c2 指向的下一个方法;

if cc1=null && cc2 =null then

c1=t1 的出栈元素;

c2=t2 的出栈元素;

else if cc1 与 cc2 是不同的方法 then

message="Error:时序调用图 SD-CG 与调用图 CG 中存在方法"+c1+"所属类"+c1.className+"间交互关系不一致";

return message;

else

将 c1 压入栈 t1 and 将 c2 压入栈 t2;

c1=cc1 and c2=cc2;

done

return message="OK";

5 实验验证

现以网上购书系统为例,根据该策略验证本文提出的一致性检测方法。利用 ArgoUML 进行建模,采用 Myeclipse 开发平台、JDK1.6 作为开发环境,以 Java 为开发语言,来完成 UML 模型与 Java 代码间的一致性检测。此处需要申明的是本文研究的前提为 UML 图形为正确的,也就是说当检测出不一致时,以模型信息为准对代码信息进行修改。

网上购书系统类图中一共包含了 4 个类:BookDB、BookDetails、ShoppingCart、ShopingCartItem,如图 5 所示。

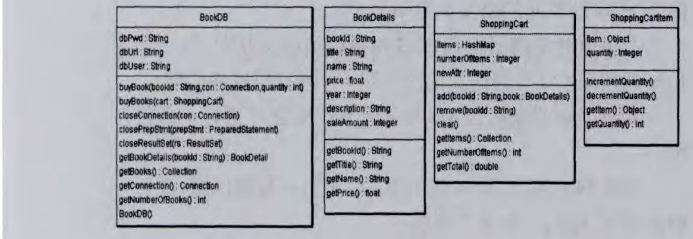

图 5 类图

图 6 给出了网上购书时查询购物车里图书过程的时序图。

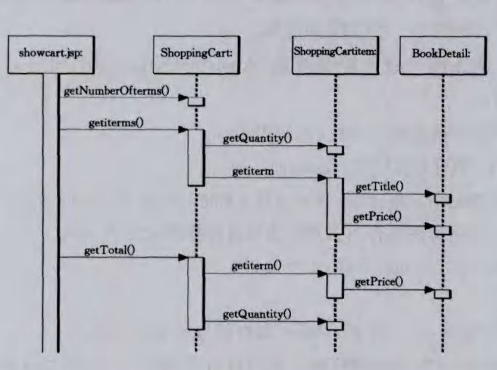

图 6 时序图

工具会将 UML 模型信息提取结果与代码信息提取结果进行检测,只有静态信息检测一致时,才会将 UML 时序图转换成时序调用图,完成之后的 UML 模型与 Java 代码动态信息的检测。而时序调用图的转换结果如图 7 所示。

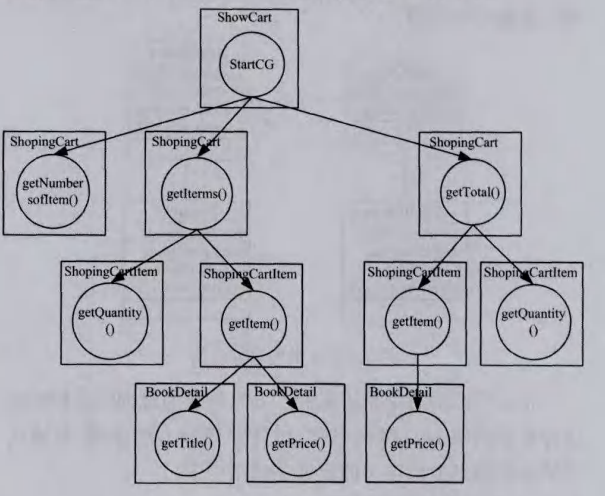

图 7 时序调用图信息

图 8 给出了检测过程中出现不一致信息返回的结果,而图 9 则为信息检测一致时的结果。注意,只有当静态信息检测一致时才继续进行动态信息检测,否则结束检测并返回不一致信息。

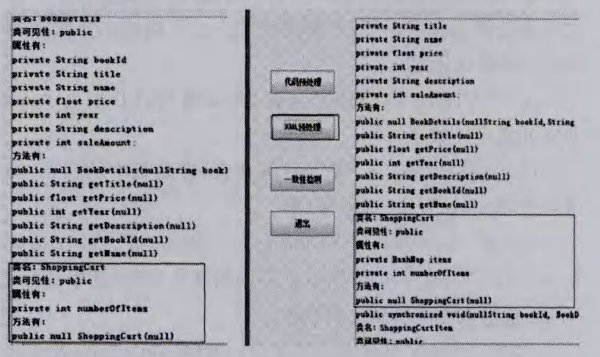

图 8 信息检测不一致

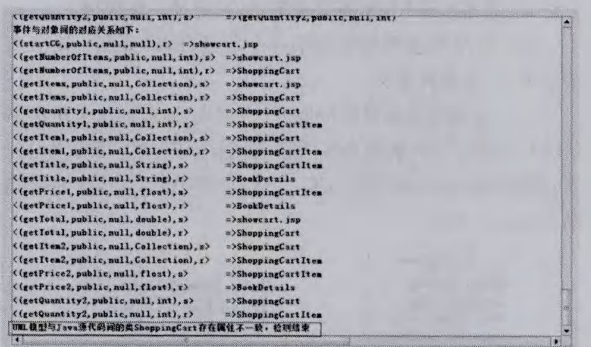

图 9 一致性检测结果

分析上述实验结果可知,UML 模型与代码间一致性检测研究是有效的,这能够有效地减少人工检测的工作。

结束语 本文以 UML 模型和 Java 源代码作为输入,先分别对 UML 和 Java 源代码进行预处理,然后对其中间结果的静态信息和动态信息交互进行一致性检测。在分析了

UML 类图、时序图结构的基础上，对时序图模型用数学方法进行抽象表示，给出其形式化定义；提取 UML 类图和 Java 源代码中类基本信息，完成对 UML 模型与 Java 源代码间静态信息的一致性检测；提出了方法(消息)调用图这一概念，提取 UML 时序图信息，并将其转化构造成时序调用图 SD-CG，通过词法分析与语法分析提取 Java 源代码中的方法调用图 CG，最终完成 UML 模型与代码间动态交互信息的一致性检测。

但是该方法还存在以下一些不足：1)在进行一致性检测时，如果出现不一致信息就会终止检测，因此，后续工作可以在此基础上进行改进，使得一致性检测工作能够最终得到一个包含完整的不一致信息的报告清单；2)在研究动态交互信息时提出的方法调用图在对 Java 源代码生成调用图分析上并不全面，未考虑 Java 的继承性与多态性，后续工作可以充分研究类间信息交互，增加对 Java 多态与继承等特性的考虑。

参 考 文 献

[1] Balzer R. "Tolerating Inconsistency" revisited[C]// Proc. of the 23rd Int'l Conf. on Software Engineering. Toronto: IEEE Computer Press, 2001: 665

[2] Ghezzi C, Nuseibeh B. Guest editorial: Introduction to the special section[J]. IEEE Trans. on Software Engineering, 1999, 25(6): 782-783

[3] Easterbrook S, Chechik M. Int'l workshop on living with inconsistency[C]// Proc. of the 23rd Int'l Conf. on Software Engineering. Toronto: IEEE Computer Press, 2001: 749-750

[4] Clarke E, Grumberg O, Long D. Verification tools for finite-state concurrent systems[C]// Lecture Notes in Computer Science 803. London: Springer-Verlag, 1994: 124- 175

[5] Holzmann J. The model checker SPIN[J]. IEEE Trans. on Software Engineering, 1997, 23(5): 279-95

[6] Heitmeyer C, Jeffords R, Kiskis D. Automated consistency checking requirements specifications[J]. ACM Trans. on Software Engineering and Methodology, 1996, 5(3): 231-261

[7] Chan W, Anderson R, Beame P, et al. Model checking large software specifications[J]. IEEE Trans. on Software Engineering, 1998, 24(7): 498-519

[8] Atlee J, Gannon J. State-Based model checking of event-driven system requirements[J]. IEEE Trans. on Software Engineering, 1993, 19(1): 24-40

[9] 丁娜. 带 OCL 约束的活动图多态测试方法的研究[D]. 重庆：重庆大学，2012

[10] 逄瑞娟. 基于 UML 顺序图的场景测试用例生成研究[D]. 青岛：青岛大学，2007

[11] 赵平. Java 源代码静态分析系统的设计与实现[D]. 长春：吉林大学，2013

[12] 章程. 基于机器学习和程序分析相结合的程序调用技术研究[D]. 上海：上海交通大学，2012

[13] 张健. 精确的程序静态分析[J]. 计算机学报，2008, 31(9): 1550-1555

[14] 逄龙，王甜甜，苏小红，等. 支持多程序语言的静态信息提取方法[J]. 哈尔滨工业大学学报，2011(3): 62-66

(上接第 135 页)

参 考 文 献

[1] 孟芳慧，曹宝香，杨义先. 钮心忻多媒体数字产品版权保护模型研究与设计[J]. 计算机科学，2013, 40(1): 98-102

[2] 张硕，马兆丰，芦效峰，等. 音乐内容动态加密与许可授权系统设计与实现[J]. 计算机科学，2011, 38(12): 43-48

[3] 锁琰，徐小岩，张毓森，等. 支持组件动态更新的远程证明[J]. 西安电子科技大学学报，2012, 38(4): 11-19

[4] Park J, Sandhu R. The $UCON_{ABC}$ usage control model[J]. ACM Transactions on Information and System Security (TISSEC), 2004, 7(1): 128-174

[5] Zhang Z, Yang L, Pei Q, et al. Research on usage control model with delegation characteristics based on OM-AM methodology [C]// IFIP International Conference on Network and Parallel Computing Workshops, 2007 (NPC Workshops). IEEE, 2007: 238-243

[6] Hu X L, Osborn S L. A new approach for delegation in usage control[C]// Proceedings of the third ACM conference on Data and application security and privacy. ACM, 2013: 269-276

[7] Lei Jian-yun. Weighted Directed Graph-Based Authorization Delegation Model[J]. Journal of Networks, 2013, 8(12): 2812-2815

[8] Gaaloul K, Proper H A, Charoy F. Delegation Protocols in Human-Centric Workflows[C]// Proceedings 13th IEEE International Conference on Commerce and Enterprise Computing 2011 (CEC 2011). New Jersey, NJ: IEEE Computer Society, 2011: 219-224

[9] Sun Dao-qing. UCSSDAP: Ubiquitous Computing Service Security Delegation Authorization Protocol[C]// 2011 IEEE International Conference on Automation and Logistics (ICAL 2011). New Jersey, NJ: IEEE Computer Society, 2011: 371-374

[10] Osborn S L, He Wang. A Survey of Delegation from an RBAC Perspective[J]. Journal of Software, 2013, 8(2): 266-275

[11] 冯雪，俞银燕，汤帜. 具有硬件适应性的多设备内容共享与版权保护方法[J]. 北京大学学报：自然科学版，2011, 47(6): 1009-1016

[12] Zhang Yong, Xiang Xue, Hai Feng, et al. An anonymous remote attestation for trusted cloud computing[C]// Proceedings 2012 IEEE 2nd International Conference on Cloud Computing and Intelligence Systems 2012. (IEEE CCIS 2012) Washington, DC: IEEE Computer Society, 2012: 426-429

[13] Yu Yue, Wang Huai-min, Liu Bo, et al. A Trusted remote attestation model based on trusted computing[C]// 2013 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications, 2013. Washington, DC: IEEE Computer Society, 2013: 1504-1509

[14] Guo Jia-ming, Wei Jiang. Analysis and research of remote attestation based on trusted computing[C]// 2013 Fourth International Conference on Digital Manufacturing & Automation, 2013. Washington, DC: IEEE Computer Society, 2013: 192-195

[15] Li Ya-ping, Zhou Wei-liang. Research on the delegation schemes of the $UCON_{ABC}$[J]. Journal of University of Science and Technology of China, 2012, 42(2): 154-160