# 基于多视角卡牌模型的需求缺陷检测

苏 若 吴 际 刘 超 杨海燕
(北京航空航天大学计算机学院 北京 100191)

**摘 要** 需求来源于不同利益相关方对现实系统的认识和期望。需求获取在整个软件产品的研发过程中至关重要,往往决定着软件产品的质量甚至成败。然而,由于各种复杂因素的影响,获取到的需求中往往存在不完整、不准确甚至冲突等缺陷。需求表达上的二义性、需求描述的不完整和不一致等是最常见的需求缺陷。文中提出一种基于多视角需求获取的卡牌模型和需求缺陷检测规则。在需求获取过程中,特别是在其初期,其能够发现来自各方需求信息中常见的不完整和不一致需求缺陷。最后,通过 3 组项目案例验证了方法的有效性。

**关键词** 多视角卡牌模型,需求获取,需求缺陷检测
**中图法分类号** TP311 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2018.10.034

## Requirement Defect Detection Based on Multi-view Card Model

SU Ruo WU Ji LIU Chao YANG Hai-yan
(School of Computer Science and Engineering, Beihang University, Beijing 100191, China)

**Abstract** Requirement stems from the understanding and expectations of different stakeholders on the real system. Requirement elicitation is crucial in the whole process of software product development, and it often decides the quality of software product, even its success or failure. Due to the influence of various complex factors in the elicited requirements, there are some defects such as incompleteness, inaccuracy and conflict. The ambiguity of requirement expression and the incompleteness and inconsistency of requirement description are the most common requirement defects. This paper proposed a card model based on multi-view requirements and requirement defect detection rules. In the process of requirement acquisition, especially in the early period, the incomplete and inconsistent requirement defects from stakeholders can be found through requirement defect detection rules. Finally, the validity of this method is verified by the experiment on three project cases.

**Keywords** Multi-view card model, Requirement elicitation, Requirement defect detection

## 1 前言

多视角需求获取强调每个视角都有对软件系统的独到理解,并有效结合来自不同视角的需求,通过多个视角的相互配合和补充来减小某些重要需求被遗漏的可能。每个视角只关心某些特定的需要或要求,降低了需求获取和描述的难度。相比从某一个角度考虑问题,多视角的形成对软件系统需求的刻画更全面、准确;同时,它使整个需求文档的结构性更强、更容易修改和扩充。

很多研究表明,软件产品中发现的缺陷有 40%~50%是在需求阶段埋下的"祸根"[1]。在需求识别和捕捉过程中产生的由于对系统目标、用户需求的错误理解而导致的需求缺陷,一直是公认的难以检测和处理的需求缺陷[2]。本质上,需求识别和捕捉是多种角色的人员在给定目标下的智能互动过程,包括提问、回答、补充意见、提供证据、推理论证和决策等。产生需求理解问题的根本原因包括:1)需求识别和捕捉过程主要依赖于分析人员的经验,带有强烈的个人色彩,因此容易产生因个人知识背景、思维习惯等不同而导致的理解偏差问题;2)需求识别和捕捉过程需要多种角色的人员之间进行频繁的交互和协同,缺少针对性的支持手段来管理交互和协同过程,使得信息在传递过程中容易歧变且不易被人识别;3)需求识别和捕捉过程中的交互协同会产生大量的信息,如分析人员与用户的讨论、分析人员之间的讨论等,没有有效的针对性方法来记录和分析这些交互信息,从而失去了发现理解偏差的重要机会[2]。

在需求缺陷检测方面,很多研究都涉及项目的不同利益相关方提出的需求信息之间的不一致问题[2]。大多数需求缺陷检测和处理集中在需求获取的后期[2],此时由于获取活动已经基本完成或部分完成,开发人员拿到的是按照一定的文档规范和使用一定的语言所编写的需求规格文档,他们虽然

到稿日期:2017-09-20 返修日期:2017-12-21 本文受民用飞机专项科研项目(MJ-S-2013-10)资助。
苏 若(1990-),男,硕士,主要研究领域为软件需求,E-mail:csuruo@gmail.com;吴 际(1974-),男,博士,副教授,主要研究领域为软件安全性与可靠性等,E-mail:wuji@buaa.edu.cn(通信作者);刘 超(1958-),男,博士,教授,主要研究领域为软件测试等;杨海燕(1974-),女,硕士,讲师,主要研究领域为软件测试等。

能够发现一些需求中的不完整和不一致等问题，但是已难以发现分析人员对需求的理解错误或偏差问题，常常导致需求获取过程的“返工”，甚至项目的延期。

因此，本研究的目标是在 RUCM(Restricted Use Case Modeling)需求建模方法[3]的基础上，提出一套需求信息的早期获取方法 REC(Requirement Elicitation Card)，按照交互协同活动追踪需求关注流 RCF(Requirement Concern Flow)，从不同视角捕捉不同利益相关方对软件产品的各种需求信息，并利用规则检查需求信息中存在的缺陷。在需求获取早期，对获取到的需求信息进行缺陷检测，从而实现需求获取过程中早期缺陷的发现，并引导需求相关方及时修订和充实需求的相关信息。为了检验本文需求缺陷检测方法的效果，将本模型在 3 组研究生教学课程案例上进行了应用，结果表明其有效地检测出了早期获取的需求信息中的常见缺陷。

本文第 2 节介绍相关研究；第 3 节介绍 REC 模型；第 4 节分析需求缺陷类型、特征并设计检测规则；第 5 节通过真实案例说明了该方法的有效性；最后总结全文并对未来的研究方向进行展望。

## 2 相关研究

在多视角需求工程方面，Ross 等基于数据流模型提出 SADT 方法[4-5]；Mulley 提出功能分解的 COR 方法[6]；Leite 提出视角消除的概念[7]；Kruchten 提出基于场景驱动的 Kruchten4+1 并发式分析和开发方法[8]；VOSE 对系统中不同参与者的活动和知识进行分类并使用模板对其进行结构化组织；Kotonya 等提出的 VORD 方法基于面向服务的模型描述视角来规约交互系统[9]；Sommerville 等提出的 PREView 方法利用多视角思想来提高需求获取的质量[10-11]；SEI 标准和 IEEE1471-2000 标准都以利益相关方为核心，SEI 用风格表示视角，IEEE1471-2000 以模板描述视点，使需求文档具有更好的结构性，同时降低视角表示的复杂性；RUCM(Restricted Use Case Modeling)需求建模方法[3]是一种基于用例的方法，由结构化的用例规约模板和一组限制规则构成。

需求不一致问题是普遍存在的[12-14]。如何检测需求中的不完整、不一致等缺陷，已经成为需求工程中的一个重要问题。需求缺陷的检测方法与需求建模原则和需求模型的表示形式密切相关。当前研究的解决方案可分为 4 类[15]：1)基于定理证明的方法。该类方法支持需求缺陷的自动检测，但是定理证明的低效性限制了其可行性[15]。2)基于模型检验的方法[16]。目前最常见的模型检测工具有 SMV[17] 和 SPIN[18]。该类方法在语义、检测过程及定理证明方面均具有良定性，但其只能检测特定类型的需求缺陷，且算法效率会随状态空间的指数型增长而降低。3)根据系统目标进行需求建模，利用目标的语义模式和关于目标的启发式规则检测需求缺陷。KAOS 方法[19-20]的需求缺陷检测策略依赖于其需求建模理念“目标”的语义，因此更适合检测需求中的语义缺陷。4)根据特定图的性质或特定图元的语义检测一个需求描述内部或一组需求描述之间的需求缺陷。文献[21]提出结合场景模型和类模型对软件系统的功能需求进行建模，并将场景定义为参与者之间交互的有序集合。通过最小化描述之间的重叠来减小缺陷的可能，系统建立交叉引用的相关关系，并检测交叉引用的不完整和不一致，通过定义规则或手工检测缺陷。

## 3 多视角卡牌模型

现代大型软件系统常常包含庞大且复杂的需求，必须采取分治的思想将项目分解为多组相对独立的场景，并通过对各个场景进行描述来建立项目需求的完整视图。REC 模型以场景为获取需求相关信息的基本单位，利益相关者协同输入需求的相关描述信息，在已有场景及其描述的基础上，检测存在的不完整和不一致等缺陷。

一般而言，场景描述了若干实体之间为了完成某个共同任务而进行的一组动作和交互[22]。以软件系统边界为参考，可将场景分为 3 种类型[23]：1)内部场景，即软件系统的内部实体以及内部实体之间的交互行为所构成的场景；2)边界场景，即软件系统的外部实体、内部实体以及内外部实体之间的交互行为构成的场景；3)外部场景，即软件系统的外部实体以及外部实体之间的交互行为所构成的场景。REC 将需求获取的后两种类型分别称为使用场景和业务场景，并分别从两个层次来描述软件需求。

业务场景是对所需处理事物的上下文以及执行顺序的抽象，不依赖于具体的软件系统。使用场景是对用户使用系统的交互方式的描述，是业务场景的实现[2]。项目的业务流程包含一组业务场景(Scenario)，但是规模可能仍然较庞大，可以继续对其细分为业务活动(Activity)。一个业务场景包括多个业务活动，业务活动包括开始和结束条件、详细的业务描述以及优先级。业务场景以及业务活动均关联多个用户(User)和数据(Data)。使用场景是从用户与系统的交互行为来刻画软件需求的一种方式，通常使用用例描述。一个用例(Usecase)用来描述一个角色在某种情境下的相关交互活动，包括开始和结束条件、详细的交互活动序列(steps)[2]。场景中涉及的数据(Data)一部分来源于系统中的关键词、领域特征词，甚至用户等概念(ConceptDict)；另一部分来源于系统的处理对象(DataObj)，表现为输入、输出或存储的数值，或对这些数值的类别、特征或含义的描述[2]。

因此，REC 模型从系统(System)、领域(Domain)、用户(User)、数据(Data)、业务场景(Senario)、业务活动(Activity)和用例(Usecase) 7 个视角对系统的整体需求进行划分和描述，以获得尽可能完备的需求相关信息。如图 1 所示，System 从系统概况角度建模，包含项目名称、项目简述、功能概述(coreFuction)、用户类别等信息；Domain 从系统所处领域角度建模，包含领域名称、领域概述、领域特征 (domainFeatures)等信息；User 从用户信息角度建模，包含职责(task)、交互方式(interactiveMode)、权限级别(authority)、使用频率(frequency)以及使用的平台和语言等信息；Data 从数据角度建模，分为概念词(ConceptDict)和系统处理对象(DataObj)，其中概念词包括领域特征词和系统关键词两类，包含数据项名称(name)和数据项描述(description)等信息，其中领域特征词与 Domain 关联；Senario 从活动场景角度建模，描述所有

用户使用系统的活动场景,此时不涉及用户和软件的交互过程,主要包含场景名称、场景描述、场景流程(Flow)和相关依赖等信息;Activity 从活动事件角度建模,描述活动场景中的一些共性事件,主要包括活动名称、活动描述、优先级(priority)、前后置条件(precondition, postCondition)和活动流程(Flow)等信息;Usecase 从用例视角建模,描述用户和系统之间的交互过程,主要包含用例名称、优先级、前后置条件、用例流程(Flow)等信息。

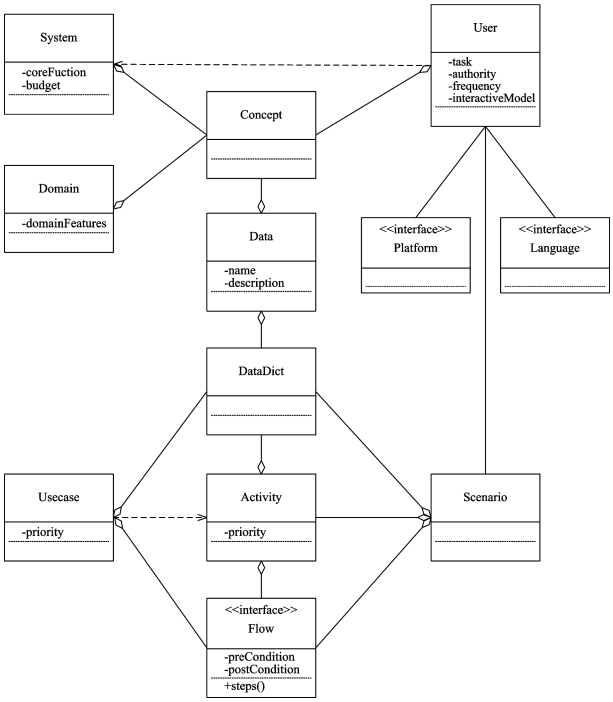


图 1　REC 模型

Fig. 1　REC model

## 4　需求缺陷检测

需求描述来源的不同,导致对同一内容或相关内容的表述不同,进而造成需求描述中普遍存在不完整和不一致等需求缺陷问题;即便为同一来源,也时常出现不一致的情况。多视角需求获取方法在综合整理不同视角捕捉的需求相关信息时,常能发现之前难以发现的不完整和不一致缺陷。REC 模型引导利益相关者从不同角度描述系统的需求相关信息,在收集过程中对这些信息进行(多视角)"分类"整理。基于这些需求信息的类型特点,确定其必需满足的约束要求,建立对应的完整性和一致性检查规则。基于这些规则,对需求信息的完整性和一致性进行检查。

在对工业领域的需求调研[24]中发现,需求中最常见的缺陷有 4 类:1)需求被表达成解决方案;2)需求表达存在二义性;3)需求描述不完整;4)不一致。针对需求缺陷的研究[25]将需求问题分为:冗余、不完备、不正确、不一致和其他问题(例如非原子的需求) 5 类,同时强调了需求中不完整和不一致的缺陷问题。因此,如图 2 所示,本文提出的基于 REC 模型的需求缺陷检测规则主要针对初步的需求信息中常见的不完整、不一致这两种需求缺陷进行检测。

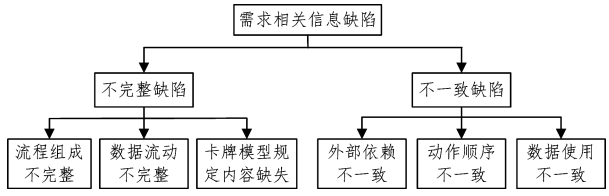


图 2　需求缺陷检测的分类

Fig. 2　Classification of requirement defect detection

### 4.1　需求不完整

一般来讲,完整的需求[2]须满足以下条件:1)需求描述了软件应该实现的所有功能;2)需求为所有有效的输入定义了响应;3)需求描述涉及的图表都有完整的标识和描述;4)需求描述中涉及的计量术语和单位均有完整的定义。需求验证针对给定的需求描述,结合其描述方法,通过分析需求中表达的逻辑来找到可能的缺失内容。例如,在用例形式的需求描述中,每个用例都应该有前置条件和后置条件[26];在涉及资源(即需求涉及的相关物质要素或信息)描述的用例需求中,每个资源都应该出现在用例的场景描述[27]中。本文依据 1),2),4)中的定义对 REC 模型获取的需求信息进行完整性检测。不完整缺陷主要包括以下几类。

1)流程组成不完整

在场景、活动和用例视角中,一条完整的控制流(ControlFlow)要从入口点(EntryNode)开始,经历若干处理过程,包括条件检测(BranchNode)、系统交互(SequentialNode)等,最终以退出点(ExitNode)结束。在 RUCM 的需求描述中,入口点(EntryNode)就是用例规约的前置条件(PreCondition),而退出点(ExitNode)就是后置条件(PostCondition)。

**规则 1**　任一条控制流都有入口结点和出口结点。

Context:Flow

inv:ControlNode. allInstances ( ) − > select ( node1 | node1. owner = self) − > exist ( node2 | node2. oclIstTypeOf ( EntryNode)) and ControlNode. allInstances ( ) − > select ( node3 | node3. Owner = self) − > exist ( node4 | node4. oclIstTypeOf (ExitNode))

**规则 2**　任一条控制流都以入口结点为起点,以出口结点为终点。

Context:Flow

inv: ControlNode. allInstances ( ) − > select ( node1 | node1. owner = self and node1. incoming < > null) − > first(). olcIsTypeOf ( EntryNode) and ControlNode. allInstances ( ) − > select( node2 | node2. owner = self and node2. outgoing < > null) − > first(). olcI-sTypeOf(ExitNode)

RUCM 中的需求描述除了以基本事件流(BasicFlow)结束以外,还可能以各种分支流(AlternativeFlow)结束。分支流(AlternativeFlow)应该是从基本事件流(BasicFlow)的条件检测语句(ConditionCheckSentence)引出的,而条件检测语句(ConditionCheckSentence)对应了 REC 模型中的分支结点(BranchNode)。因此任何一个分支结点(BranchNode)的流出边(outgoing)都必须是两条,相对地,任何一个流出边(outgoing)是两条的控制结点(ControlNode)只可能是分支结点

(BranchNode),这样才能保证每一个条件检测语句(ConditionCheckSentence)都引出了一条分支流(AlternativeFlow)。

**规则 3** 流程中任何一个分支结点的流出边都必须是两条。

Context:BranchNode

inv:self.outgoing->size()=2

**规则 4** 任何一个流出边是两条的控制结点只可能是分支结点。

Context:ControlNode

inv:ControlNode.allInstance()->select(node1|node1.outgoing->sieze()=2)->forAll(node2|node2.oclIstTypeOf(BranchNode))

2)数据流动不完整

数据流是数据项定义及使用的过程。REC 中的数据项由数据传感结点(DataSensingNode)定义(即通过传感器或 I/O 设备采集数据),由其他类型的数据结点(DataNode)使用,包括分别对应数据传输的数据传输结点(DataTransferNode)、数据发送给传动器的数据传动结点(DataEffectionNode)和数据经由存储设备存取下来的数据存储结点(DataStorageNode)。

**规则 5** 数据流中应该至少有一个数据结点中的操作为定义数据。

Context:Flow

inv:self.head.oclIsTypeOf(DataSensingNode)

**规则 6** 数据流中至少有一个数据结点中的操作是使用数据。

Context:Flow

inv:DataNode.allInstance()->select(node1|node1.owner=self and node1.outgoing=null)->forAll(node2|node2.oclIsTypeOf(DataEffectionNode or node2.oclIstTypeOf(DataStorageNode)))

3)卡牌模型的规定内容缺失

对于一个完整的动作(ActionType),REC 中规定必须包含两个要素,即操作动作(operation)和操作对象(object)。因此,动作结点(ActionNode)中包含的所有动作都必须有操作动作和操作对象值,否则为描述不完整。

**规则 7** 动作结点中包含的所有动作的操作动作和操作对象属性值都不可为空。

Context:ActionNode

inv:self.actions->forAll(action|action.operation<>null and action.object<>null)

### 4.2 需求不一致

不一致缺陷在需求获取过程中不可避免。需求一致性是指一个需求规约中不同描述成分之间是否具有逻辑矛盾的特性[28],逻辑矛盾表明某些需求描述成分的逻辑存在不一致现象。需求不一致缺陷主要表现为[29]:对外部对象(如资源或者外部实体)的要求或约束不一致;需求描述动作之间在描述逻辑或执行时序上不一致;对一个对象(如数据项或参与者)的描述和术语使用不一致。针对上述 3 种不一致缺陷,本文提出如下解决方案。

1)外部依赖不一致。REC 中涉及不同外部对象(资源或实体)的交互,为保证系统功能满足预期,需要对外部对象的依赖关系进行约束。所谓外部依赖不一致,是指不同需求或同一需求内部对同一外部对象的依赖不同。

**规则 8** 在数据流中,同一个数据项不能出现多次定义。

Context:Flow

inv:DataNode.allInstance()->select(node|node.owner=self and node.oclIsTypeOf(DataSensingNode))->forAll(n1,n2|n1.data<>n2.data)

2)动作顺序不一致。REC 中动作时序关系库(ActionTemporaryRelationRules)定义了一系列动作之间的时序关系(ActionPair)。时序关系通过 firstItem 和 secondItem 两个属性表示两个动作(ActionType)发生的先后顺序。如果两个动作时序关系中包含两个完全相同的动作对,但是动作间的先后顺序不一致,即为动作顺序不一致。

**规则 9** 如果动作时序关系库中的任意两个动作对包含的动作一致,则它们的时序关系也应该一致。

Context:ActionTemporaryRelationRules

inv:self.temporaryRules->forAll(r1,r2:ActionPair|((r1.firstItem<>r2.secondItem and r1.secondItem<>r2.firstItem))or ((r1.firstItem=r2.firstItem and r1.secondItem<>r2.secondItem)))

3)数据使用不一致。在需求流程中,数据流是数据项定义以及使用的过程。

**规则 10** 数据在使用之前必须要有定义。

Context:Flow

inv:DataNode.allInstance()->select(node1|node1.owner=self and (node1.oclIsTypeOf(DataTransferNode)or node1.oclIsTypeOf(DataEffectionNode) or node1.oclIsTypeOf(DataStorageNode)))->forAll(node2|DataNode.allInstance()->select(node3|node3.owner=self and node3.oclIstTypeOf(DataSensingNode))->first().data=node2.data)

## 5 应用案例

本节使用真实案例对上述方法的有效性进行分析。案例分析的目标是回答如下问题:

1)所设计的多视角卡牌模型是否能够从需求的初步信息中提取出有效的需求相关信息模型?

2)所设计的需求缺陷检测规则能够发现案例中哪些已知的缺陷?

3)所设计的检测规则是否会误报(相对于人工审查)和漏报缺陷?

### 5.1 案例选择

选取研究生教学课程实验的 3 类项目作为案例,参与需求获取过程的人员为系统地学习过软件工程知识的研究生。课程项目案例的规模适中,并且易于获取利益相关方的信息,尤其是从不同软件产品的直接使用者角度提出的初步需求信息,可有效检验不同利益相关方之间的不一致需求等缺陷。本案例主要以录音形式在领域分析、需求获取和需求分析等

阶段采集,团队获取的初步需求信息的过程数据(团队会议中,不同队员在不同阶段分别代表不同利益相关方从不同视角表达项目的需求相关信息);然后通过 IBM Waston 项目的 Speech to Text Demo 机器翻译和人工校对的方式将录音整理成文字,提取出相应的多视角 REC 模型,并使用本文定义的缺陷检测规则进行检测。3 个案例相互补充,覆盖了检测规则所涉及的各类缺陷。这 3 个项目的相关信息如下。

1)简易配置管理系统,用于管理软件开发过程中产生的代码和各种文档。由服务器和客户端组成,支持用户权限,能够执行 check-in 和 check-out,能够检测冲突,且能够定义基线和版本。课程项目按照 SVN 的核心功能来分析和确定系统的功能,并结合 Git 的核心功能进行了补充。该案例的特点为场景活动中的动作描述较多,能很好地验证与动作时序相关的规则的验证效果。

2)轨道交通信号灯控制系统,通过信号灯来控制在任何时刻一个行车区间内的列车数量都不能多于设定的容量,且列车之间的距离必须确保在安全距离之外。该系统通过轨道传感器获得列车的行驶速度,通过区段传感器感知列车的进入和退出。区段之间顺序相连。系统由信号灯、信号灯控制器、列车速度传感器、列车位置传感器、区段监护传感器和控制模型组成。该系统处理的异常情况较多,能很好地体现和控制流相关规则的验证效果。

3)单电梯控制系统,由一个垂直升降梯和多个楼层上的相应设备构成。该系统的用例主要描述了系统对管理员和乘客的请求处理。该案例的用例事件流比较简单,分支流数量较少,用例之间不存在包含或扩展关系,因此,对于用例依赖和分支流验证的效果不明显,但是可以通过对这种简单的用例规约来检测本文方法是否能检测出一定数量的、无效的误报错误。

### 5.2　案例结果及分析

在课程教学过程中,通过老师、助教和组间互评人工审查案例需求,累计发现了 51 个错误,结果如表 1 所列。

表 1　人工审查缺陷个数的统计

Table 1　Statistics of number of defects acquired by manual check

| 项目名称 | 不完整错误个数 | 不一致错误个数 |
|---|---|---|
| 简易配置管理系统 | 13 | 5 |
| 轨道交通信号灯控制系统 | 16 | 3 |
| 单电梯系统 | 4 | 10 |
| 总计 | 33 | 18 |

基于多视角卡牌模型的需求缺陷检测首先将案例中的需求信息描述转换成不同视角的模型表达。3 个案例合计 76 个术语、28 个场景、41 个活动、41 个用例,如表 2 所列。

表 2　模型提取结果的统计

Table 2　Statistics of results extracted by model

| 项目名称 | 术语个数 | 场景个数 | 活动个数 | 用例个数 | 提取模型个数 |
|---|---|---|---|---|---|
| 简易配置管理系统 | 16 | 13 | 25 | 25 | 25 |
| 轨道交通信号灯控制系统 | 30 | 6 | 9 | 9 | 9 |
| 单电梯系统 | 30 | 9 | 7 | 7 | 7 |
| 总计 | 76 | 28 | 41 | 41 | 41 |

采用本文提出的检测规则对这些模型进行缺陷检查,检查出的缺陷情况如表 3 和表 4 所列,其中不完整缺陷数为 34,不一致缺陷数为 22。表中还分别列出了误报和漏报缺陷数。

表 3　基于 REC 模型的不完整需求缺陷的统计

Table 3　Statistics of incomplete requirement defects based on REC model

| 项目名称 | 不完整缺陷 | | | |
|---|---|---|---|---|
| | 已知缺陷个数 | 检查缺陷个数 | 误报缺陷个数 | 漏报缺陷个数 |
| 简易配置管理系统 | 13 | 15 | 3 | 1 |
| 轨道交通信号灯控制系统 | 16 | 15 | 1 | 2 |
| 单电梯系统 | 4 | 3 | 0 | 1 |
| 总计 | 33 | 33 | 4 | 4 |

表 4　基于 REC 模型的不一致需求缺陷的统计

Table 4　Statistics of inconsistent requirement defects based on REC model

| 项目名称 | 不一致缺陷 | | | |
|---|---|---|---|---|
| | 已知缺陷个数 | 检查缺陷个数 | 误报缺陷个数 | 漏报缺陷个数 |
| 简易配置管理系统 | 5 | 8 | 3 | 0 |
| 轨道交通信号灯控制系统 | 3 | 4 | 1 | 0 |
| 单电梯系统 | 10 | 10 | 1 | 1 |
| 总计 | 18 | 22 | 5 | 1 |

经过实验数据的验证,基于多视角卡牌模型的检测方法能够检测出 80%(41/51)的已通过人工审查方式检测出的缺陷,从而证明了该方法的有效性。数据显示不一致缺陷误报个数相对较多,这可能是因为在提取动作顺序关系时提取出来的不一致顺序在不同的领域并没有太强的顺序限制,所以不能够算作缺陷。

### 5.3　讨论

结合案例实施的过程和所得结果,本文提出了基于多视角卡牌模型的缺陷检测方法,其能够有效地在多视角卡牌模型获取的需求相关信息的基础上,根据需求缺陷检测规则,采用 OCL 的方式将规则表示为限定在 REC 元模型的模型元素上的约束表达式,从而支持自动化验证。针对 3 个实际案例,本文方法检测出了 80%(41/51)的已知缺陷,从而证明了其有效性。

鉴于需求获取阶段的重要性、复杂性和反复性等特征,所提方法的主要目的是基于需求缺陷基本类别的划分提出相应规则,进而帮助分析人员及时发现问题,但并不能保证发现全部问题。在以后的研究和工作中将深入支持需求演化过程的研究,进而实现需求信息的有据可循。

**结束语**　发现甚至解决不同利益相关方的需求相关信息之间的缺陷,保证项目整体需求的完整性和一致性,在需求获取活动中发挥着至关重要的作用。目前针对初步需求信息的缺陷检测的研究较少,且常忽略自动化检测在需求缺陷发现中的应用。本文从自动化检测初步需求信息中不完整和不一致缺陷的角度出发,描述了项目的初步需求信息获取和分析的过程,构建了缺陷检测模型,分析了其中的重要实体和关联。依据此模型,课题组将继续系统需求获取和分析验证的协同化和自动化研究。

本研究的应用案例为研究生教学课程实验，具有一定的代表性，但仍需在更多领域、工业项目上进行实验验证，从而增强本模型的适应性和可用性。

## 参 考 文 献


[1] DAVIS A M. Just Enough Requirements Management: Where Software Development Meets Marketing[M]. New York: Dorest House, 2005.

[2] WIEGERS K, BEATTY J. 软件需求[M]. 北京：清华大学出版社，2016.

[3] TAO Y. Automatically Deriving a UML Analysis Model from a Use Case Model[D]. Ottawia: Carleton University, 2010.

[4] ROSS D T. Structured Analysis (SA): A Language for Communicating Ideas[M]// Programming Methodology. Springer New York, 1978: 16-34.

[5] ROSS D T, JR K E S. Structure Analysis for Requirements Definition[J]. IEEE Transactions on Software Engineering, 1977, 3(1): 6-15.

[6] MULLERY G P. CORE-a method for controlled requirement specification[C]// International Conference on Software Engineering. 1979: 126-135.

[7] LEITE J C S P, FREEMAN P A. Requirements Validation Through Viewpoint Resolution[J]. IEEE Transactions on Software Engineering, 1991, 17(12): 1253-1269.

[8] KRUCHTEN P. Architectural Blueprints — The "4 + 1" View Model of Software Architecture [J]. IEEE Software, 1995, 12(6): 42-50.

[9] KOTONYA G, SOMMERVILLE I. Requirements engineering with viewpoints [J]. Software Engineering Journal, 2002, 11(1): 5-18.

[10] SOMMERVILLE I, SAWYER P. Requirements Engineering: A Good Practice Guide[J]. European Journal of Dental Education, 1997, 168(1): 220-221.

[11] SOMMERVILLE I, SAWYER P, VILLER S. Viewpoints for Requirements Elicitation: A Practical Approach[C]// Third International Conference on Requirements Engineering, 1998. IEEE, 1998: 74-81.

[12] BALZER R. "Tolerating Inconsistency" revisited[C]// International Conference on Software Engineering. Toronto: IEEE Computer Press, 2001: 665.

[13] GHEZZI C, NUSEIBEH B. Guest editorial: Introduction to the special section[J]. IEEE Transactions on Software Engineering, 1999, 25(6): 782-783.

[14] EASTERBROOK S, CHECHIK M. Int'l workshop on living with inconsistency[C]// Internationa Conference on Software Engineering. Toronto: IEEE Computer Press, 2001: 749-750.

[15] ZHU X F, JIN Z. About inconsistency management in software requirements [J]. Journal of Software, 2005, 16(7): 1221-1231. (in Chinese)
朱雪峰，金芝. 关于软件需求中的不一致性管理[J]. 软件学报，2005，16(7)：1221-1231.

[16] CLARKE E, GRUMBERG O, LONG D. Verification tools for finite-state concurrent systems[C]// REX School /Symposium on A Decade of Concurrency, Beflections and Perspectives. London: Springer-Verlag, 1994: 124-175.

[17] MCMILLAN L. Symbolic model checking [D]. Pittsburgh: Carnegie Mellon University, 1992.

[18] HOLZMANN J. The model checker SPIN[J]. IEEE Transactions on Software Engineering, 1997, 23(5): 279-295.

[19] LAMSWEERDE V, DARIMONT R, LETIER E. Managing conflict in goal-driven requirements engineering[J]. IEEE Transactions on Software Engineering, 1998, 24(11): 908-926.

[20] LAMSWEERDE V, LETIER E. Handling obstacles in goal-oriented requirements engineering [J]. IEEE Transactionson Software Engineering, 2000, 26(10): 978-1005.

[21] GLINZ M. A lightweight approach to consistency of scenarios and class models[C]// Proceedings of the 4th Int'l Conference on Requirements Engineering. Schaumburg: IEEE Computer Press, 2000: 49-58.

[22] CARROLL J M. Scenario-based design: envisioning work and technology in system development [M]. John Wiley & Sons, Inc., 1995: 375-376.

[23] JARKE M, BUI X T, CARROLL J M. Scenario Management: An Interdisciplinary Approach[J]. Requirements Engineering, 1998, 3(3/4): 155-173.

[24] FANMUY G, FRAGA A, LLORENS J. Requirements Verification in the Industry[J]. Complex Systems Design & Management, 2012, 13(169): 145-160.

[25] YAN Y Q, LI S X, MEI X Y. Defect needs analysis and management model[J]. Computer Science, 2009, 36(4): 140-144. (in Chinese)
严玉清，李师贤，梅晓勇. 缺陷需求分析与管理模型[J]. 计算机科学，2009，36(4)：140-144.

[26] KOESTERS G. Coupling Use cases and Class Models as a Means for validation and verification of requirements Specification[J]. Requirements Engineering, 2001, 6(1): 3-17.

[27] LEITE J C S D P, HADAD G D S, DOORN J H, et al. Scenario Construction Process [J]. Requirements Engineering, 2000, 5(1): 38-61.

[28] Software Considerations in Airborne Systems and Equipment Certification: DO-178B/C[S]. 1982.

[29] IEEE. Recommended practice for software requirements specifications[S]. 1998.