面向不同软件制品的需求追踪方法研究综述

陶传奇[1),2),3),4)] 张 萌[1)] 郭虹静[1)] 黄志球[1),2),4)]

[1)](南京航空航天大学计算机科学与技术学院 南京 211106)

[2)](南京航空航天大学高安全系统的软件开发与验证技术工业和信息化部重点实验室 南京 211106)

[3)](南京大学计算机软件新技术国家重点实验室 南京 210023)

[4)](软件新技术与产业协同创新中心 南京 210093)

摘 要 近年来,随着软件规模和复杂度的不断提升,软件系统在开发过程中产生了大量的需求文档、设计图、代码类、测试文档等中间产物,即软件制品,这些软件制品中蕴含着海量的数据信息.需求工程影响软件开发的整个生命周期,当软件需求不断变更时,软件制品呈现碎片分散化的形态,缺乏全局、统一的组织整理,软件制品间缺乏关联.因此,建立需求追踪关系可以显著提高软件开发与维护效率,这已经成为软件工程领域的研究热点.当前综述工作主要集中在需求追踪关系构建方法的描述,缺乏对软件生命周期过程中需求与不同软件制品间追踪关系建立的分析,导致了需求追踪关系类型的单一性.针对该问题,本文采用系统性文献综述的方法,以需求追踪研究为核心,选取近 10 年来的 135 篇研究文献,从软件生命周期与制品类型角度,分析需求与不同软件制品间追踪关系的构建方法、应用现状与发展趋势,并将需求追踪技术在真实软件开发项目中进行应用,提高智能化软件开发效率.首先,依据软件全生命周期的各个阶段,本文对软件制品按文本内容分为文档级软件制品、代码级软件制品以及产品级软件制品,重点分析需求与文档级软件制品、需求与代码、需求与非特定软件制品间追踪关系建立技术的研究进展及研究效果;其次,总结相关文献中的实验数据集、工具研发情况以及需求追踪技术的应用场景;接着,通过智能化软件开发项目中真实的案例分析,讨论了需求追踪技术在提高软件开发效率方面的有效性.最后,本文对未来值得关注的研究方向进行展望,包括追踪链接类型、追踪方法以及自动化工具,便于研究人员进一步研究.

关键词 需求追踪技术;软件制品;信息检索;机器学习;智能化软件开发

中图法分类号 TP311 **DOI 号** 10.11897/SP.J.1016.2022.02393

Review of Research on Requirements Traceability Approaches for Different Software Artifacts

TAO Chuan-Qi[1),2),3),4)] ZHANG Meng[1)] GUO Hong-Jing[1)] HUANG Zhi-Qiu[1),2),4)]

[1)](*College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing* 211106)

[2)](*Key Laboratory of Ministry of Industry and Information Technology for Safety-Critical Software, Nanjing University of Aeronautics and Astronautics, Nanjing* 211106)

[3)](*State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing* 210023)

[4)](*Collaborative Innovation Center of Novel Software Technology and Industrialization, Nanjing* 210093)

Abstract In recent years, as the scale and complexity of software have increased continuously, a large number of intermediate products, namely software artifacts, have been produced in the software development process. For example, requirements documents, design drawings, code classes, test documents, etc. These software artifacts contain a large amount of data information. In particular, it is well known that requirements are of great significance for software development,

收稿日期:2021-06-01;在线发布日期:2022-07-22.本课题得到国家自然科学基金、JW 装发预研项目(公开)(31511110204)、国家重点研发计划项目(2018YFB1003900)资助.**陶传奇**,博士,副教授,中国计算机学会(CCF)高级会员,主要研究方向为智能化软件开发与测试.E-mail:taochuanqi@nuaa.edu.cn.**张 萌**,硕士研究生,中国计算机学会(CCF)会员,主要研究方向为需求工程.**郭虹静**,博士研究生,中国计算机学会(CCF)会员,主要研究方向为智能化软件工程.**黄志球**,博士,教授,中国计算机学会(CCF)理事,主要研究领域为软件工程、形式化方法和云计算.

and requirements engineering affects the entire software development life cycle. When the requirements of the software are constantly changing, the software artifacts are scattered and present a fragmented form. In addition, there is a lack of effective methods to organize these messages in a global and unified manner, resulting in a lack of correlation between software artifacts. Therefore, establishing the traceability between requirements and other software artifacts can significantly improve the efficiency of software development and maintenance, which has become a hot research topic in the field of software engineering. The previous literature reviews mainly focus on requirements tracing methods and introduce the technologies commonly used when establishing requirements traceability. Thus, there is a lack of analysis on the technology of establishing the traceability between requirements and different software artifacts from the perspective of the software life cycle process, resulting in a single type of requirements traceability established by researchers. Towards addressing the aforementioned limitation, the paper uses a methodology of Systematic Literature Review (SLR) that has been validated by many scholars. Meanwhile, it takes the requirements traceability research as the core and selects a total of 135 studies in the past 10 years. Taking into the software life cycle and the types of software artifacts account, the paper analyzes the approaches, applications, and development trends of the traceability between requirements and different software artifacts. What's more important is to apply requirements traceability technology in real software development projects to improve the efficiency of intelligent software development. Firstly, according to the various stages of the software life cycle, the paper classifies software artifacts based on text content. These software artifacts are divided into three categories: document-level software artifacts, code-level software artifacts, and product-level software artifacts. This paper mainly investigates and analyzes the research advance of the establishment of the traceability between requirements and document-level software artifacts, requirements and code, requirements and non-specific software artifacts. Besides, the research effect of the existing requirements traceability approaches is analyzed. Secondly, summarize open-source data sets commonly used in experimental design, tools developed by researchers, and application scenarios of tracing techniques in the relevant literature in detail. Last but not least, through the real case analysis in the intelligent software development projects, the paper discusses the effectiveness of the requirements traceability technologies in improving the efficiency of software development. In general, there are still many problems in the field of requirements traceability that need to be further resolved. The paper specifically looks forward to future research directions that are worthy of attention, including types of traceability, tracing methods, and automated tools. Hence, domestic and foreign researchers can conduct further research on requirements traceability.

Keywords requirements traceability approach; software artifacts; information retrieval; machine learning; intelligent software development

1 引 言

近年来,随着现代软件产业的快速发展,软件系统愈加复杂,子系统及功能模块相互关联且彼此依赖,在大规模软件开发过程中积累了海量的软件制品,包括需求文档、设计模型、代码类文件、测试文档、缺陷报告、讨论记录以及邮件列表等.这些软件制品中隐含了海量的数据信息.例如,一架 F-22 喷气式歼击机包含的代码约为 200 万行,而一台高端汽车包含的代码更是高达 1 亿行[1],同时一辆 2004 年的汽车包含的系统规格说明书已经达到 20000 页[2].然而,这些软件制品结构各异,缺乏有效的组织和整理,使得软件制品间追踪关系的建立和分析不完善、软件开发效率低下.需求工程贯穿软件开发的整个生命周期,确定涉众和涉众的需求,并将其文档化,

可以用于支持系统设计、代码开发、测试工程以及软件的后期维护[3]. 因此，面向不同软件制品的需求可追踪性问题已经成为软件工程领域的研究难点和关键点.

需求可追踪性，即在软件开发过程中使用需求追踪技术建立需求与各种软件制品间的关联关系，并将关联关系以可视化的方式展示给开发人员，生成相应的追踪候选链接，包括有效链接和无效链接. 其中，有效链接表示需求与软件制品间存在关联，否则反之. 1994 年 Gotel 和 Finkelstein[4] 给出了需求可追踪性的明确定义，即“在系统的整个生命周期中，从前后两个方向描述并追踪需求的能力”，其目的是确保所有工作成果符合用户需求. 目前，需求追踪技术已被广泛应用于众多软件工程任务，如变更影响分析、安全认证、系统验证以及依赖影响分析等，可以帮助提高软件开发[5] 和维护[6-8] 效率.

需求可追踪性的重要性在工业界已广为人知，且需求追踪技术已被众多机构和安全关键领域采用. 在 1968 年举行的北约工作会议上，“需求可追踪性”这一概念被首次提出，并在文件中明确指出，开发人员必须显式的标记需求与代码元素间的追踪关系[9]. 1980 年，需求可追踪性已被众多军用软件开发标准考虑，例如美国国防部军标 DOD-STD-2167A[10]. 1992 年，Watkins 和 Neal[11] 根据自身的实践经验提出，需求可追踪性可以支持各类软件（不限于军用软件）的可靠性验证、版本控制以及变更管理，并认为需求可追踪性是规范软件开发过程的必要条件. 2015 年，Mäder 和 Egyed[12] 通过实验表明，在需求可追踪性的支持下，第三方开发项目中的软件维护任务执行速度平均提高了 24%，正确率平均提高了 50%.

然而，需求追踪技术在工业界的实际使用率并不高. 其主要原因在于，建立需求与其他软件制品间的追踪关系时，早期的需求追踪技术需要耗费大量的人力劳动才能实现高准确率，同时随着软件系统的快速发展，该方式总是耗时且容易出错的，比如基于模型的方法[13]、基于本体的方法[14] 以及基于规则的方法[15]. 为了解决这一问题，研究界一直在积极探索解决方案，目前主要以基于信息检索的方法[16] 以及基于机器学习的方法[17-18] 为主. 这些方法在保证一定准确率的前提下，试图实现自动化或半自动化的需求追踪技术，减少人工参与. 除此之外，研究学者根据现有方法开发了有效的需求追踪工具，使得需求追踪关系以可视化的方式表示[19].

为了增加研究学者对当前需求追踪领域研究现状的了解，现有的研究工作旨在分析需求追踪关系建立时常用的关键技术，没有重点研究需求与不同软件制品间追踪关系的建立，最终导致建立的需求追踪关系类型相对单一，影响了软件开发的整体性. 同时，现有综述中暂无对需求追踪在智能化软件开发项目中应用的研究. 因此，本文以需求与不同软件制品间追踪关系的建立为主，对近 10 年以来需求追踪领域的研究现状进行分析，并且将相关技术应用于真实的智能化软件开发项目中，以验证需求追踪技术的有效性.

本文第 2 节系统地介绍需求追踪方法的整体研究框架；第 3 节通过文献调查来分析需求追踪领域的研究现状；第 4 节给出智能化软件开发项目中真实的案例研究；第 5 节介绍综述相关的国内外研究工作；第 6 节对全文进行总结，并讨论未来的研究方向.

2　研究框架

面向不同软件制品的需求追踪方法整体研究框架如图 1 所示，主要分为数据资源、支撑技术以及实际应用三个方面. 首先，本文以支撑技术为核心，研究面向不同软件制品的需求追踪方法，分别对需求与文档级软件制品间追踪关系建立、需求与代码间追踪关系建立以及需求与非特定软件制品间追踪关系建立的相关技术进行描述，并且对常用的需求追踪技术及其研究效果进行分析. 接着，对实验中常用的数据资源进行收集和讨论，包括工业项目、学生开发项目以及开源软件项目的分析，并且给出用于需

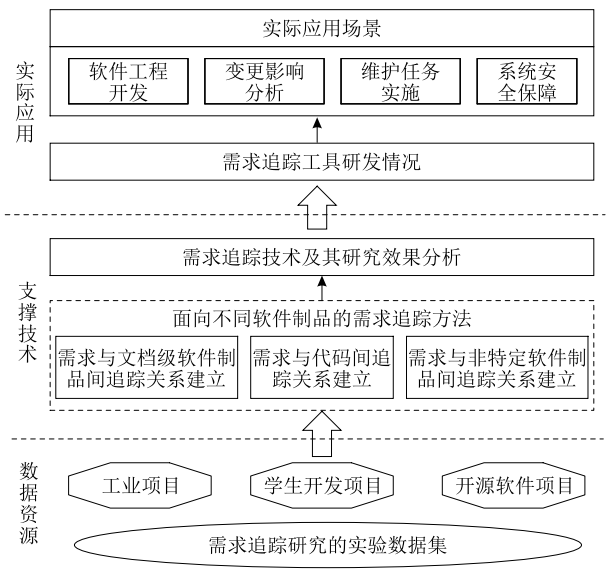


图 1　需求追踪方法整体研究框架

求追踪研究的常用实验数据集.最后,探究目前需求追踪工具研发情况以及需求追踪技术在软件工程中的实际应用场景,表明需求追踪技术的有效性.

2.1 面向软件制品的需求追踪方法分类

软件制品作为追踪对象,已经成为需求追踪方法中至关重要的部分.因此,根据不同的软件制品,本文对需求追踪方法进行了分类研究.

2.1.1 软件制品划分

对于面向不同软件制品的需求追踪方法的研究,本文首先需要明确追踪的对象,即软件制品的类型.根据软件全生命周期的各个阶段以及文本内容,本文对不同阶段产生的常用软件制品进行了划分,如图 2 所示.从图中可以看出,软件制品被划分为三类:文档级软件制品、代码级软件制品以及产品级软件制品.

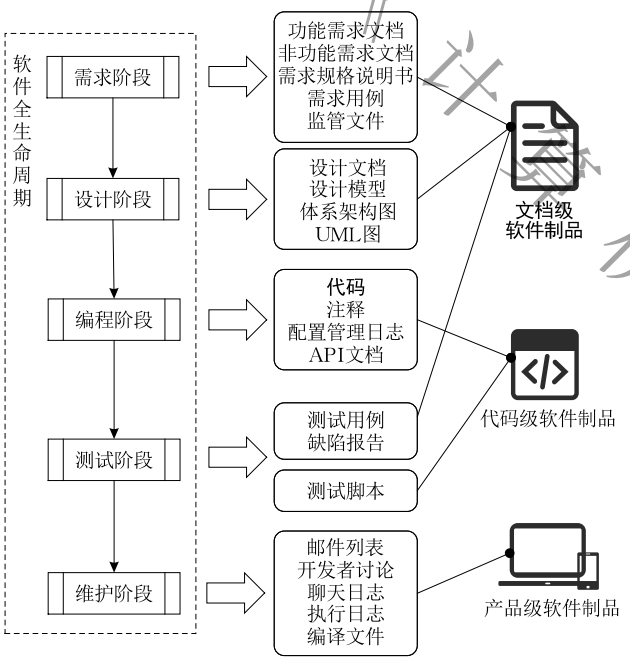


图 2 软件制品的划分

文档级软件制品的文本内容主要由自然语言编写形成,且有相似的非结构化或半结构化的文本结构,通常需要分词、词性标注、句法分析、命名实体识别、同义词识别以及去除停用词等自然语言处理技术进行预处理,因此该类软件制品具有共性.基于这一特点,文档级软件制品主要包括需求、设计以及测试的不同软件开发阶段产生的制品.对于需求阶段,主要包括功能需求文档、非功能需求文档、需求规格说明书、需求用例以及监管文件.对于设计阶段,主要包括设计文档、设计模型、体系架构图以及 UML 图.而测试阶段中只有测试用例以及缺陷报告属于文档级软件制品.

代码级软件制品中包含源代码,通常使用编程语言形成文本内容,因此可以使用抽象语法树对源程序进行解析并提取源文件中的关键信息.该类制品通常在编程阶段以及测试阶段产生,包括源代码、注释、配置管理日志、API 文档以及测试脚本.其中,注释以及配置管理日志等制品可以用于辅助需求与代码间追踪关系的建立.

产品级软件制品则主要指软件系统维护阶段得到的中间产物,其文本内容可以由自然语言和程序语言编写形成.

同时,加粗的文字表示本文主要的研究目标为需求与文档级软件制品以及需求与代码间追踪关系的建立,其他软件制品当前关注度较少,这里暂不讨论.

2.1.2 需求追踪方法分类

支撑技术的研究主要包括面向不同软件制品的需求追踪方法介绍和常用需求追踪技术及其研究效果的分析.根据 2.1.1 节中对软件全生命周期中常用软件制品的划分,本文将面向不同软件制品的需求追踪方法分为三类:

(1)需求与文档级软件制品间追踪关系建立,该类方法主要用于建立需求与需求阶段、设计阶段以及测试阶段产生的不同制品间的追踪关系,由于文档级软件制品主要使用自然语言进行描述,因此该类方法侧重于研究自然语言文本间关联信息的挖掘.

(2)需求与代码间追踪关系建立,该类方法旨在得到需求与源代码文本间的追踪链接,由于源代码文本主要使用编程语言进行描述,因此该类方法还涉及源程序中关键信息的提取以及不同语言间的相似性比较.同时,注释、配置管理日志等制品的使用可以帮助开发人员建立更多潜在的追踪关系.

(3)需求与非特定软件制品间追踪关系建立,该类方法是一种不针对需求与某一种特定软件制品间可追踪性的建立技术,而是适用于各种制品的追踪关系建立过程,具有一定的通用性.

通过基于不同软件制品类型的追踪方法研究,可以对当前需求追踪领域有一个整体的把握,并且帮助利益相关者选择合适的方法进行追踪关系的建立.

2.2 需求追踪方法

2.2.1 需求追踪过程

需求与软件制品间追踪关系的建立通常是双向追踪的过程,即包括向前的可追踪性以及向后的可追踪性.对于确定的追踪对象,向前的可追踪性是指将其他制品链接到对应的需求,而向后的可追踪性是指将需求链接到其他不同的软件制品.通过双向

追踪关系的建立，可以获取需求与其他软件制品间的相互对应关系，使得系统功能完整，并且确保了系统实现符合用户需求.

图 3 所示为需求追踪关系的建立过程，旨在选择合适的追踪策略来建立需求文本与其他制品间的双向追踪关系，并将判断正确的追踪链接进行存储，便于开发人员的后续使用，协助软件工程中各项任务的完成. 早期的实践活动中，追踪关系通常被存储在需求跟踪矩阵(Requirements Traceability Matrix，RTM)中，使得需求与制品间的追踪关系被简单而直观地表达出来[20]. 同时，部分学者也选择使用文本方式对需求可追踪性进行永久存储，自定义了追踪关系的展示和存储类型，包括 csv 文件、xml 文件以及 Json 文件等，允许用户自行下载使用. 同时，针对初始生成的追踪链接，可以使用不同的方法进行优化并存储追踪关系[21]，以尽可能去除结果中的假阳性链接，并挖掘更多潜在的有效链接，提高需求追踪结果的准确性，增加工业实践者的信任度. 该类方法主要以人工干预方法、基于外部结构信息的方法以及混合方法为主，目的在于得到质量更高的追踪结果.

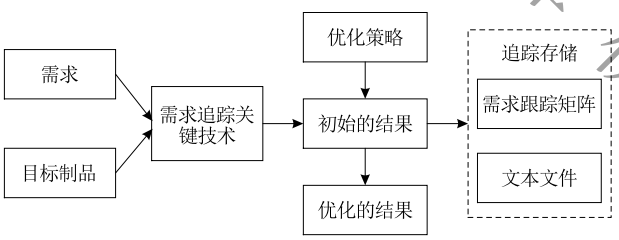


图 3　需求追踪关系建立过程

2.2.2　需求追踪关键技术

有效的需求追踪关键技术有利于需求与不同软件制品间追踪关系的建立. 目前，众多研究学者已经对需求的可追踪性进行了大量的研究，并且提出了有效且可行的需求追踪关键技术. 表 1 所示为常用的需求追踪关键技术及其对应特征的分析，具体分为以下 6 种方法：基于规则的方法、基于本体的方法、基于模型的方法、基于信息检索的方法、基于机器学习的方法以及混合方法.

表 1　需求追踪关键技术及特征分析

关键技术	技术特征
基于规则的方法	建立软件制品间的推导/转换规则，通过规则建立追踪关系
基于本体的方法	提供软件制品间对领域知识的共享和共同的概念(以领域概念模型为主)，使用概念的关联建立追踪关系
基于模型的方法	主要以模型驱动工程构建的追踪模型(不包括领域概念模型)为主建立追踪关系
基于信息检索的方法	对于提取的关键字，基于信息检索模型或语义模型进行相似度计算来建立追踪关系
基于机器学习的方法	训练机器学习分类器来识别有效或无效链接
混合方法	综合多种需求追踪领域相关技术建立追踪关系

其中，基于规则的方法、基于本体的方法以及基于模型的方法均以早期研究学者提出的手动的需求追踪技术为主. 基于规则的方法根据推导或转换规则来建立追踪关系. 基于本体的方法主要通过构建领域概念模型建立追踪关系. 而基于模型的方法指构建除了领域概念模型以外的其他追踪模型来建立追踪关系，有利于需求与 UML 图等设计制品间追踪关系的建立，已经被广泛使用. 对于小规模系统，该方法能取得好的需求追踪结果. 然而，随着软件系统的复杂化，这些方法耗时、冗余且容易出错[22]. 因此，需求追踪技术逐渐趋向于自动化或半自动化.

基于信息检索的方法以及基于机器学习的方法是当前最受欢迎的两种方法，也是需求追踪技术趋向于自动化的典型代表. 其中，基于信息检索的方法旨在使用信息检索模型或相似度模型来建立追踪关系，并在此基础上提出改进策略. 基于机器学习的方法旨在利用分类算法训练机器学习分类器，并自动识别有效或无效链接. 这两种方法均提高了需求追踪过程的自动化程度，但是追踪结果的准确度仍需要进一步提高.

除此之外，混合方法则包括综合由多种追踪方法生成的候选追踪链接的优化方法以及在需求追踪关系建立的不同阶段采用不同关键技术的方法，主要以基于信息检索的方法或者基于机器学习的方法为主. 该类方法综合考虑了多种方法的技术特征，使得混合方法的效果达到最佳，弥补了单一方法的不足.

3　研究现状

为了对需求可追踪性研究现状进行讨论与分析，本文选取了在 2011 年 1 月～2020 年 12 月期间发表的文献. 通过对检索到的文献进行筛选、评估、数据抽取与整合的过程，本文依据提出的研究内容详细阐述了需求追踪领域的发展情况.

3.1　研究现状概述

首先，本文以中文关键字“需求追踪”、“需求跟踪”以及英文关键字“requirements traceability”分别在 IEEE Xplore、ACM Digital library、Elsevier ScienceDirect、EI Compendex、SpringerLink、ISI Web of Science 和 CNKI 的 7 个数据库中对论文的标题、摘要和关键词进行检索，得到论文的初选清单. 接

着,对所选文献的摘要与正文进行解读和评估,筛选出与需求追踪方法或工具相关的论文,同时利用滚雪球[23]的方式对参考文献完成遴选和补遗.根据文献检索过程,本次综述得到的研究文献共 135 篇.

(1)本文对该领域近年来的刊物发表情况进行了统计和分析,如图 4 所示.可以看出,2011 年～2013 年以及 2017 年～2020 年两个时间段内发表的论文数量相对较多,而 2014 年～2016 年间发表的论文数量相对较少,其原因在于,自 2017 年起,信息

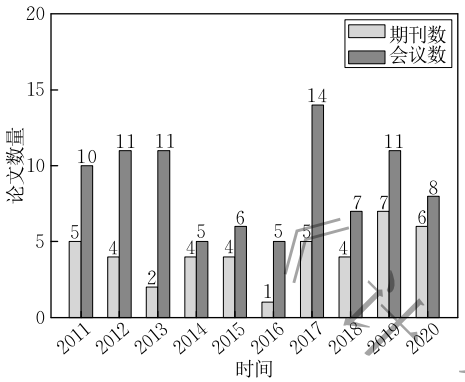


图 4 2011 年～2020 年论文数量统计

检索以及机器学习技术迅速发展,众多研究学者尝试将这些技术应用于需求追踪领域,极大地促进了需求追踪技术的发展,从侧面表明需求追踪领域的研究一直受到众多研究学者的高度关注.

(2)通过对论文发表情况的统计可知,在所选的 135 篇研究文献中,10 年间共有 42 篇论文(占比 31.11%)发表在不同的期刊上,其余 88 篇论文(占比 65.19%)发表在不同的会议上,表明每年有一定比例的会议和期刊接受需求追踪领域的相关论文.参照中国计算机协会认定的软件工程领域国际期刊及会议目录,对论文发表源进行统计,按照论文数量从大到小排序,如表 2 所示,其中列举了部分期刊和会议上的论文发表情况.从表中可以看出,众多的国际期刊和会议均可以收录需求追踪领域的论文,其中排名前 5 的发表源是 RE、ICSE、IST、JSS 和 ASE.其他的期刊和会议也均有一定的论文占比,从侧面反映出国际及国内学术界对需求追踪领域非常重视,具有一定的研究意义.除此之外,我们发现大多数文献发表在软件工程领域的多个主流刊物上,反映了需求追踪是软件工程领域的研究热点.

表 2 论文发表源统计

发表源全称	简称	类型	论文数
Information and Software Technology	IST	期刊	4
Journal of Systems and Software	JSS	期刊	4
Empirical Software Engineering	ESE	期刊	4
Information Systems	IS	期刊	3
Requirements Engineering	RE	期刊	2
IEEE International Requirements Engineering Conference	RE	会议	8
International Conference on Software Engineering	ICSE	会议	6
International Conference on Automated Software Engineering	ASE	会议	4
IEEE International Conference on Program Comprehension	ICPC	会议	3
International Conference on Advanced Information Systems Engineering	CAiSE	会议	2
International Conference on Software Analysis, Evolution, and Reengineering	SANER	会议	2

(3)除此之外,本文还统计了该方面主要的研究学者名单,按论文数量从大到小排序,如表 3 所示,记录了主要研究人员所在单位以及在所有研究文献中发表的论文数量,其中在该研究问题上最为活跃的是来自加拿大皇后大学的 Ali Nasir 教授以及来自美国芝加哥德保罗大学 Cleland-Huang Jane 教授、密西西比州立大学的 Niu Nan 教授所领导的研究小组,国内则以武汉大学计算机学院的彭蓉教授、南京大学计算机软件新技术国家重点实验室的匡宏宇老师为代表,近 10 年来,这些研究学者所在的研究小组持续发表了较多的论文并分享其研究成果,成为新技术研究的典型代表.

表 3 研究人员统计

研究人员	所在单位	论文数量
Ali Nasir	Queen's University, Canada	8
Cleland-Huang Jane	DePaul University, USA	6
Niu Nan	Mississippi State University, USA	6
Mahmoud Anas	Louisiana State University, USA	4
Mäder Patrick	Johannes Kepler University, Austria	3
Tsuchiya Ryosuke	Waseda University, Tokyo, Japan	3
Haidrar Saida	Mohammed V University in Rabat, Morocco	3
匡宏宇	南京大学	3
彭蓉	武汉大学	2
De Lucia Andrea	University of Salerno, Italy	2

(4)根据论文的研究方向,本文也对研究文献进行了划分和统计.表 4 所示为研究文献的分类

结果以及对应的研究文献列表，分为需求追踪技术和需求追踪工具两个方面，其中需求追踪技术按照需求追踪链接类型的不同进行分类，包括需求与需求、需求与设计、需求与测试、需求与代码、需求与非特定软件制品，共包含文献 130 篇，可以看出需求与设计以及需求与代码间追踪关系的建立是当前的研究热点；需求追踪工具是指研究人员依据新提出的方法进行开发的工具或被用于实验辅助的工具，共包含文献 20 篇. 下面将依据研究问题对面向不同软件制品的需求追踪方法研究现状进行讨论.

表 4　研究文献分类结果

技术/工具		论文数量	研究文献
需求追踪技术	需求与需求	21	[8][14][17][24-41]
	需求与设计	39	[8][13][42-78]
	需求与测试	7	[31][79-84]
	需求与代码	45	[16][64][85-127]
	需求与非特定软件制品	18	[128-145]
需求追踪工具		20	[17-18][29][34-35][60][79-80][105][108][118][135][141][146-152]

3.2　需求与文档级软件制品间追踪关系的建立

依据对文档级软件制品的划分，本小节致力于分析需求与需求、需求与设计、需求与测试间追踪关系建立的技术进展情况，包含的研究文献具体如表 4 所示，通过对文献的整理，得到需求与文档级软件制品间建立追踪关系时常用的方法以及具体的实施手段.

3.2.1　需求与需求间追踪关系建立

需求与需求间追踪关系的建立包括与需求相关的所有软件制品间追踪关系的建立，具体分为功能需求文档、非功能需求文档、需求规格说明书、需求用例以及监管文件，该类制品内容较短且具有非结构化的特点. 表 5 所示为需求与需求间追踪关系建立时常用关键技术与研究文献的对应关系，下面对每类关键技术的研究进展进行介绍.

表 5　需求与需求间追踪关系建立技术与研究文献的对应关系

关键技术	研究文献
基于模型的方法	[8][26][36][41]
基于规则的方法	[29][35][37]
基于本体的方法	[14][30][32]
基于信息检索的方法	[24-25][38]
基于机器学习的方法	[17][27][39]
混合方法	[28][31][33]

基于模型的方法. 该方法通过构建追踪模型实现需求追踪关系的建立. 文献[8]提出一种基于谓词逻辑的语义模型，通过基于谓词逻辑的形式系统描述制品间的追踪信息，给出需求追踪关系的形式化语义，自动推导需求与需求间的追踪关系. Dominik 等人[26]提出一种针对概念图的元模型，该模型中包含需要建立连接的需求相关的软件制品，并将它们以图形符号的形式表示，通过规则分析与推导需求及其受影响的对象或利益相关者，实现追踪链接的可视化. Zheng 等人[41]构建面向软件安全性需求分析过程的追踪模型，规定应当追踪的实体以及关联，并通过追踪约束关系建立安全性相关信息的追踪.

基于规则的方法. 该方法通过形式化逻辑推理的方法挖掘需求制品中隐藏的关联信息，从而建立需求与需求间的追踪关系. Hallerstede 等人[29]通过形式化推理完成需求与需求规格说明书的关联. Filax 等人[35]将形式化推理方法应用于安全系统中，建立安全需求与需求规格说明书间的追踪关系. Lockerbie 等人[37]使用推理引擎限制需求的自然语言描述，建立软件工程领域的需求追踪关系.

基于本体的方法. 该方法旨在描述需求追踪领域知识的通用概念模型，是一种是形式化的、共享概念体系明确说明的方法[153]，通过概念间的相互关联可以建立需求特征间的追踪关系. Gazzawe 等人[14]以本体形式定义需求追踪模型，使用公共属性或概念描述需求相关制品间的关联关系. Murtazina 等人[30]选择 OWL 本体建模语言对需求中的知识进行建模，以建立共享概念下需求元素间的追踪关系. Saputri 等人[32]利用基于本体的知识表示和信息检索的技术来自动或半自动更新需求文档中的追踪链接.

基于信息检索的方法. 该方法主要通过信息检索模型或直接语义相似性计算的方式建立需求与需求间的追踪关系. 信息检索模型以向量空间模型以及潜在语义索引为主. Eder 等人[25]通过提出确定潜在语义索引方法配置的算法，恢复了需求制品间的追踪链接. 直接语义相似性计算则重点关注需求语义信息的提取. Mahmood 等人[24]提出一种基于语义的方法来恢复需求制品间的追踪链接，通过 DBpedia 知识库、Babelnet2.5 多语言字典以及语义网络计算需求间的相似性，并使用三重提取算法和三重消歧算法完成追踪关系的建立.

基于机器学习的方法. 该方法目前利用机器学习的分类算法完成需求追踪链接的分类或需求跟踪矩阵的生成. 由于无监督学习在训练数据集中不需要加入确定的样本标签，大大减少了人力成本，因此

聚类算法受到很多研究学者的关注.文献[27]中利用了词嵌入、概率相关度计算以及层次聚类的方法对需求特征进行分类.然而,无监督学习在评估性能的有效性方面非常困难,因此,部分研究学者已经对监督学习算法展开了研究.Wieloch 等人[17]通过实验证明决策树算法和特征子集选择的组合有利于追踪项目中重复出现的类似形式的软件制品.

混合方法.该方法主要结合多种方法来建立需求追踪关系.Jain 等人[28]将基于规则和信息检索的方法结合,通过规则提取软件制品中的词典、语法和语义特征,根据不同的语义相似性计算的方式建立需求用例与监管文件间的追踪关系.文献[31]将向量空间模型与迭代主题模型结合,通过引入一种擅长处理短文本的模型 BTM,提高了基于向量空间模型生成追踪链接的准确性.文献[33]将基于本体和机器学习的方法结合,提出一种知识丰富的方法,包括基于手工构建本体派生的额外特征以及基于注释派生的额外训练实例、扩展支持向量机模型的训练数据集、通过有监督的学习对需求进行追踪.

3.2.2 需求与设计间追踪关系建立

与设计相关的软件制品包括设计文档、设计模型、体系结构图以及 UML 图,因此设计制品通常以图形描述为主.本问题旨在建立需求与所有设计相关的制品间的追踪关系,目前已经有众多研究学者对需求与设计间的追踪关系建立技术展开研究,表 6 列举了该问题常用关键技术与研究文献的对应关系,从表中可以看出基于模型的方法是当前需求与设计间追踪关系建立的主要技术手段.

表 6 需求与设计间追踪关系建立技术与研究文献的对应关系

关键技术	研究文献
基于模型的方法	[8][13][42-46][49-50][52][55-59][61][63][65][69-70][74-75]
基于本体的方法	[47][66-67][71][73]
基于信息检索的方法	[53][68]
基于机器学习的方法	[48]
混合方法	[51][54][62]

基于模型的方法.使用该方法进行需求与设计间追踪关系建立时以构建模型为主,包括需求建模以及追踪模型两种方式.

在将需求进行建模的过程中,通过将需求以图形化的方式呈现,可以自动建立需求与设计模型间的追踪关系,以下出现的 SysML 和 AADL 是两种标准建模语言,分别描述嵌入式系统中的需求和设计.Taromirad 等人[52]基于模型驱动工程,提出一种特定领域的敏捷建模方法来管理需求追踪,采用特定领域的建模语言来描述追踪模型,并表示目标领域关于追踪目标的结构、行为和特征.文献[61]中则利用 UML 建模工具(半)自动更新需求和设计模型间的追踪关系.Deng 等人[75]通过对 SysML 模型的扩展,对需求与设计制品进行建模用于捕获追踪信息.同时,在通过需求建模自动建立的需求与设计模型间追踪关系的基础上,研究学者也对已建立的追踪链接进行优化.文献[43]提出了一种基于 SysML 的方法增强 SysML 建模图中的需求可追踪性,包括丰富需求定义和可追踪性的 SysML 配置文件以及生成追踪模型的算法,通过算法将需求与设计元素关联起来,生成不同粒度级别的追踪链接.Haidrar 等人[46]利用模型驱动工程的优势,从 SysML 模型中自动生成需求与设计模型间的追踪关系,同时面向 SysML 语言设计了一个自动化追踪模型提取算法的通用需求追踪框架[49].

除此之外,可以直接构建追踪模型并制定约束规则建立需求与设计间的追踪关系.文献[8]提出一种基于谓词逻辑的语义模型,通过定义约束规则实现需求与设计间纵向追踪关系的推导与检验.Jadoon 等人[13]提出一种基于模型的需求追踪框架,通过元模型的转化建立需求规格说明书与 UML 图间的追踪关系.文献[59]采用需求关系和体系结构验证技术,通过追踪元模型定义常用的追踪类型,自动生成需求和体系结构间的追踪关系.Wen 等人[70]使用行为树构建追踪模型,通过确定需要执行的行为构建需求与设计文档间的追踪关系.

基于本体的方法.该方法通过需求与设计制品共有的本体属性以及不同的本体类型,建立双方之间的追踪关系.文献[47]提出一种将通用本体和特征领域本体信息结合建立需求与设计文档间的候选追踪链接.文献[67]使用通用本体建立需求规范和体系结构设计间的可追踪性.Lapeña 等人[71]使用自然语言处理和本体论的方法扩展隐性需求,恢复需求与模型间的追踪链接.文献[73]中则提出一种基于修饰词本体的关键词语义判断方法,建立需求文档与设计文档间的可追踪性.

基于信息检索的方法.该方法依据设计相关制品的特征对现有信息检索模型或语义模型进行改进来建立需求与设计间的追踪关系.Udagawa[53]提出两种增强向量空间模型的方法,包括使用术语的文档标识符进行检索以及对查询中相关的术语进行处理,从而建立需求与设计文档间的追踪关系.Kchaou 等人[68]通过构建语义模型,应用相似性计算方法建

立需求和 UML 元素间的追踪关系.

基于机器学习的方法. 该方法从特征提取的角度对需求与设计间的追踪关系进行分类. Guo 等人[48]使用深度学习(词嵌入和递归神经网络)将需求相关的软件制品和领域知识结合生成需求和设计描述间的追踪链接.

混合方法. 该方法综合多种方法建立需求追踪链接. Mirakhorli[51]将基于机器学习和规则的方法结合,通过训练分类器来识别设计决策片段,并使用追踪模式进行重构,建立设计决策与需求间的逆向追踪关系. 文献[54]将基于本体和机器学习的方法结合,利用需求与设计的词汇特征和基于本体派生的特征,通过聚类算法对需求与设计间的追踪链接进行分类. Marcén 等人[62]将基于机器学习和演化计算的方法结合,通过使用一种基于学习排序算法的进化算法,恢复需求与模型片段间的追踪链接.

3.2.3　需求与测试间追踪关系建立

相比于需求,测试相关的软件制品具有结构化或半结构化的特点,包括测试用例和缺陷报告. 目前关于需求与测试间追踪关系建立的研究相对较少,表 7 展示了关键技术与研究文献的对应关系.

表 7　需求与测试间追踪关系建立技术与研究文献的对应关系

关键技术	研究文献
基于模型的方法	[80-81][83]
混合方法	[31][84]

基于模型的方法旨在建立需求与测试间的可追踪性模型或框架,依据测试用例结构化的组成部分,将追踪关系的建立过程以抽象化的形式表达. Aichernig 等人[80]提出需求驱动的测试生成框架,通过需求接口驱动,将非正式需求形式化同步到测试用例包含的不同的数据流视图中,以建立需求与测试用例间的追踪关系. Azmi 等人[81]定义追踪链接元模型,将追踪链接以标识符的形式呈现,建立需求与测试文档间的追踪关系. Espinoza 等人[83]使用一种可追踪性元模型 TmM,该模型支持角色、链接规则以及用户可定义的追踪链接 3 个特征,为敏捷开发过程中的涉众提供广泛支持.

测试用例的文本具有简短的特点,因此使用混合方法将向量空间模型与迭代主题模型结合,尽量从有限的文本内容中提取信息. 文献[31]将短文本处理模型 BTM 和向量空间模型结合,通过从较少的文本内容中提取术语并进行主题建模,提高需求与测试用例间追踪链接的准确性. 同时在文献[84]提出一种生成需求追踪链接的混合方法 VSM+BTM-GA,将基于信息检索和演化计算的方法结合,其中 BTM-GA 是一种管理短文本制品以及短文本处理模型初始参数的遗传算法模型,降低用户对经验的依赖性高,并提高了方法的可用性,捕获基于向量空间模型建立追踪关系过程中忽略的隐藏关联信息.

3.3　需求与代码间追踪关系的建立

由于编程语言与自然语言有着完全不同的语法结构,因此建立需求与代码间的追踪关系是非常困难的,本综述重点研究了该方面的技术进展,通过对该问题的回答,了解需求与代码间追踪关系建立的主要技术. 在本小节中,其他代码级软件制品(如 API 文档、注释以及配置日志)的出现均用于辅助需求与代码间追踪关系的建立. 表 4 描述了需求与代码间追踪关系建立相关的研究文献,共计 45 篇,由于目前对该研究问题的理论成果较多,因此本文根据常用的关键技术对研究文献进行分类,如表 8 所示,可以看出当前需求与代码间追踪关系建立最常用的方法是基于信息检索的方法.

表 8　需求与代码间追踪关系建立技术与研究文献的对应关系

关键技术	研究文献
基于信息检索的方法	[16][86-87][90-91][94][97][100-102][105-109][112][114][116-119][121-124]
基于机器学习的方法	[85]
基于规则的方法	[110]
基于模型的方法	[120][125]
混合方法	[88-89][92-93][95][98-99][103][111]

3.3.1　基于信息检索方法的改进策略

对于需求与代码间追踪关系的建立,常用的方法是基于信息检索的方法,其基本思想是将需求和代码映射到向量空间中,通过相似度计算得到候选追踪链列表. 在基于信息检索(Information Retrieval, IR)的方法中,需求和代码通常分别被视为查询和文档,基于现有的信息检索模型,包括向量空间模型(Vector Space Model, VSM)、潜在语义索引(Latent Semantic Index, LSI)、概率模型、主题模型、Jensen-Shannon(JS)模型以及语言模型等,引入不同的改进策略来提高候选追踪链列表的准确性. 常用的信息检索模型为向量空间模型和潜在语义索引,通过软件制品的预处理、候选链接的过滤以及外部信息的引入等 3 个方面对基于信息检索的方法进行优化,下面进行详细介绍.

软件制品的预处理. 该方法旨在需求和代码文本的预处理阶段,加强对文本信息的分析和提取,从

而提高需求与代码间追踪链接的准确性.

需求文本是由自然语言编写的,因此在处理需求文本时可以通过上下文分析来加强需求语义分析. Zhou 等人[16]利用以需求上下文和需求意图为特征的上下文信息,通过加权知识模型在需求用例与相关的代码类文件间建立追踪链接. Charrada 等人[94]通过识别代码中可能影响需求的更改,并提取描述变化的关键字来查找代码变更时的过时需求.

代码文本是由程序语言编写的,而代码重构可以使代码更加规范化,有助于提高代码质量,发现更多隐含的需求与代码间的追踪关系. 文献[116]中通过评估不同重构方法对于基于信息检索的追踪方法的影响,确定使用重命名标识符的重构方法有助于追踪关系的建立.

候选链接的过滤. 该方法旨在对基于信息检索方法建立的需求与代码间的候选追踪链接结果进行筛选或重排序,从而得到优化的追踪链接列表,提高结果的可靠性. 直接过滤候选链接有助于提高现有追踪结果的准确性,剪枝策略是常用的手段,它是指通过判断来砍掉搜索树上不必要的子树. Ali 等人[112]使用动词的组合约束策略对候选链接进行剪枝,以显著提高现有需求追踪结果的准确性. 对候选链接列表的重排序通常需要与其他方法结合,这将在 3.3.2 节的混合方法中进行介绍.

外部信息的引入. 该方法旨在引入新的外部信息来改进基于信息检索方法建立的需求追踪关系.

首先,代码类之间存在继承和调用关系,而隐含这种关系的多个代码类通常对应相同的需求,根据该特点,研究学者对信息检索方法进行了改进. Ghabi 等人[109]通过研究代码内部具有调用关系的上下文,验证候选追踪链接的正确性. Shao 等人[122]利用类间继承关系所产生的层次信息形成类簇,以簇为单位确定需求与代码类间的追踪关系. Du 等人[124]利用代码类间的关系获取每个代码元素的代码模式,即代码类之间依赖和被依赖的关系,以便对候选列表进行重排序.

其次,代码级软件制品中的其他信息,如注释、配置管理日志等,可以用于辅助需求与代码间追踪关系的建立. 文献[108]中通过实验强调考虑注释的重要性,具有注释的代码不仅更容易理解,而且注释作为外部文档可以对代码标识符进行补充说明,更容易建立需求与代码间的追踪关系. 由于配置管理日志包含有关软件构件的修改信息,Tsuchiya 等人[118]根据其中包含消息和文件路径的版本控制系统的日志,将基于相似性的方法与基于日志的方法结合来恢复需求与源代码文件间的追踪链接.

最后,借助于用户反馈等人工干预的方式对已有候选链接进行优化. Kuang 等人[107]允许在用户接受或拒绝某些链接后,对计算相似性的方案进行调整,并重新排序未验证的候选链接. Tsuchiya 等人[118]认为用户必须验证已恢复的候选链接,因为结果中可能包含不正确的或忽略的链接,只有经过用户验证才能提高结果的准确性,从而不给用户造成额外的负担.

3.3.2 混合方法

混合方法不仅可以通过结合多种方法建立追踪链接,也可以将多种方法生成的候选链接结合生成新的候选链接. 一方面,将基于信息检索的方法和其他方法得到的不同候选链接结果结合并进行重排序,发现可能丢失的追踪链接. 文献[98]将基于信息检索和演化计算的方法结合,采用排序遗传算法,在基于 Jaccard 相似度和余弦相似度的两个目标函数间寻找最优解,以便得到更加准确的追踪链接. Gethers 等人[99]将信息检索的多种模型生成的候选链接结果结合,对信息检索模型中的向量空间模型、概率模型以及主题模型等 3 种模型得到的候选链接进行正交来恢复需求与代码间的追踪链接. 另一方面,将基于机器学习和逻辑推理的方法结合,试图发现更多可能的追踪链接. Wang 等人[93]结合机器学习和逻辑推理探索用例和代码特征,并使用监督学习算法训练分类器,同时通过推理规则增量发现追踪链接.

3.3.3 其他方法

除此之外,还有一些其他方法用于需求与代码间追踪关系的建立,包括基于机器学习的方法、基于规则的方法以及基于模型的方法,均产生了一定的研究成果. 其中,基于机器学习的方法在选择初始数据集时尽量挖掘需求与代码特征,从而训练出有效的机器学习分类器,并识别出需求与代码间的有效或无效链接,Mills 等人[85]在文本检索排名和查询质量指标等特征的基础上,利用机器学习方法自动将需求用例与源代码文件间的追踪链接分类为有效或无效链接. 其余两种方法均采用人力劳动的方式解决问题,使得需求与代码间追踪链接的准确性较高. 基于规则的方法通过追踪模式将需求与代码联系起来,Florez 等人[110]通过模式将功能约束及其实现进行精确匹配来建立需求与代码间的追踪链接. 基于模型的方法通过构建原型领域模型提取需求与

代码中的实体，利用实体间的关联建立需求与代码间的追踪关系，Vinárek 等人[120]从需求规格说明书中构建原型领域模型，通过实体提取与关联自动恢复需求与源代码间的追踪关系.

3.4　需求与非特定软件制品间追踪关系的建立

在所有的需求追踪技术中，一部分可以建立需求与文档级软件制品间的追踪关系，另一部分可以建立需求与代码间的追踪关系，而其余部分可以建立需求与非特定软件制品间的追踪关系，包括需求与需求、需求与设计、需求与测试以及需求与代码间追踪关系的建立，共包含研究文献 18 篇，如表 4 中的“需求-非特定软件制品”类别所示. 本小节旨在分析通用关键技术进展情况，表 9 列举了需求与非特定软件制品间追踪关系建立通用技术和研究文献的对应关系，从表中可以看出，基于机器学习的方法是当前解决该问题的主要手段.

表 9　需求与非特定软件制品间追踪关系建立技术和研究文献的对应关系

关键技术	研究文献
基于机器学习的方法	[129-131][133][139][142]
基于信息检索的方法	[134]
基于本体的方法	[136]
基于规则的方法	[137-138][144]
混合方法	[128][132][135][141]

对于需求与非特定软件制品间追踪关系的建立，基于机器学习的方法主要从训练机器学习分类器以及软件制品的处理两个角度进行考虑. 训练有效的机器学习分类器的关键点在于机器学习算法的选择. 国内外研究学者采用了不用的机器学习算法训练得到机器学习分类器，以便识别出更多可能的追踪链接，提高方法的可靠性. Niu 等人[131]提出一种基于聚类的假设，通过识别和过滤低质量的追踪链接来提高结果的准确性. Omoronyia 等人[139]提出分别使用贝叶斯和线性推理的技术来建立需求追踪链接，并将两种方法得到的需求追踪结果进行对比，结果表明，线性推理方法具有一定的优势. 软件制品的有效特征提取对于需求追踪关系的建立是至关重要的. Wang 等人[130]提出一种基于人工神经网络的方法来提升自动识别制品中多义项的能力，从而提高自动追踪的精度. Li 等人[133]提出一种无监督的学习结构，利用降维技术和神经网络模型训练需求语言特征. 同时，由于非功能需求散落在需求文档的各个角落，文献[129]基于聚类和信息论的假设，提出一种无监督的方法来提取和追踪软件系统中的非功能需求，以便建立非功能需求与软件制品间的追踪关系.

混合方法可以进一步补充单一方法的缺陷. Bella 等人[135]将基于信息检索和机器学习的方法结合，使用 4 种不同的信息检索技术进行相似性度量，通过对候选链接结果分析识别最有可能正确的链接作为初始训练数据集，并以该数据集和 4 种信息检索技术得到的结果作为训练模型的输入，采用半监督学习的方式训练机器学习分类器，从而对真假链接进行分类. Rochimah 等人[141]将基于信息检索和规则的方法结合，利用语义相似度匹配、链接优先级排序以及基于启发式的方法来生成软件制品间的追踪链接.

除此之外，基于信息检索的方法、基于本体的方法以及基于规则的方法也被用于需求与非特定软件制品间追踪关系的建立. 基于信息检索的方法在改进时则面向所有的软件制品，比如 De Lucia 等人[134]建议使用平滑滤波器来减少噪声对软件制品的影响. 基于本体的方法通过抽取不同软件制品的共有概念或属性来建立关联，Liu 等人[136]利用软件制品的领域语料库生成特定领域的概念模型，基于概念模型改进追踪结果. 基于规则的方法则应用于整个追踪关系建立过程的某一阶段，将制品与其他信息间的联系提取出来，并根据需要进行转换，Dietrich 等人[137]提出一种追踪查询转换技术，依据规则将用作查询的软件制品进行转换，以建立新查询与目标制品间的追踪关系.

3.5　需求追踪技术及其研究效果分析

为了对现有的需求追踪技术及其研究效果进行分析，本章节将从度量指标、需求追踪关键技术分析和研究效果分析三个方面进行讨论.

3.5.1　度量指标

度量指标是衡量研究方法性能的重要依据，研究人员通常根据它来判断算法的优劣. 需求追踪领域中最常用的度量指标是准确率(Accuracy)、查准率(Precision)、查全率(Recall)以及 *F*-Measure，用于验证分类效果. 准确率是指检索结果中判断正确的样本数与所有样本数的比率，衡量的是检索系统中所有样本的准确率；查准率是指检索结果中判断正确的相关样本数与检索结果中所有相关样本数的比率，衡量的是检索系统中相关样本的精确率；查全率是指检索结果中判断正确的相关样本数与语料库中所有相关样本数的比率，衡量的是检索系统中相关样本的召回率. 为了综合考虑算法的性能，*F*-Measure 指标将 Precision 和 Recall 结合起来. 该指标认为查准率和

查全率同等重要，使用加权调和平均的方式解决了高查准率和高查全率之间的冲突，有助于进一步评估检索结果质量.

AP(Average Precision)与 MAP(Mean Average Precision)是另外两个常用的实验度量指标. 其中，AP 用于度量每次查询检索到的相关样本的排序质量. AP 的计算方法如下：

$$AP=\frac{\sum_{r=1}^{N}(Precision(r)\times isRelevant(r))}{|RevelantDocuments|},$$

其中，r 表示被查询对象在候选列表中的排序，$Precision(r)$表示前 r 个结果的精确率. $isRevelant(r)$是一个二值函数，如果样本相关，则返回 1，若无关，则返回 0. MAP 是所有 AP 的平均值：

$$MAP=\frac{\sum_{q=1}^{Q}AP_q}{Q},$$

其中，q 是单词查询，Q 是查询的总数. MAP 值越大表示检索到的相关样本的排序质量越好.

3.5.2 需求追踪关键技术分析

根据需求追踪技术的自动化程度，现有的关键技术主要包括基于信息检索的方法、基于机器学习的方法、混合方法以及其他方法等. 表 10 列举了需求追踪领域中各项关键技术相关的论文数量及应用的软件制品范围，符号“＋”表示当前技术已应用于该软件制品.

表 10 需求追踪技术文献统计及应用制品分析

关键技术	论文数量	应用制品				
		需求	设计	测试	代码	非特定软件制品
基于规则的方法	7	＋			＋	＋
基于本体的方法	9	＋	＋			＋
基于模型的方法	31	＋	＋	＋	＋	
基于信息检索的方法	31	＋	＋		＋	＋
基于机器学习的方法	12	＋	＋		＋	＋
混合方法	20	＋	＋	＋	＋	＋

从表 10 中可以看出，对于需求与不同软件制品间追踪关系的建立，有许多共享的关键技术，比如，基于规则的方法可以用于需求与需求、需求与代码间追踪关系的建立，基于本体的方法可以用于需求与需求、需求与设计间追踪关系的建立，基于模型的方法可以用于需求与需求、需求与设计、需求与测试、需求与代码间追踪关系的建立. 对于非特定软件制品，即所有软件制品共享的关键技术中，基于规则的方法以及基于本体的方法通常用于需求追踪关系建立的某一阶段，比如文本预处理阶段. 基于信息检索的方法、基于机器学习的方法以及混合方法可以用于需求与不同软件制品间追踪关系的建立. 根据文献调查，当前对于需求与测试间追踪关系建立的研究相对较少，使得有部分方法仍未被使用. 对于需求与代码间追踪关系建立技术的研究相对成熟，因此手动的需求追踪技术已经被较少关注. 除此之外，由于机器学习分类器通过训练数据集来识别有效或无效链接，因此，基于机器学习的方法主要用于需求与非特定软件制品间追踪关系建立技术的研究.

3.5.3 研究效果分析

近年来，研究学者使用了不同的需求追踪技术对需求与不同软件制品间追踪关系的建立展开了研究，以自动化或半自动化技术为主，包括基于信息检索的方法、基于机器学习的方法以及混合方法，表 11 展示了不同需求追踪技术对应的具体的研究方法. 基于信息检索的方法包括常用的信息检索模型以及基于信息检索模型的改进策略，基于机器学习的方法包括基于分类算法的监督学习以及基于聚类算法的无监督学习，混合方法是指将信息检索方法或机器学习方法与其他方法的结合. 不同的研究方法均产生了不同的研究效果.

表 11 需求追踪技术对应的研究方法

关键技术	研究方法
基于信息检索的方法	常用的信息检索模型(VSM、LSI、JS、主题模型)
	改进策略(语义分析、剪枝、用户反馈、外部信息的引入)
基于机器学习的方法	聚类算法
	分类算法(k 近邻算法、线性回归、朴素贝叶斯、随机森林、支持向量机)
混合方法	混合信息检索模型
	信息检索＋规则
	信息检索＋演化计算
	信息检索＋深度学习
	机器学习＋本体
	机器学习＋规则
	机器学习＋演化计算
	信息检索＋机器学习

为了更好地对比各项研究的效果，并表明需求追踪技术的有效性，本文使用 Precision 和 Recall 两个基本指标对不同的研究方法进行了性能评估. 对于基于信息检索的方法，本文以 iTrust 开源数据集为基础，总结了常用的信息检索模型以及改进策略的评估结果，如图 5 所示. 结果表明，基于信息检索方法的改进策略使得基于信息检索模型的查准率维持在 40％左右，查全率在最低限度和最高限度分别提高了 3％和 8％左右. 对于基于机器学习的方法，

由于聚类算法主要用于需求与文档级软件制品间追踪关系的建立，具有局限性，因此，本文主要调查了分类算法的研究效果，并分别以 iTrust 和 eTour 开源数据集为基础. 在常用的机器学习分类算法中，随机森林算法总是能取得更优的结果[88]，在查准率为 57%时，该方法在数据集上的查全率分别为 64%和 65%，具有较好的需求追踪结果. 对于混合方法，本文以 eTour 开源数据集为基础. 由于论文中实验数据集的不同，本文调查了部分混合方法的研究效果. 已有研究中，混合信息检索模型的查准率和查全率分别为 40%和 52%；基于信息检索和演化计算方法的查准率和查全率分别为 75%和 75%；基于机器学习和规则方法的查准率和查全率分别为 82%和 64%；基于信息检索和机器学习方法的查准率和查全率分别为 69%和 35%.

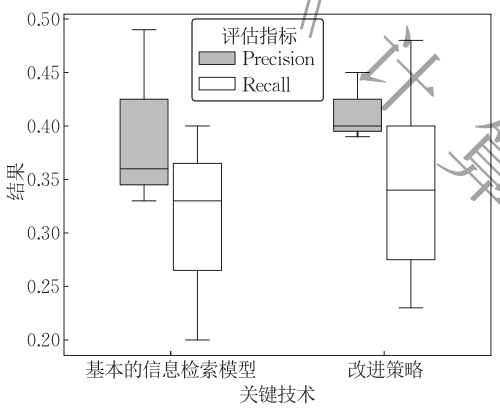


图 5　基于信息检索方法的研究效果对比

同时，通过对不同需求追踪技术的横向对比，说明目前本领域中先进的研究成果. 基于 iTrust 数据集的调查结果显示，相比基于信息检索的方法的最佳值，基于机器学习的方法在查准率和查全率方面均有明显提高，分别提高了 7%和 14%左右. 基于 eTour 数据集的调查结果显示，相比基于机器学习的方法，基于信息检索和演化计算的方法、基于机器学习和规则的方法对于需求追踪结果有了进一步提高，分别提高了 18%和 10%、25%和－1%，说明了这两种混合方法的有效性.

本文也对需求追踪技术的不同研究方法进行了可重现性的分析. 对于信息检索的方法，Mills 等人[142]从 6 个系统中提取了 11 个实验数据集，对基于信息检索模型的方法进行评估，结果表明，最佳准确率和最差准确率相差 33%左右，可重现性较差. Ali 等人[112]在 4 个实验数据集上对改进策略进行了评估，最佳准确率和最差准确率相差 41%左右，方法的可重现性更差. 对于基于机器学习的方法，Du 等人[88]从 5 个中型软件系统中提取了 7 个实验数据集，对使用分类算法的技术进行评估，结果表明，最佳准确率和最差准确率相差 43%左右，该方法具有较差的可重现性. 对于混合方法，Rodriguez 等人[98]在 3 个实验数据集上对基于信息检索和演化计算的方法进行评估，结果表明，最佳准确率和最差准确率相差 16%左右，方法的可重现性较好. Du 等人[88]基于 7 个实验数据集，对基于信息检索和机器学习的方法进行了评估，结果表明，最佳查准率和最差查准率相差 37%左右，方法的可重现性较差.

除此之外，本文对影响需求追踪技术的有效性进行了分析，主要包括 5 个方面. 首先，实验数据集的影响. 不同的需求追踪技术需要在多个数据集上进行实验结果的验证，除了开源数据集，应该考虑更多工业项目的数据集，因为工业项目比开源项目更复杂. 对于基于机器学习的方法，需要在每个数据集上对每种方法进行多次实验，对结果取平均值后使用统计检验来增加实验结果分析的严谨性，并且在大型软件系统上进行实验结果的验证. 其次，软件制品本身的影响. 当需求和其他软件制品具有较好的文本质量时，需求追踪结果准确率较高. 接着，方法中参数值的影响. 参数值通常由多次实验得到，不同的参数值将会产生不同的需求追踪结果. 同时，对于混合方法，有时需要将前一种方法产生的结果作为后一种方法的输入，因此，中间结果的准确率将会对后一种方法的结果产生影响. 最后，在案例研究中进行的问卷调查或访谈，参与者应为有经验的领域专家.

3.6　需求追踪研究的实验数据集

为了方便研究学者对提出的方法进行实验验证，本小节总结了目前需求追踪领域的实验数据集来源. 通过对所选文献的统计，目前的实验数据集主要来源于 3 个方面：工业项目、学生开发项目以及开源软件项目，表 12 统计了常用实验数据集来源以及可利用情况. 其中，开源软件项目数据集通常是公开的且可重复利用的，以 CoEST① 社区网站中的项目数据为主. CoEST 社区是一个由研究人员和实践者组成的社区，自 2002 年起，他们主要的工作是实现可伸缩的、有效的软件制品间的追踪关系，并以 xml 文件或 csv 文件的格式呈现出来，因此，研究学者不需要额外的成本和努力即可获得软件项目中不同软

① CoEST：Center of Excellence for Software Traceability [OL]，[2016-8-25]. http://www.CoEST.org

件制品间完整、准确且值得信赖的需求追踪关系.目前在该网站共有 15 个可用的软件项目,每个项目均包含了不同的软件制品数据,研究人员可以依据自身方法选择合适的数据集进行实验设计.其中,部分数据集相关信息如表 13 所示,包括项目中具体包含的软件制品及数量、有效链接数等.

表 12 实验数据集来源和可利用情况统计

数据集来源	可利用数据集的数量/个	不可利用数据集的数量/个	数据集总数/个
工业项目	13	16	29
学生开发项目	11	5	16
开源软件项目	30	0	30

表 13 实验数据集相关信息

数据集	描述	软件制品及数量	有效链接数
eAnci	意大利市政管理系统	需求用例(140)、代码类(55)	567
eTour	导游系统	需求用例(58)、代码类(116)	308
SMOS	高校学生监控系统	需求用例(67)、代码类(100)	1044
iTrust	健康系统	需求用例(131)、代码类(367)	534
Easy Clinic	医院管理系统	需求用例(30)、交互图(20)、测试用例(63)、代码类(47)	1257
Ice Breaker	破冰设备系统	高级需求(201)、UML 图(73)	457
CM1-NASA	科学仪器	高级需求(235)、低级需求(220)	4050

除此之外,表 12 的实验统计结果表明,部分研究学者还使用从工业界获取的特定领域的项目数据以及由大学生开发的项目数据来进行实验验证.对于工业数据,根据需求追踪技术的主要应用领域,比如医疗保健[86]、汽车和航空航天等安全关键系统[41-42]以及软件工程开发库[79,95]等,研究人员可以从工业合作伙伴处获取该领域内的项目数据集[163],并对提出的需求追踪方法进行有效性验证.但是由于签署了保密协议,有超过一半的数据集不可以被重复利用.对于由学生开发的项目数据[16],通常会被直接上传到开放网站或者通过发送电子邮件从作者处获取,因此,有大约 69%可重复利用的数据集.然而,通过领域专家标注获得有效和无效链接的方式极大地消耗了人力资源[95],同时维护具有高质量的软件制品是困难的,因此,当前仍然缺乏大量的实验数据集.

3.7 需求追踪工具的研发情况

现有工具可以帮助研究学者生成候选追踪链接、用于实验方法对比或者促进工业界开发人员对于需求追踪关系的应用.本小节统计了近 10 年间研发的工具以及被用于实验辅助的工具,便于研究学者对于需求追踪领域工作的开展.表 14 详细统计了相应关键技术中使用的工具信息,根据建立需求追踪关系的关键技术,本文将工具进行分类,并列举每种工具对应的相关文献,分析工具的开发平台及其适用对象,其中“—”符号表示暂时未知,若无明确说明,则需求、设计、测试、代码分别指每个开发阶段相关的所有软件制品.可以看出,基于信息检索方法涉及的工具相对较多,其中,FacTrace 工具支持投票机制,TraceMe 工具包括用户反馈和覆盖分析.同时,最常用的工具有 DOORS、RETRO 以及 TraceLab.

表 14 工具研发情况

关键技术	使用的工具	研究文献	开发平台	适用软件制品
基于模型的方法	ProR	[29]	Eclipse 插件	需求
	TraceMan	[146]	Sparx Systems 插件	需求、设计、测试、代码
	TimeTracer	[149]	—	需求、设计、代码
	R2A	[150]	—	需求、设计
基于本体的方法	TRAILS	[60]	—	需求、设计
基于信息检索的方法	RETRO	[19,34,88]	—	需求、设计、测试等文档
	DOORS	[35,150]	—	需求、设计、测试
	FacTrace	[105]	—	需求、代码、变更日志
	MRTA	[141]	Java 开发平台	需求、设计、测试、代码
	TraceMe	[147]	Eclipse 插件	需求、设计、测试、代码
基于机器学习的方法	TraceLab	[17,88,135]	.NET	需求、设计、测试、代码
	EA-Tracer	[148]	—	需求、代码

商业工具 DOORS 是业界公认的全球使用范围最广的需求管理工具，提供了所有被需要的捕获、跟踪与管理用户需求的功能特点，用于确保项目符合所描述的需求和标准. 利用类似 Word 的界面，可以在 DOORS 中直接输入需求，或者将多种格式文件中的需求导入 DOORS. DOORS 捕获需求后，则可以在整个项目生命周期中使用多种功能，如视图、链接与可跟踪性分析来跟踪与管理它们.

工具 RETRO 支持自动化建立非结构化的软件制品间的追踪关系，包括需求文档、设计文档以及测试文档间追踪链接的生成. 该工具包含多种基于信息检索的方法，生成的候选链接结果具有代表性，因此，学术界经常使用该工具产生的结果作为实验中的基线结果. 同时该工具支持手动建立追踪链接，使得开发人员能够验证与确认自动生成的追踪链接，对追踪链接进行维护.

工具 TraceLab 以基于信息检索的方法为主，提供了一个功能完整的实验环境，研究学者可以使用已有的组件或自定义组件来生成需求与软件制品间的候选链接，并进行实验对比. 目前该工具已被广泛用于基于机器学习方法的分析与研究，通过将自定义组件集成到 TraceLab，可以方便地实现软件制品的预处理、追踪链接的自动生成以及候选结果的质量评估，帮助需求追踪技术改进策略的提出.

研究表明，现阶段开发的需求追踪工具仅考虑需求与不同软件制品间追踪关系的建立，较少考虑需求追踪工具与其他不同软件开发阶段工具的整合与桥接问题，这是未来需要关注的研究方向.

3.8　实际应用场景

随着信息检索和机器学习技术的发展，针对实际场景中特定的软件工程任务，研究学者对需求追踪技术的应用进行了研究. 通过对 135 篇研究文献的分析发现，当前需求追踪技术的应用场景非常广泛，图 6 所示为需求追踪技术的应用情况统计，其中

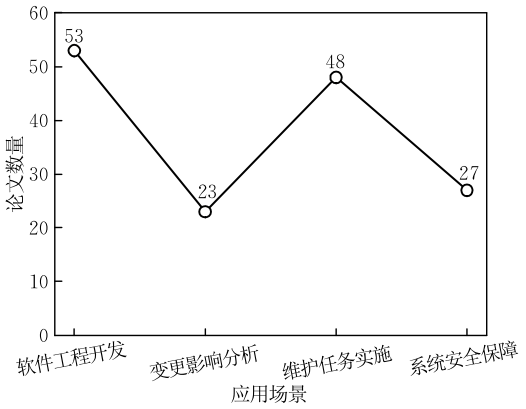


图 6　需求追踪技术的应用情况统计

每篇文献的研究成果可能被应用至多种场景，结果表明，需求追踪技术主要应用于软件开发和软件维护任务，具体如下：

软件工程开发. 软件工程任务开发阶段. Berry[5] 提出“若在软件开发生命周期中忽视了需求追踪，或使用了不完整、不一致的需求追踪关系，将导致系统质量的下降和反复修改，从而提高系统开发和时间成本”. 当前软件需求追踪可以辅助软件开发生命周期中很多活动的执行，比如确认系统功能完整性以及术语一致性、检查产品设计是否符合其预期的目的(Verification and Validation，V&V)、帮助开发人员复用系统组件等[145]. 然而，为了保证软件开发任务的有效进行，需要提前建立需求与软件开发阶段不同软件制品间的追踪关系，包括需求、设计、编程、测试等阶段. 比如，当需要检查产品设计是否符合预期时，应建立需求规格说明书与设计文档间的追踪关系，根据追踪链接检查每部分产品设计与需求点的对应关系，直到所有需求点均被覆盖，节省了人工核查的时间成本.

变更影响分析. 软件功能维护的评审阶段. 文献[7]指出，变更影响分析(Change Impact Analysis，CIA)是“识别变更的潜在后果，包括副作用和连锁反应，或估计在做出变更前需要修改的内容”，利用这些信息可以为问题重新进行软件系统设计. 而使用软件制品间的追踪关系来识别软件更改的影响是常用的方法，根据提前建立好的追踪链接可以获取源制品相关的目标制品的数量，确定变更源制品后可能影响的范围，以进行变更影响分析，比如 Aung 等人[154]通过自动化追踪链接恢复方法的文献研究，分析了需求、设计、测试以及程序影响集，即分别确定其影响范围，代替了人工递归验证的方法，减少了时间成本和出错率.

维护任务实施. 软件功能维护的实施阶段，当进行变更影响分析之后，如果确定变更当前的软件制品，则需要对该软件制品及其受影响的制品进行相应的变更，以解决软件维护任务. 需求追踪关系的建立是软件维护的一项基本任务，可以使开发人员了解软件制品间的关联关系. 同时在软件维护阶段，当使用追踪链接解决任务时，比如需求变更时对软件需求、设计、代码、测试等所有相关制品的修改，根据追踪链接可以快速准确地定位目标制品，缩小高能力和低能力学科以及学术和行业学科之间的差距. Jaber 等人[6]通过研究工业界和学术界的 28 名试验者发现，使用追踪链接的试验者在解决软件维护任务时准确率提高了 86.06%.

系统安全保障.软件性能维护阶段.能力成熟度集成模型明确地指出,在开发过程中维护需求的可追踪性及软件制品之间的可追踪性是安全攸关领域标准的基本要求,也是保障系统安全的重要前提[155].文献[8]研究安全攸关嵌入式系统的需求追踪方法,用于支持嵌入式系统的安全性分析以及系统的维护与演化,避免了由此产生的设计缺陷甚至是安全性事故.通常情况下,对于安全性要求较高的嵌入式软件系统,均有安全性要求相关的文件,根据安全规定和追踪链接可以准确获取安全相关的需求说明、产品设计、代码文档以及测试用例,以便检查所有的安全规定在每个环节是否均被考虑和实现.

综上所述,需求追踪可以确保软件系统开发过程中前后的一致性和功能的完整性,提高软件开发效率以及软件开发过程的整体性.同时,高效维护需求变更时的软件制品,并通过追踪链接快速检查软件系统的安全性要求,提高软件系统的维护效率.因此,需求追踪对于利益相关者是至关重要的,包括开发者、测试者、项目管理者、产品管理者以及商业分析师等.然而,需求追踪应用方面仍面临许多挑战:(1)追踪对象是不确定的,每个软件开发阶段产生的软件制品是多样的,应明确要追踪的软件制品,同时,对于特定的软件制品,也有不同粒度的追踪链接,应根据实际情况选择确定粒度的追踪对象,促进需求追踪的应用;(2)缺乏有效的方法进行追踪查询,一个大型软件系统包含的有效链接数量达到上万条,很难完全查找到所需的追踪链接;(3)追踪链接的可信度不高,高质量的追踪链接可以增加工业实践者的信任度,提高需求追踪技术在工业界的利用率.

4 案例研究

为了验证需求追踪技术的有效性,本研究小组依托承担的软件工程领域的智能化软件开发项目,对需求与代码间的追踪技术展开了研究,其目的是提高软件开发整体性和开发效率,完成代码推荐任务以及基础代码智能库的生成.

4.1 基于信息检索的需求追踪关系的建立

根据对当前需求追踪技术的调查发现,现阶段研究问题主要有三点:(1)基于信息检索方法的追踪质量高度依赖所提供的文本文档,低质量的文本文档甚至会使追踪结果没有参考价值;(2)没有充分利用 UML 图中的结构和逻辑信息,使得追踪结果准确性较低;(3)目前仍缺乏中文的需求文档与代码间追踪关系建立技术的研究.

针对以上问题,本项目在现有信息检索方法的基础上,结合 UML 图对中文描述的需求文档完成了需求的可追踪性研究.图 7 所示为项目中采用的方法框架图,其中输入有三个部分:需求文档、代码以及 UML 图.首先提取需求文档和代码文本特征项,使用信息检索的概率模型进行相似度匹配,通过设置相似度阈值筛选得到信息检索方法的分析结果;其次根据 UML 图的特征,包括类图中的继承关系、用例图中参与者和用例之间的关系以及活动图中的活动集内关联比例等,通过提取其中隐含的逻辑结构推测需求与代码间可能的追踪关系,以修正信息检索分析结果中的某些判定,最终得到需求与代码间的追踪关系.

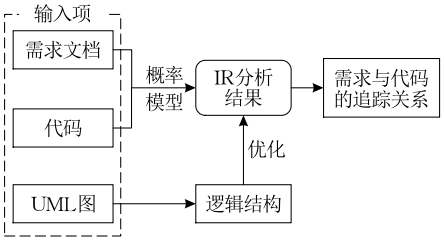


图 7 引入 UML 图的需求与代码间追踪关系建立框架

本方法重点分析类图、用例图以及活动图这三种 UML 图的逻辑结构,原因在于这三种图提供的信息与需求追踪关系的建立密切相关.对于类图,主要目的是找到有继承关系的类,并把它们归纳为同一个簇,若一个簇内与某个需求相关联的类的比例过少,则认为是误判,并将整个簇内的所有类都与该需求解除追踪关系,否则反之;对于用例图,主要目标是描述参与者和用例之间的关系,因为参与者对应类图中的实体,用例场景在需求中有所描述,从而建立需求和代码类之间的关系;对于活动图,主要任务是描述函数内部的业务细节流程,因此,本方法将活动图内所有活动划分为一个活动簇,类似于类图,结合用例图分析结果,根据活动簇内的关联比例合理推测出潜在的追踪关系.

为了表明本方法的有效性,本项目设计了基于概率模型的信息检索方法与本方法的对比实验,以 Precision 作为评估指标,实验项目为本地项目,结果如表 15 所示.其中,阈值为相似度阈值.实验结果表明,本项目提出的方法相比单纯的信息检索方法,准确率有一定的提高,尤其是有关联部分的准确率提高了 10%左右,表明本方法有助于发现潜在的追踪关系.

表 15　实验结果

阈值	Precision			
	IR		本方法	
	最终结果	关联结果	最终结果	关联结果
0.12	0.58	0.42	0.63	0.52
0.20	0.63	0.36	0.68	0.46
0.35	0.61	0.16	0.66	0.19

4.2　工业应用及评估

在实际项目中，由于开发人员对结果高准确性和高可靠性的要求，软件制品间追踪关系的建立仍主要采用手动的方式. 表 16 为本项目通过手动建立的需求与代码间的正向追踪表，并且追踪关系是人通过追踪表的形式验证的. 然而，该方法消耗了大量的人力资源. 同时，由于程序员的遗漏或疏忽，软件制品间往往缺乏更细粒度的语义关联或生成很多无效的需求追踪链接，因此，采用自动化或半自动化的方式建立需求追踪关系并且保证追踪结果的准确率是本项目研究的重点内容.

表 16　需求的正向追踪性

软件需求		对应实现的程序名称	
需求名称/标识	本文档的章节号	部件名称/标识	程序名称
代理功能/TS_DL	2.1.2.5	XX 安装/TS-DAZ	RJAZDlg.h、RemoteWorker.h RJAZDlg.cpp、RemoteWorker.cpp
		XX 更新/TS-DGX	RJSJDlg.h、RemoteWorker.h RJSJDlg.cpp、RemoteWorker.cpp
		段卸载/TS-DXZ	RJXZDlg.h、RemoteWorker.h RJXZDlg.cpp、RemoteWorker.cpp

为了验证提出的基于信息检索方法改进策略的有效性，本研究小组将该方法集成到需求追踪工具中，并部署在中国电子科技集团公司某研究所的实际系统中. 具体操作步骤包括：(1) 提供实际系统的需求文本、代码类文件以及 UML 设计图；(2) 利用基于信息检索的方法建立需求与代码间的追踪关系，并使用 UML 图对需求追踪候选链接进行过滤；(3) 将需求与代码间的追踪链接结果与专家手动标注的结果进行对比，计算需求追踪结果的准确性. 通过在 5 个实际项目中运行，并与多个专家手动标注的综合结果进行对比，本方法的准确率分别为 64%、67%、63%、65%和 62%. 实验结果表明，本研究小组提出的方法是有效的. 除此之外，在工具运行 15 天之后，针对使用该工具的 20 位开发人员，本研究小组进行了简短的问卷调查和采访，询问他们关于工具的效果和未来需要提升的方向，共设计了以下 4 个问题：

Q1：您认为需求追踪工具的引入可以减少人力劳动吗？

Q2：您认为需求追踪工具建立的需求与代码间的追踪关系准确吗？

Q3：您认为需求追踪工具可以提高软件开发效率吗？

Q4：您认为需求追踪工具未来还有哪些需要改进的方面？

其中，前 3 个问题均是有关工具效果的问题，本研究小组分别设置了 5 个选项：非常同意、同意、一般、不同意以及非常不同意，开发人员可以根据实际想法进行相应的选择. 第 4 个问题是关于需求追踪工具未来的改进方向，目的是了解开发人员的实际需求，以便进行后续改进.

表 17 展示了本研究小组对开发人员进行问卷调查的前 3 个问题的结果，记录了每个问题中每个选项对应的人数. 从表中可以看出，18 位(90%)开发人员同意需求追踪工具的引入可以减少人力劳动，只有 2 位(10%)开发人员持中立意见，结果表明引入需求追踪工具在很大程度上可以减少人力劳动. 同时，有超过一半(70%)的开发人员认为，通过需求追踪工具建立的需求与代码间的追踪关系是准确的，4 位开发人员保持中立，2 位开发人员认为需求追踪关系不准确，可以认为该工具建立的大多数需求与代码间的追踪关系都是准确的. 此外，20 位开发人员均同意需求追踪工具可以提高软件开发效率，在很大程度上有助于代码推荐、基础代码智能库生成等智能化软件开发任务的进行. 同时，开发人员认为该工具在用户界面的友好性和操作的便捷性方面可以进一步提高.

表 17　开发人员调查结果

问题	非常同意	同意	一般	不同意	非常不同意
Q1	6	12	2	0	0
Q2	3	11	4	2	0
Q3	16	4	0	0	0

通过以上实例研究表明，建立需求与软件制品间的追踪关系可以有效地帮助工业实践者完成软件开发任务，验证了需求追踪技术的有效性.

5 相关工作

近年来,已有部分研究学者对需求追踪技术进行了系统的整理与分析.本次共调查 11 篇,具体情况如表 18 所示,描述了不同文献调查的时间范围、研究方法以及研究主题,并将本综述的研究内容与调查的研究文献进行对比,突出本综述的研究重点.

表 18 相关研究总结

研究文献	时间范围	研究方法	研究主题
[154]	2012～2019	文献综述	基于变更影响分析的自动化可追踪性方法研究
[156]	1997～2007	系统性文献综述	需求可追踪性分析与工业案例研究
[157]	2012 年以前	系统性文献综述	基于模型驱动工程的可追踪性管理
[158]	1993～2013	文献综述	需求工程会议上有关需求可追踪性的话题研究
[159]	2013 年以前	系统性文献综述	基于信息检索方法的软件可追踪性系统映射
[160]	2004～2014	调查研究	软件可追踪性的发展趋势和未来方向分析
[161]	2006～2016	系统性文献综述	需求的可追踪性技术分析和技术转移决策支持
[162]	2009 年以前	文献综述	动态需求跟踪方法及跟踪精度问题研究
[163]	2002～2017	调查研究	近 15 年自动化需求追踪研究的数据集来源
[164]	2010～2017	文献综述	基于信息检索的需求跟踪方法研究
[165]	2010～2017	系统性文献综述	需求追踪技术和工具的分析
本文	2011～2020	系统性文献综述	研究需求与不同软件制品间追踪关系建立技术进展;总结实验数据集、工具以及需求追踪技术的应用场景;以实际项目案例展开研究

其中,文献[156-157,159,161,165]均采用系统性文献综述的方法,即遵循 SLR 方法的整体框架,包括研究问题、文献检索、质量评估以及数据抽取与整合等 4 个方面,对研究文献展开科学地分析与总结,使得文献综述结果更加可靠;文献[154,158,162,164]采用普通文献综述的方法,通过检索文献分析得出结果,并分析未来的发展趋势;文献[160,163]则采用调查研究的方法,强调客观事实,不外加个人观点.除此之外,也有研究学者针对近 10 年的研究文献进行整理与分析.文献[154]研究了 2012 年～2019 年间发表的相关文献,主要关注了自动化追踪链接恢复方法,根据变更影响分析集评估不同构件类型间的追踪链接.文献[164]研究了 2010 年～2017 年间发表的相关文献,以信息检索方法为研究对象,讨论了现有的 IR 模型、相应的改进策略、工具研发情况以及实验度量指标.同时文献[165]研究了同一时间段内发表的相关文献,主要分析了需求追踪方法模型和工具,并评估其优缺点.

然而,现有综述主要围绕需求追踪方法,介绍需求追踪关系建立时常用的关键技术,缺乏对软件全生命周期中不同软件开发阶段产生的软件制品的分析,没有重点关注需求与不同软件制品间追踪关系的建立,并且当前没有研究将需求追踪技术应用于智能化软件开发项目中.为了提高软件开发整体性,本文对 2011 年～2020 年间的发表的论文进行整理与分析,研究了需求与不同软件制品间追踪关系建立技术进展,并对常用的需求追踪技术进行对比分析,突出当前的研究成果;总结了实验设计时的开源项目数据以及工具研发情况;分析了需求追踪技术的实际应用场景.同时,以实际项目案例展开研究,验证需求追踪技术在智能化软件开发项目中的有效性.

6 研究趋势与展望

近年来,由于外在智能技术的发展和内在应用场景的需求,推动了需求追踪技术在整个软件生命周期中的应用,包括软件工程中的各种软件开发与维护任务.现有的工作主要围绕需求与需求、设计、代码以及测试间追踪关系的建立技术进行展开,重点利用基于信息检索和基于机器学习的方法,实现了需求追踪过程的自动化或半自动化,提高了需求与软件制品间追踪关系的准确性,并设计和实现可视化工具用于辅助需求追踪关系的建立,但相关理论和技术还不够成熟.

通过研究现状分析与实际应用需求,本文认为以下方面研究在未来工作中值得关注:

(1)面向软件全生命周期的需求追踪方法

当前的需求追踪关键技术主要集中于需求与代码间追踪关系的建立,其他需求追踪链接类型关注度较少.Jaber 等人[6]通过工业调查发现,行业利益相关者合作时最需要的追踪链接的类型是多样的,它包括需求、设计、代码、测试等不同生命周期阶段的软件制品追踪链接,因此研究学者应考虑不只一种特定领域的需求追踪关系的建立.除此之外,为满足不同的利益相关者的需求,应建立多粒度制品间

追踪关系，比如需求与源代码中某一具体方法或函数间可追踪性的建立. 在此基础上，研究人员应考虑将研发的需求追踪工具与不同软件工程阶段开发的工具进行整合与桥接.

(2) 面向非功能需求的追踪关系建立

在实际软件开发过程中，需求制品通常包括功能需求和非功能需求. 由于功能需求占据主要地位，且易于追踪，因此当前对功能需求的可追踪性研究较多. 然而，非功能需求是软件系统中的高质量约束，关系到软件体系结构设计的稳定性，因此建立非功能需求与不同软件制品间的追踪关系仍是需求追踪研究的重要内容. 非功能需求与系统的总体属性相关，包括容错性、安全性、可用性、可扩展性以及其他关键的质量特性，属于横切方面的问题，在设计过程中通常难以解释. 因此，目前已经有研究学者通过 NFR 框架来将抽象和模糊的非功能需求转化为更加详细的细节和描述，以作为设计的基础，但是自动化程度不高，且依赖于特定领域. 在未来工作中，研究人员应进一步关注非功能需求、软件架构及其相关制品间的关联信息，并应用现有的信息检索技术以及机器学习技术，建立非功能需求与各种制品间的追踪链接，提高通用性，使得在大规模的软件维护和演化过程中保障软件系统的性能，包括安全性以及稳定性等.

(3) 基于智能技术的追踪方法优化

早期研究主要使用手动的方式建立需求与不同软件制品间的追踪关系，具有较高的准确率，但其依赖于特定的领域或软件制品. 近些年，研究者们提出了基于信息检索和机器学习的方法，在保证一定准确率的前提下提高了需求追踪技术的自动化程度.

然而，基于信息检索的方法对软件制品本身的质量要求较高，且经常由于“词汇失配”问题导致需求追踪结果准确性较低. 针对该问题，研究人员应考虑使用其他智能技术来提取软件制品中的语义信息，以便在向量空间中进行相似度评估. 比如，使用词嵌入和自注意力机制等特征表示技术对文本中的语义信息进行学习，通过对不同的单词分配自注意力权重来生成文本的语义向量. 除此之外，可以使用图挖掘技术表征软件制品中的依赖关系，使用文本的上下文信息来建立需求与制品间的追踪关系.

当前基于机器学习的方法在构建初始训练数据集时需要领域专家手动标注大量的追踪链接. 未来工作中，研究人员应考虑使用其他智能技术来优化资源和减少成本. 比如，使用迁移学习技术将其他项目中已标注的追踪信息进行迁移，包括其他特定领域中的项目，用于对新项目中未标注的数据集直接进行预测，提高需求追踪方法的通用性，自动建立需求追踪链接. 同时，可以使用主动学习的技术对少量标注的链接进行学习，以自动识别有效或无效链接.

(4) 面向需求追踪关系的高效搜索技术

对于大规模的软件项目，随着软件制品种类和数量的增加，追踪链接的数量也显著扩增. 然而，现有的追踪链接主要靠人工或者文本文件自带的查找方式进行搜索，缺乏对软件制品本身的搜索技术，阻碍了需求追踪技术在工业界的应用. 因此，研究高效的搜索技术对需求追踪链接进行快速查找是必要的. 采用多维索引的高效搜索算法，可以提升软件工程中其他任务的开发效率，同时节省人力资源的消耗. 比如在软件制品库的基础上，从不同表达形式的维度来建立制品的多维索引，包括不同软件制品的开发语言、应用领域、所属组织、评价星级等. 除此之外，结合机器学习方法的性能优势，可以提高搜索的精确度，同时能够在软件制品的多维索引以及搜索条件下快速准确的给出搜索结果.

(5) 需求追踪方法的多维度评估

对需求追踪方法进行正确且有效地评估，尤其是在人工智能技术应用的背景下，增加可解释性、可测试性、可信性等多维度评估，不仅有助于改进需求追踪技术策略，也对提升工业界对需求追踪技术的可信度具有重要意义. 现有的工作主要采用较为基础且单一的度量指标对需求追踪结果进行正确性方面的质量评估，缺乏对追踪技术可用性、稳定性以及覆盖率方面的评估. 因此，研究人员可以进一步通过用户调查的方式评估技术是否可用，包括时间、资金和人力资源等耗费情况的分析. 同时，随着软件系统的演化，评估追踪技术是否一直有效，以验证其稳定性. 并且通过将需求追踪技术应用于不同类型的软件制品，以评估其制品空间的种类覆盖率，验证其通用性，提高需求追踪技术的整体质量.

(6) 需求追踪技术应用场景的扩展

目前，需求追踪技术已被广泛应用于软件工程中的重要任务，包括变更影响分析以及系统安全保障等，用于辅助软件开发和维护任务，提升软件开发效率. 未来需求追踪技术还可以进一步应用于其他软件工程活动，比如代码推荐、API 文档增强、以及程序语言翻译等. 其中，代码推荐可以利用需求与代码间的追踪关系建立技术来推荐相关的 API 给开发人员，减少开发人员查找、理解、组合和调试代码

的工作量. API 文档增强旨在建立 API 类型与从 Stack Overflow 转储文件生成的一系列问答对之间的追踪链接,利用需求追踪技术有助于提取 API 类型对应的高质量代码实例以及使用场景,并嵌入 API 文档. 程序语言翻译则根据需求追踪技术建立了不同语言间的联系,便于生成易于理解的文档.

致 谢 感谢审稿专家和编辑老师提出宝贵意见!

参 考 文 献

[1] Maro S, Steghöfer J, Staron M. Software traceability in the automotive domain: Challenges and solutions. Journal of Systems and Software, 2018, 141: 85-110

[2] Heumesser N, Houdek F. Experiences in managing an automotive requirements engineering process//Proceedings of the 12th IEEE International Requirements Engineering Conference (RE). Kyoto, Japan, 2004: 322-327

[3] Nuseibeh B, Easterbrook S M. Requirements engineering: A roadmap//Proceedings of the 22nd International Conference on the Future of Software Engineering (ICSE). Limerick, Ireland, 2000: 35-46

[4] Gotel O C Z, Finkelstein A. An analysis of the requirements traceability problem//Proceedings of the 1st IEEE International Conference on Requirements Engineering (ICRE). Colorado, USA, 1994: 94-101

[5] Berry D M. Aybüke aurum and claes wohlin (eds): Engineering and managing software requirements. Requirements Engineering, 2006, 11(2): 152-154

[6] Jaber K, Sharif B, Liu C. A study on the effect of traceability links in software maintenance. IEEE Access, 2013, 1: 726-741

[7] Ibrahim S, Idris N B, Munro M, et al. A requirements traceability to support change impact analysis. Asian Journal of Information Technology, 2005, 4(4): 345-355

[8] Wang Fei, Huang Zhi-Qiu, Yang Zhi-Bin, et al. A requirements traceability approach for safety-critical embedded system. Chinese Journal of Computers, 2018, 41(3): 652-669(in Chinese)
(王飞, 黄志球, 杨志斌等. 一种安全攸关嵌入式系统需求追踪方法. 计算机学报, 2018, 41(3): 652-669)

[9] Naur P, Randell B. Software engineering: Report of a conference sponsored by the NATO science committee. Brussels, Belgium: NATO Scientific Affairs Division, 1969

[10] Dorfman M, Thayer R H. Standards, Guidelines and Examples on System and Software Requirements Engineering. Washington, USA: IEEE Computer Society Press, 1990

[11] Watkins R, Neal M. Why and how of requirements tracing. IEEE Software, 1994, 11(4): 104-106

[12] Mäder P, Egyed A. Do developers benefit from requirements traceability when evolving and maintaining a software system? Empirical Software Engineering, 2015, 20(2): 413-441

[13] Jadoon G, Shafi M, Jan S. A model-oriented requirements traceability framework for small and medium software industries //Proceedings of the 20th International Arab Conference on Information Technology (ACIT). Al Ain, United Arab Emirates, 2019: 91-96

[14] Gazzawe F, Lock R, Dawson C W. Use of ontology in identifying missing artefact links//Proceedings of the 7th International Conference on Software and Computer Applications (ICSCA). Kuantan, Malaysia, 2018: 6-9

[15] Spanoudakis G, Zisman A, Pérez-miñana E, et al. Rule-based generation of requirements traceability relations. Journal of Systems and Software, 2004, 72(2): 105-127

[16] Zhou J, Lu Y, Lundqvist K. A context-based information retrieval technique for recovering use-case-to-source-code trace links in embedded software systems//Proceedings of the 39th Euromicro Conference on Software Engineering and Advanced Applications (SEAA). Santander, Spain, 2013: 252-259

[17] Wieloch M, Amornborvornwong S, Cleland-Huang J. Trace-by-classification: A machine learning approach to generate trace links for frequently occurring software artifacts//Proceedings of the 7th International Workshop on Traceability in Emerging Forms of Software Engineering (TEFSE). California, USA, 2013: 110-114

[18] Gervasi V, Zowghi D. Supporting traceability through affinity mining//Proceedings of the 22nd IEEE International Requirements Engineering Conference (RE). Karlskrona, Sweden, 2014: 143-152

[19] Filho G A C, Lencastre M, Rodrigues A, et al. RETRATOS: Requirement traceability tool support//Proceedings of the 21st IEEE International Requirements Engineering Conference (RE). Rio de Janeiro, Brazil, 2013

[20] Ramesh B, Jarke M. Toward reference models for requirements traceability. IEEE Transactions on Software Engineering, 2001, 27(1): 58-93

[21] Rempel P, Mäder P, Kuschke T. Towards feature-aware retrieval of refinement traces//Proceedings of the 7th International Workshop on Traceability in Emerging Forms of Software Engineering (TEFSE). San Francisco, USA, 2013: 100-104

[22] Dekhtyar A, Hayes J H, Antoniol G. Benchmarks for traceability?//Proceedings of the 1st International Workshop on Traceability in Emerging Forms of Software Engineering (TEFSE). New York, USA, 2007: 1-10

[23] Wohlin C, Runeson P, Höst M, et al. Experimentation in software engineering — An introduction. Massachusetts, USA: Kluwer Academic, 2000

[24] Mahmood K, Takahashi H, Alobaidi M. A semantic approach

for traceability link recovery in aerospace requirements management system//Proceedings of the 12th IEEE International Symposium on Autonomous Decentralized Systems (ISADS). Taichung, China, 2015: 217-222

[25] Eder S, Femmer H, Hauptmann B, et al. Configuring latent semantic indexing for requirements tracing//Proceedings of the 2nd IEEE/ACM International Workshop on Requirements Engineering and Testing (RET). Florence, Italy, 2015: 27-33

[26] Dominik W, Udo L. Manage interdisciplinarity based on requirements traceability: A graph-based tool support for requirements traceability//Proceedings of the 17th Portland International Conference on Management of Engineering and Technology (PICMET). Portland, USA, 2017: 1-8

[27] Li Y, Schulze S, Saake G. Reverse engineering variability from requirement documents based on probabilistic relevance and word embedding//Proceedings of the 22nd International Systems and Software Product Line Conference (SPLC). Oregon, USA, 2018: 121-131

[28] Jain R, Ghaisas S, Sureka A. SANAYOJAN: A framework for traceability link recovery between use-bases in software requirement specification and regulatory documents//Proceedings of the 3rd International Workshop on Realizing Artificial Intelligence Synergies in Software Engineering (RAISE). Hyderabad, India, 2014: 12-18

[29] Hallerstede S, Jastram M, Ladenberger L. A method and tool for tracing requirements into specifications. Science of Computer Programming, 2014, 82: 2-21

[30] Murtazina M S, Avdeenko T. An ontology-based approach to support for requirements traceability in agile development. Procedia Computer Science, 2019, 150: 628-635

[31] Wang B, Peng R, Wang Z, et al. Combining VSM and BTM to improve requirements trace links generation//Proceedings of the 31st International Conference on Software Engineering and Knowledge Engineering (SEKE). Lisbon, Portugal, 2019: 567-747

[32] Saputri T R D, Lee S. Ensuring traceability in modeling requirement using ontology-based approach//Proceedings of the 3rd Asia-Pacific Requirements Engineering Symposium (APRES). Nagoya, Japan, 2016: 3-17

[33] Li Z, Chen M, Huang L, et al. Recovering traceability links in requirements documents//Proceedings of the 19th Conference on Computational Natural Language Learning (CoNLL). Beijing, China, 2015: 237-246

[34] Sultanov H, Hayes J H, Kong W. Application of swarm techniques to requirements tracing. Requirements Engineering, 2011, 16(3): 209-226

[35] Filax M, Gonschorek T, Ortmeier F. Building models we can rely on: Requirements traceability for model-based verification techniques//Proceedings of the 5th International Symposium on Model-based Safety and Assessment (IMBSA). Trento, Italy, 2017: 3-18

[36] Carniel C A, Pegoraro R A. Metamodel for requirements traceability and impact analysis on agile methods//Proceedings of the 8th Brazilian Workshop on Agile Methods (WBMA). Belém, Brazil, 2017: 105-117

[37] Lockerbie J, Maiden N A M, Williams C, et al. A requirements traceability approach to support mission assurance and configurability in the military//Proceedings of the 23rd International Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ). Essen, Germany, 2017: 308-323

[38] Broy M. A logical approach to systems engineering artifacts: Semantic relationships and dependencies beyond traceability—From requirements to functional and architectural view. Software and Systems Modeling, 2018, 17(2): 365-393

[39] Mezghani M, Kang J, Kang E, et al. Clustering for traceability managing in system specifications//Proceedings of the 27th IEEE International Requirements Engineering Conference (RE). Jeju Island, Korea, 2019: 257-264

[40] Goknil A, Kurtev I, van den Berg K, et al. Semantics of trace relations in requirements models for consistency checking and inferencing. Software and Systems Modeling, 2011, 10(1): 31-34

[41] Zheng Pei-Zhen, Yuan Chun-Chun, Liu Chao, et al. Traceability model oriented to software safety requirement analysis process. Computer Science, 2017, 44(4): 30-34(in Chinese) (郑培真, 苑春春, 刘超等. 面向软件安全性需求分析过程的追踪模型. 计算机科学, 2017, 44(4): 30-34)

[42] Alenazi M, Niu N, Savolainen J. A novel approach to tracing safety requirements and state-based design models//Proceedings of the 42nd International Conference on Software Engineering (ICSE). Seoul, Korea, 2020: 848-860

[43] Haidrar S, Anwar A, Roudiès O. A SysML-based approach to manage stakeholder requirements traceability//Proceedings of the 14th IEEE/ACS International Conference on Computer Systems and Applications (AICCSA). Hammamet, Tunisia, 2017: 202-207

[44] Wang F, Yang Z, Huang Z, et al. An approach to generate the traceability between restricted natural language requirements and AADL models. IEEE Transactions on Reliability, 2020, 69(1): 154-173

[45] Yu D, Geng P, Wu W. Constructing traceability between features and requirements for software product line engineering //Proceedings of the 19th Asia-Pacific Software Engineering Conference (APSEC). Hong Kong, China, 2012: 27-34

[46] Haidrar S, Anwar A, Roudies O. On the use of model transformation for requirements trace models generation//Proceedings of the 3rd International Conference on Wireless Technologies, Embedded and Intelligent Systems (WITS). Fez, Morocco, 2017: 1-6

[47] Li Y, Cleland-Huang J. Ontology-based trace retrieval//Proceedings of the 7th International Workshop on Traceability in Emerging Forms of Software Engineering (TEFSE). California, USA, 2013: 30-36

[48] Guo J, Cheng J, Cleland-Huang J. Semantically enhanced software traceability using deep learning techniques//Proceedings of the 39th International Conference on Software Engineering (ICSE). Buenos Aires, Argentina, 2017: 3-14

[49] Haidrar S, Anwar A, Roudiès O. Towards a generic framework for requirements traceability management for SysML language //Proceedings of the 4th IEEE International Colloquium on Information Science and Technology (CiSt). Tangier, Morocco, 2016: 210-215

[50] Noyer A, Iyenghar P, Pulvermüller E, et al. Traceability and interfacing between requirements engineering and UML domains using the standardized ReqIF format//Proceedings of the 3rd International Conference on Model-Driven Engineering and Software Development (MODELSWARD). Loire Valley, France, 2015: 370-375

[51] Mirakhorli M. Tracing architecturally significant requirements: A decision-centric approach//Proceedings of the 33rd International Conference on Software Engineering (ICSE). Hawaii, USA, 2011: 1126-1127

[52] Taromirad M, Paige R F. Agile requirements traceability using domain-specific modelling languages//Proceedings of the 1st Extreme Modeling Workshop Proceedings (XM). Innsbruck, Austria, 2012: 45-50

[53] Udagawa Y. Enhancing information retrieval to automatically trace requirements and design artifacts//Proceedings of the 13th International Conference on Information Integration and Web-based Applications and Services (iiWAS). Ho Chi Minh City, Vietnam, 2011: 292-295

[54] Li Z, Chen M, Huang L, et al. Tracing requirements in software design//Proceedings of the 2nd International Conference on Software and System Process (ICSSP). Paris, France, 2017: 25-29

[55] Vidal E J, Villota E R. SysML as a tool for requirements traceability in mechatronic design//Proceedings of the 4th International Conference on Mechatronics and Robotics Engineering (ICMRE). Valenciennes, France, 2018: 146-152

[56] Yoshino K, Matsuura S. Requirements traceability management support tool for UML models//Proceedings of the 9th International Conference on Software and Computer Applications (ICSCA). Langkawi, Malaysia, 2020: 163-166

[57] Nejati S, Sabetzadeh M, Falessi D, et al. A SysML-based approach to traceability management and design slicing in support of safety certification: Framework, tool support, and case studies. Information and Software Technology, 2012, 54(6): 569-590

[58] Maté A, Trujillo J. A trace metamodel proposal based on the model driven architecture framework for the traceability of user requirements in data warehouses. Information Systems, 2012, 37(8): 753-766

[59] Goknil A, Kurtev I, van den Berg K. Generation and validation of traces between requirements and architecture based on formal trace semantics. Journal of Systems and Software, 2014, 88: 112-137

[60] Wolfenstetter T, Basirati M R, Böhm M, et al. Introducing trails: A tool supporting traceability, integration and visualisation of engineering knowledge for product service systems development. Journal of Systems and Software, 2018, 144: 342-355

[61] Mäder P, Gotel O. Towards automated traceability maintenance. Journal of Systems and Software, 2012, 85(10): 2205-2227

[62] Marcén A C, Lapeña R, Pastor O, et al. Traceability link recovery between requirements and models using an evolutionary algorithm guided by a learning to rank algorithm: Train control and management case. Journal of Systems and Software, 2020, 163: 110519-110543

[63] Ríos-Zapata D, Pailhès J, Mejía-Gutiérrez R. Multi-layer graph theory utilisation for improving traceability and knowledge management in early design stages. Procedia CIRP, 2017, 60: 308-313

[64] Badreddin O B, Sturm A, Lethbridge T C. Requirement traceability: A model-based approach//Proceedings of the 4th IEEE International Model-Driven Requirements Engineering Workshop (MoDRE). Karlskrona, Sweden, 2014: 87-91

[65] Ou A Y, Rahmaniheris M, Jiang Y, et al. SafeTrace: A safety-driven requirement traceability framework on device interaction hazards for MD PnP//Proceedings of the 33rd Annual ACM Symposium on Applied Computing (SAC). Pau, France, 2018: 1282-1291

[66] Gazzawe F, Lock R, Dawson C W. Traceability framework for requirement artefacts//Proceedings of the 8th Science and Information Conference (SAI). London, UK, 2020: 97-109

[67] Tang A, Liang P, Clerc V, et al. Traceability in the co-evolution of architectural requirements and design. Relating Software Requirements and Architectures. Berlin: Springer-Verlag, 2011: 35-60

[68] Kchaou D, Bouassida N, Mefteh M, et al. Recovering semantic traceability between requirements and design for change impact analysis. Innovations in Systems and Software Engineering, 2019, 15(2): 101-115

[69] Pavalkis S, Nemuraite L. Process for applying derived property based traceability framework in software and systems development life cycle//Proceedings of the 19th International Conference on Information and Software Technologies (ICIST). Kaunas, Lithuania, 2013: 122-133

[70] Wen L, Tuffley D, Dromey R G. Formalizing the transition from requirements' change to design change using an evolutionary traceability model. Innovations in Systems and Software Engineering, 2014, 10(3): 181-202

[71] Lapeña R, Pérez F, Cetina C, et al. Improving traceability links recovery in process models through an ontological expansion of requirements//Proceedings of the 31st International Conference on Advanced Information Systems Engineering (CAiSE). Rome, Italy, 2019: 261-275

[72] Ishibashi S, Hisazumi K, Nakanishi T, et al. Model-based methodology establishing traceability between requirements,

design and operation information in lifecycle-oriented architecture. New Trends in E-service and Smart Computing. Cham, Switzerland: Springer, 2018: 47-63

[73] Tang Chen, Li Yong-Hua, Rao Meng-Ni, et al. Semantic judgement method of polysemous keywords in dynamic requirement traceability. Journal of Computer Applications, 2019, 39(5): 1299-1304(in Chinese)
(唐晨，李勇华，饶梦妮等. 动态需求跟踪中多义关键词的语义判断方法. 计算机应用，2019，39(5)：1299-1304)

[74] Li Xiao, Wei Chang-Jiang. Requirements tracking method among multi-view meta-model. Computer Systems & Application, 2019, 28(9): 41-49(in Chinese)
(李潇，魏长江. 多视点元模型间需求追踪性方法. 计算机系统应用，2019，28(9)：41-49)

[75] Deng Liu-Meng, Shen Guo-Hua, Huang Zhi-Qiu, et al. Extended SysML for supporting requirements trace model automatic generation. Journal of Frontiers of Computer Science & Technology, 2019, 13(6): 950-960(in Chinese)
(邓刘梦，沈国华，黄志球等. 扩展 SysML 支持需求追踪模型的自动生成. 计算机科学与探索，2019，13(6)：950-960)

[76] Martins J C C, Machado R J. Ontologies for product and process traceability at manufacturing organizations: A software requirements approach//Proceedings of the 8th International Conference on the Quality of Information and Communications Technology (QUATIC). Lisbon, Portugal, 2012: 353-358

[77] Toranzo Céspedes M A, Cysneiros Filho G, Gómez Y, et al. Towards a framework for improving requirement traceability. Ingeniería e Investigación, 2012, 32(1): 48-52

[78] Song S, Kim Y, Park S, et al. A non-functional requirements traceability management method based on architectural patterns. Computers, Networks, Systems, and Industrial Engineering 2011. Berlin, Germany: Springer-Verlag, 2011: 25-35

[79] Ziftci C, Krueger I. Tracing requirements to tests with high precision and recall//Proceedings of the 26th IEEE/ACM International Conference on Automated Software Engineering (ASE). Kansas, USA, 2011: 472-475

[80] Aichernig B K, Hörmaier K, Lorber F, et al. Require, test, and trace IT. International Journal on Software Tools for Technology Transfer, 2017, 19(4): 409-426

[81] Azmi A, Ibrahim S. Test management traceability model to support software testing documentation//Proceedings of the 1st International Conference on Digital Information and Communication Technology and Its Applications (DICTAP). Dijon, France, 2011: 21-32

[82] Mustafa N, Labiche Y. Using semantic web to establish traceability links between heterogeneous artifacts//Proceedings of the 12th International Joint Conference on Software Technologies (ICSOFT). Madrid, Spain, 2017: 91-113

[83] Espinoza A, Garbajosa J. A study to support agile methods more effectively through traceability. Innovations in Systems and Software Engineering, 2011, 7(1): 53-69

[84] Wang B, Peng R, Wang Z, et al. An automated hybrid approach for generating requirements trace links. International Journal on Software Engineering and Knowledge Engineering, 2020, 30(7): 1005-1048

[85] Mills C, Haiduc S. A machine learning approach for determining the validity of traceability links//Proceedings of the 39th International Conference on Software Engineering (ICSE). Buenos Aires, Argentina, 2017: 121-123

[86] Faiz F, Easmin R, Gias A U. Achieving better requirements to code traceability: Which refactoring should be done first?//Proceedings of the 10th International Conference on the Quality of Information and Communications Technology (QUATIC). Lisbon, Portugal, 2016: 9-14

[87] Ali N, Sharafi Z, Guéhéneuc Y, et al. An empirical study on requirements traceability using eye-tracking//Proceedings of the 28th IEEE International Conference on Software Maintenance (ICSM). Trento, Italy, 2012: 191-200

[88] Du T, Shen G, Huang Z, et al. Automatic traceability link recovery via active learning. Frontiers of Information Technology and Electronic Engineering, 2020, 21(8): 1217-1225

[89] Rodriguez D V, Carver D L. An IR-based artificial bee colony approach for traceability link recovery//Proceedings of the 32nd IEEE International Conference on Tools with Artificial Intelligence (ICTAI). Maryland, USA, 2020: 1145-1153

[90] Kuang H, Nie J, Hu H, et al. Analyzing closeness of code dependencies for improving IR-based traceability recovery//Proceedings of the IEEE 24th International Conference on Software Analysis, Evolution and Reengineering (SANER). Klagenfurt, Austria, 2017: 68-78

[91] Kuang H, Mäder P, Hu H, et al. Do data dependencies in source code complement call dependencies for understanding requirements traceability?//Proceedings of the 28th IEEE International Conference on Software Maintenance (ICSM). Trento, Italy, 2012: 181-190

[92] Rahimi M, Goss W, Cleland-Huang J. Evolving requirements-to-code trace links across versions of a software system//Proceedings of the 32nd IEEE International Conference on Software Maintenance and Evolution (ICSME). North Carolina, USA, 2016: 99-109

[93] Wang S, Li T, Yang Z. Exploring semantics of software artifacts to improve requirements traceability recovery: A hybrid approach//Proceedings of the 26th Asia-Pacific Software Engineering Conference (APSEC). Putrajaya, Malaysia, 2019: 39-46

[94] Charrada E B, Koziolek A, Glinz M. Identifying outdated requirements based on source code changes//Proceedings of the 20th IEEE International Requirements Engineering Conference (RE). Illinois, USA, 2012: 61-70

[95] Chen X, Grundy J C. Improving automated documentation to code traceability by combining retrieval techniques//Proceedings of the 26th IEEE/ACM International Conference on Automated Software Engineering (ASE). Kansas, USA, 2011: 223-232

[96] Kern M, Erata F, Iser M, et al. Integrating static code analysis toolchains//Proceedings of the 43rd IEEE Annual Computer Software and Applications Conference (COMPSAC). Wisconsin, USA, 2019: 523-528

[97] Ali N, Jaafar F, Hassan A E. Leveraging historical co-change information for requirements traceability//Proceedings of the 20th Working Conference on Reverse Engineering (WCRE). Koblenz, Germany, 2013: 361-370

[98] Rodriguez D V, Carver D L. Multi-objective information retrieval-based NSGA-II optimization for requirements traceability recovery//Proceedings of the 17th IEEE International Conference on Electro/Information Technology (EIT). 2020: 271-280

[99] Gethers M, Oliveto R, Poshyvanyk D, et al. On integrating orthogonal information retrieval methods to improve traceability recovery//Proceedings of the 27th IEEE International Conference on Software Maintenance (ICSM). Virginia, USA, 2011: 133-142

[100] Nagano S, Ichikawa Y, Kobayashi T. Recovering traceability links between code and documentation for enterprise project artifacts//Proceedings of the 36th Annual IEEE Computer Software and Applications Conference (COMPSAC). Izmir, Turkey, 2012: 11-18

[101] Zhou J. Requirements development and management of embedded real-time systems//Proceedings of the 22nd IEEE International Requirements Engineering Conference (RE). Karlskrona, Sweden, 2014: 479-484

[102] Ali N, Guéhéneuc Y, Antoniol G. Requirements traceability for object-oriented systems by partitioning source code//Proceedings of the 18th Working Conference on Reverse Engineering (WCRE). Limerick, Ireland, 2011: 45-54

[103] Ghannem A, Hamdi M S, Kessentini M, et al. Search-based requirements traceability recovery: A multi-objective approach//Proceedings of the 19th IEEE Congress on Evolutionary Computation (CEC). San Sebastián, Spain, 2017: 1183-1190

[104] Delater A, Paech B. Tracing requirements and source code during software development: An empirical study//Proceedings of the 7th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM). Maryland, USA, 2013: 25-34

[105] Ali N, Guéhéneuc Y, Antoniol G. Trust-based requirements traceability//Proceedings of the 19th IEEE International Conference on Program Comprehension (ICPC). Ontario, Canada, 2011: 111-120

[106] Ali N, Guéhéneuc Y, Antoniol G. TrusTrace: Mining software repositories to improve the accuracy of requirement traceability links. IEEE Transactions on Software Engineering, 2013, 39(5): 725-741

[107] Kuang H, Gao H, Hu H, et al. Using frugal user feedback with closeness analysis on code to improve IR-based traceability recovery//Proceedings of the 27th International Conference on Program Comprehension (ICPC). Montreal, Canada, 2019: 369-379

[108] Mahmoud A, Niu N. Source code indexing for automated tracing//Proceedings of the 6th International Workshop on Traceability in Emerging Forms of Software Engineering (TEFSE). Hawaii, USA, 2011: 3-9

[109] Ghabi A, Egyed A. Code patterns for automatically validating requirements-to-code traces//Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering (ASE). Essen, Germany, 2012: 200-209

[110] Florez J M. Automated fine-grained requirements-to-code traceability link recovery//Proceedings of the 41st International Conference on Software Engineering: Companion Proceedings (ICSE). Montreal, Canada, 2019: 222-225

[111] Blasco D, Cetina C, Pastor O. A fine-grained requirement traceability evolutionary algorithm: Kromaia, a commercial video game case study. Information and Software Technology, 2020, 119: 106235- 106247

[112] Ali N, Cai H, Hamou-Lhadj A, et al. Exploiting parts-of-speech for effective automated requirements traceability. Information and Software Technology, 2019, 106: 126-141

[113] Hübner P, Paech B. Interaction-based creation and maintenance of continuously usable trace links between requirements and source code. Empirical Software Engineering, 2020, 25(5): 4350-4377

[114] Nyamawe A S, Liu H, Niu Z, et al. Recommending refactoring solutions based on traceability and code metrics. IEEE Access, 2018, 6: 49460-49475

[115] Wang S, Li T, Yang Z. Using graph embedding to improve requirements traceability recovery//Proceedings of the 2nd International Conference on Applied Informatics (ICAI). Madrid, Spain, 2019: 533-545

[116] Mahmoud A, Niu N. Supporting requirements to code traceability through refactoring. Requirements Engineering, 2014, 19(3): 309-329

[117] Chhabra J K, et al. Requirements traceability through information retrieval using dynamic integration of structural and co-change coupling//Proceedings of the 1st International Conference on Advanced Informatics for Computing Research (ICAICR). Jalandhar, India, 2017: 107-118

[118] Tsuchiya R, Washizaki H, Fukazawa Y, et al. Interactive recovery of requirements traceability links using user feedback and configuration management logs//Proceedings of the 27th International Conference on Advanced Information Systems Engineering (CAiSE). Stockholm, Sweden, 2015: 247-262

[119] Ali N, Sharafi Z, Guéhéneuc Y, et al. An empirical study on the importance of source code entities for requirements traceability. Empirical Software Engineering, 2015, 20(2): 442-478

[120] Vinárek J, Hnetynka P, Simko V, et al. Recovering traceability links between code and specification through domain

model extraction//Proceedings of the 10th International Workshop on Enterprise and Organizational Modeling and Simulation(EOMAS). Thessaloniki, Greece, 2014: 187-201

[121] Tsuchiya R, Washizaki H, Fukazawa Y, et al. Recovering traceability links between requirements and source code using the configuration management log. IEICE Transactions on Information and Systems, 2015, 98(4): 852-862

[122] Shao J, Wu W, Geng P. An improved approach to the recovery of traceability links between requirement documents and source codes based on latent semantic indexing//Proceedings of the 13th International Conference on Computational Science and Its Applications(ICCSA). Ho Chi Minh City, Vietnam, 2013: 547-557

[123] Tsuchiya R, Kato T, Washizaki H, et al. Recovering traceability links between requirements and source code in the same series of software products//Proceedings of the 17th International Software Product Line Conference(SPLC). Tokyo, Japan, 2013: 121-130

[124] Du Tian-Bao, Shen Guo-Hua, Huang Zhi-Qiu, et al. Through code pattern to improve information retrieval-based requirements and code traceability recovery method. Journal of Chinese Mini-Micro Computer Systems, 2019, 40(5): 1107-1114(in Chinese)
(杜天保，沈国华，黄志球等. 通过代码模式改进基于IR的需求和代码之间追踪生成方法. 小型微型计算机系统, 2019, 40(5): 1107-1114)

[125] Wang Jin-Shui, Xue Xing-Si, Tang Zheng-Yi, et al. Recovering traceability links using named entity recognition. Application Research of Computers, 2016, 33(1): 132-135(in Chinese)
(王金水，薛醒思，唐郑熠等. 一种基于命名实体识别的需求跟踪方法. 计算机应用研究, 2016, 33(1): 132-135)

[126] Kuang H, Mäder P, Hu H, et al. Can method data dependencies support the assessment of traceability between requirements and source code?. Journal of Software: Evolution and Process, 2015, 27(11): 838-866

[127] Asghar M W, Marchetto A, Susi A, et al. Maintainability-based requirements prioritization by using artifacts traceability and code metrics//Proceedings of the 17th European Conference on Software Maintenance and Reengineering (CSMR). Genova, Italy, 2013: 417-420

[128] Zhao T, Cao Q, Sun Q. An improved approach to traceability recovery based on word embeddings//Proceedings of the 24th Asia-Pacific Software Engineering Conference(APSEC). Nanjing, China, 2017: 81-89

[129] Mahmoud A. An information theoretic approach for extracting and tracing non-functional requirements//Proceedings of the 23rd IEEE International Requirements Engineering Conference (RE). Ontario, Canada, 2015: 36-45

[130] Wang W, Niu N, Liu H, et al. Enhancing automated requirements traceability by resolving polysemy//Proceedings of the 26th IEEE International Requirements Engineering Conference(RE). Calgary, Canada, 2018: 40-51

[131] Niu N, Mahmoud A. Enhancing candidate link generation for requirements tracing: The cluster hypothesis revisited//Proceedings of the 20th IEEE International Requirements Engineering Conference(RE). Illinois, USA, 2012: 81-90

[132] Chen L, Wang D, Wang J, et al. Enhancing unsupervised requirements traceability with sequential semantics//Proceedings of the 26th Asia-Pacific Software Engineering Conference (APSEC). Putrajaya, Malaysia, 2019: 23-30

[133] Li Y, Schulze S, Saake G. Extracting features from requirements: Achieving accuracy and automation with neural networks//Proceedings of the 25th IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER). Campobasso, Italy, 2018: 477-481

[134] De Lucia A, Penta M D, Oliveto R, et al. Improving IR-based traceability recovery using smoothing filters//Proceedings of the 19th IEEE International Conference on Program Comprehension(ICPC). Ontario, Canada, 2011: 21-30

[135] Bella E E, Gervais M, Bendraou R, et al. Semi-supervised approach for recovering traceability links in complex systems //Proceedings of the 23rd International Conference on Engineering of Complex Computer Systems (ICECCS). Melbourne, Australia, 2018: 193-196

[136] Liu Y, Lin J, Zeng Q, et al. Towards semantically guided traceability//Proceedings of the 28th IEEE International Requirements Engineering Conference(RE). Zurich, Switzerland, 2020: 328-333

[137] Dietrich T, Cleland-Huang J, Shin Y. Learning effective query transformations for enhanced requirements trace retrieval//Proceedings of the 28th IEEE/ACM International Conference on Automated Software Engineering (ASE). California, USA, 2013: 586-591

[138] Liu Y, Lin J, Cleland-Huang J. Traceability support for multi-lingual software projects//Proceedings of the 17th International Conference on Mining Software Repositories (MSR). Seoul, Korea, 2020: 443-454

[139] Omoronyia I, Sindre G, Stålhane T. Exploring a Bayesian and linear approach to requirements traceability. Information and Software Technology, 2011, 53(8): 851-871

[140] de Jesus T O, dos Santos Soares M. A multi-criteria analysis of techniques and tools for tracing software requirements. IEEE Latin America Transactions, 2017, 15(5): 922-927

[141] Rochimah S, Wan-Kadir W M N, Abdullah A H. Utilizing multifaceted requirement traceability approach: A case study. International Journal on Software Engineering and Knowledge Engineering, 2011, 21(4): 571-603

[142] Mills C, Escobar-Avila J, Haiduc S. Automatic traceability maintenance via machine learning classification//Proceeding of the 34th International Conference on Software Maintenance and Evolution(ICSME). Madrid, Spain, 2018: 369-380

[143] Mäder P, Gotel O. Ready-to-use traceability on evolving projects. Software and Systems Traceability. London: Springer, 2012: 173-194

[144] Merilinna J, Yrjönen A, Räty T. NFR+framework method to support bi-directional traceability of non-functional requirements. Computer Science-Research and Development, 2015, 30(1): 35-49

[145] Wang Jin-Shui, Weng Wei, Peng Xin, et al. Recovering traceability links using syntactic analysis. Journal of Computer Research and Development, 2015, 52(3): 729-737(in Chinese)
(王金水，翁伟，彭鑫等. 一种基于句法分析的跟踪关系恢复方法. 计算机研究与发展，2015，52(3)：729-737)

[146] Antonino P O, Keuler T, Germann N, et al. A non-invasive approach to trace architecture design, requirements specification and agile artifacts//Proceedings of the 23rd Australian Software Engineering Conference (ASWEC). New South Wales, Australia, 2014: 220-229

[147] Bavota G, Colangelo L, De Lucia A, et al. TraceMe: Traceability management in eclipse//Proceedings of the 28th IEEE International Conference on Software Maintenance (ICSM). Trento, Italy, 2012: 642-645

[148] Sardinha A, Yu Y, Niu N, et al. EA-Tracer: Identifying traceability links between code aspects and early aspects//Proceedings of the 28th ACM Symposium on Applied Computing(SAC). Trento, Italy, 2012: 1035-1042

[149] Mayr-Dorn C, Vierhauser M, Keplinger F, et al. TimeTracer: A tool for back in time traceability replaying//Proceedings of the 42nd International Conference on Software Engineering (ICSE). Seoul, Korea, 2020: 33-36

[150] Turban B. PROVEtech: R2A—A tool for dedicated requirements traceability//Turban B ed. Tool-Based Requirement Traceability between Requirement and Design Artifacts. Wiesbaden, Germany: Springer Vieweg, 2013: 259-378

[151] Turban B. Tool-Based Requirement Traceability Between Requirement and Design Artifacts. Wiesbaden, Germany: Springer Vieweg, 2013

[152] Kamburjan E, Stromberg J. Tool support for validation of formal system models: interactive visualization and requirements traceability//Proceedings of the 5th Workshop on Formal Integrated Development Environment (F-IDE@FM). Porto, Portugal, 2019: 70-85

[153] Richter H, Gandhi R A, Liu L, et al. Incorporating multimedia source materials into a traceability framework//Proceedings of the 1st International Workshop on Multimedia Requirements Engineering (MERE). Minnesota, USA, 2006: 7-12

[154] Aung T W W, Huo H, Sui Y. A literature review of automatic traceability links recovery for software change impact analysis//Proceedings of the 28th International Conference on Program Comprehension(ICPC). Seoul, Korea, 2020: 14-24

[155] Peraldi-Frati M, Albinet A. Requirement traceability in safety critical systems//Proceedings of the 1st Workshop on Critical Automotive Applications: Robustness & Safety(CARS). Valencia, Spain, 2010: 11-14

[156] Torkar R, Gorschek T, Feldt R, et al. Requirements traceability: A systematic review and industry case study. International Journal on Software Engineering and Knowledge Engineering, 2012, 22(3): 385-434

[157] Santiago I, Jiménez Á, Vara J M, et al. Model-driven engineering as a new landscape for traceability management: A systematic literature review. Information and Software Technology, 2012, 54(12): 1340-1356

[158] Nair S, de la Vara J L, Sen S. A review of traceability research at the requirements engineering conference[re@21]//Proceedings of the 21st IEEE International Requirements Engineering Conference(RE). Rio de Janeiro(RJ), Brazil, 2013: 222-229

[159] Borg M, Runeson P, Ardö A. Recovering from a decade: A systematic mapping of information retrieval approaches to software traceability. Empirical Software Engineering, 2014, 19(6): 1565-1616

[160] Cleland-Huang J, Gotel O, Hayes J H, et al. Software traceability: Trends and future directions//Proceedings of the 36th Future of Software Engineering(FOSE). Hyderabad, India, 2014: 55-69

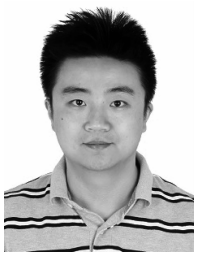
[161] Wang B, Peng R, Li Y, et al. Requirements traceability technologies and technology transfer decision support: A systematic review. Journal of Systems and Software, 2018, 146: 59-79

[162] Li Yin, Li Juan, Li Ming-Shu. Research on dynamic requirement traceability method and traces precision. Journal of Software, 2009, 20(2): 177-192(in Chinese)
(李引，李娟，李明树. 动态需求跟踪方法及跟踪精度问题研究. 软件学报，2009，20(2)：177-192)

[163] Zogaan W, Sharma P, Mirakhorli M, et al. Datasets from fifteen years of automated requirements traceability research: Current state, characteristics, and quality//Proceedings of the 25th IEEE International Requirements Engineering Conference(RE). Lisbon, Portugal, 2017: 110-121

[164] Hu Cheng-Hai, Peng Rong, Wang Bang-Chao. A survey of requirement tracking method based on information retrieval. Computer Applications and Software, 2017, 34(10): 20-28 (in Chinese)
(胡成海，彭蓉，王帮超. 基于信息检索的需求跟踪方法综述. 计算机应用与软件，2017，34(10)：20-28)

[165] Tufail H, Masood M F, Zeb B, et al. A systematic review of requirement traceability techniques and tools//Proceedings of the 2nd International Conference on System Reliability and Safety(ICSRS). Milan, Italy, 2017: 450-454



TAO Chuan-Qi, Ph. D., associate professor. His research interests include intelligent software development and testing.

ZHANG Meng, M. S. candidate. Her main research interest is requirements engineering.

GUO Hong-Jing, Ph. D. candidate. Her main research interest is intelligent software engineering.

HUANG Zhi-Qiu, Ph. D., professor. His research interests include software engineering, formal methods and cloud computing.

Background

Along with the rapid development of the modern software industry, the scale of software expands rapidly and the software system becomes more complex. This leads to a large number of software artifacts in the software development process, which contain a large amount of data information. How to effectively establish the traceability between software artifacts, especially the requirements traceability, has become a hot research topic in the field of software engineering, to assist other software development and maintenance tasks. Current research mainly analyzes the techniques commonly used in the establishment of requirements traceability. However, traceability between requirements and different software artifacts is not analyzed in detail. As a result, the types of requirements traceability obtained are singularized, and software development integrity is poor.

In this paper, we study the traceability between requirements and different software artifacts using a methodology of systematic literature review. And this paper applies requirements traceability to real intelligent software development projects. In total, we selected 135 studies over the last 10 years. On the one hand, according to the text content, we classify different software artifacts in the software development life cycle, including document-level software artifacts, code-level software artifacts, and product-level software artifacts. We mainly studied the traceability between requirements and document-level software artifacts, requirements and code. Meanwhile, we organized the experimental data sets, the tools under development, and the application scenarios. On the other hand, through the application in actual intelligent software development projects, we have verified the usefulness of requirements traceability technology in improving the efficiency of intelligent software development. In addition, we analyze the future research trends and look forward to the future.

This work is supported by the JW Pre-research Project of Equipment Development Department (Public) (31511110204), the National Key Research & Development Program of China under Grant No. 2018YFB1003900, the National Natural Science Foundation of China under Grant Nos. 61602267, 61402229. Through the research of the latest artificial intelligence technology, the team has improved the organic integration of military software requirements, design, implementation, and testing, as well as the degree of automation and intelligence. And they have published several research articles about requirement modeling, code recommendation, redundant code detection, and intelligent testing. This paper aims to provide the theoretical basis for traceability between requirements and different software artifacts. Meanwhile, we establish the traceability to assist with completing the task of code recommendation and base code intelligence library generation, to improve software development integrity and development efficiency.