软件缺陷标题质量的实证研究

续永, 孙龙飞, 张汤浩然, 毛新军

引用本文

续永, 孙龙飞, 张汤浩然, 毛新军. 软件缺陷标题质量的实证研究[J]. 计算机科学, 2023, 50(12): 14-23.

XU Yong, SUN Longfei, ZHANGTANG Haoran, MAO Xinjun. Examining the Quality of Bug Report Titles:An Empirical Study [J]. Computer Science, 2023, 50(12): 14-23.

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.230300211

软件缺陷标题质量的实证研究

续 永 孙龙飞 张汤浩然 毛新军
国防科技大学计算机学院 长沙 410073
湖南省复杂系统软件工程重点实验室 长沙 410073
(xuyong20a@nudt.edu.cn)

摘 要 软件缺陷标题用简洁的语言描述了软件缺陷的关键信息,有助于软件开发者快速地掌握软件缺陷的梗概,进而高效地开展软件缺陷管理工作。当前诸多软件开发实践中可以发现,软件缺陷标题的质量参差不齐,存在冗长、晦涩、缺乏对关键信息的描述等问题,导致难以阅读和理解,影响了软件缺陷管理的效率和质量。因此有必要深入探究影响软件缺陷标题质量的具体因素以及当前软件缺陷标题的质量情况。文中围绕这两个方面的问题开展了定性和定量相结合的实证研究,选取190个在线文档进行定性分析以获取开发者对缺陷标题质量的需求,基于分析结果采取GQM范式构建了缺陷标题质量度量模型,并以此对GitHub中5个开源项目的1804个软件缺陷标题进行质量问题普遍性分析。研究结果表明:1)开发者主要关注软件缺陷标题4个方面的质量需求,即简洁(110,58%)、清楚(65,34%)、提供期望信息(157,83%)和提供具体描述(67,35%);2)70%的软件缺陷标题存在不同程度的质量问题。缺乏期望信息和描述不具体是最常见的两类质量问题,42%的软件缺陷标题缺乏期望信息,24%的软件缺陷标题需要补充具体描述。文中的研究发现有助于指导报告者提交高质量的软件缺陷标题。

关键词:缺陷标题;度量模型;质量问题分布;主题分析;软件缺陷管理
中图法分类号 TP311

Examining the Quality of Bug Report Titles:An Empirical Study

XU Yong,SUN Longfei,ZHANGTANG Haoran and MAO Xinjun
College of Computer Science and Technology,National University of Defense Technology,Changsha 410073,China
Hunan Key Laboratory of Software Engineering for Complex Systems,Changsha 410073,China

Abstract The title of a software bug serves as a concise summary of the bug,which can help developers swiftly comprehend the bug and facilitate effective software bug management. Current software development practices reveal that the quality of bug titles varies considerably,characterized by issues such as verbosity,obscurity,and a lack of crucial information,ultimately impacting the efficiency of software bug management. To this end,this study attempt to understand which factors influence the quality of bug title and identify the current quality status. We examine 190 online documents to elicit the quality requirements of developers for bug title,construct a bug title quality measurement model using the GQM paradigm,and analyze the prevalence of quality issues in 1 804 bug titles from five open-source projects on GitHub. The findings indicate that developers primarily focus on four aspects of quality for bug titles,i.e. conciseness(110,58%),clarity(65,34%),description of the core idea of the bug(157,83%),and accurate descriptions(67,35%). Approximately 70% of bug titles exhibit varying degrees of quality issues,with a lack of crucial information and inaccurate descriptions being the most common issues. Specifically,42% of bug titles lack information expected by developers,and 24% require being rewritten accurately. The study can offer guidance for reporters seeking to submit high-quality bug titles.

Keywords Title of bug report,Measurement model,Distribution of quality issues,Theme analysis,Software bug management

1 引言

浏览软件缺陷报告列表是开发者在进行软件维护过程中的一项常见活动。而软件缺陷报告的标题是开发者在浏览过程中可以将不同软件缺陷报告进行有效区分的重要信息来源。作为开发者在浏览缺陷报告列表时的主要信息来源,一个高质量的缺陷标题有助于开发者快速地掌握软件缺陷的梗概,进而高效地开展软件缺陷管理工作。相反,一个表达不明或缺乏有效信息的缺陷标题往往会迫使开发者花费额外的时间来阅读报告正文中的细节描述,以了解缺陷概况。如果软件缺陷报告列表中充斥着这样的缺陷标题,那么快速对这些缺陷报告进行甄别、理解、分类、排序就会变得十分困难。

到稿日期:2023-03-28 返修日期:2023-07-10
基金项目:国家自然科学基金(62172426);科技部重点研发计划(2018YFB1004202)
This work was supported by the National Natural Science Foundation of China(62172426) and Key Research Project of the Ministry of Science and Technology(2018YFB1004202).
通信作者:毛新军(xjmao@nudt.edu.cn)

例如，Android studio 团队在他们的报告指南中提到“你会惊讶于有多少缺陷标题被设置为‘Bug’‘Issue’‘Exception’或‘not working’等，这让我们很难对这些缺陷报告进行排序”[1)]。然而，报告者提交的缺陷标题的质量往往参差不齐，许多缺陷标题冗长、晦涩、缺乏对关键信息的描述，这会影响软件缺陷管理的效率和质量。为了促使报告者提供高质量的缺陷标题，有必要了解哪些因素会影响缺陷标题的质量以及需要着重关注改进缺陷标题在哪方面的质量。

先前的部分研究工作侧重于自动生成高质量的缺陷标题[1]。例如，Chen 等[1]设计了缺陷标题生成工具，该工具基于一些简单的句法规则来度量缺陷标题质量和构建可靠数据集。在他们的工作中，语义层面的质量影响因素并没有被考虑。还有部分研究工作侧重于通过分析缺陷标题来研究相关的现状与特点[2-3]。例如，Ko 等[2]分析了近 20 万条缺陷标题来探究报告者描述缺陷标题的话语模式。整体而言，关于缺陷标题质量的理解和认识仍有所欠缺。

为了加深对缺陷标题质量的理解，本文提出了以下研究问题：

RQ1　从开发者的角度，缺陷标题具体应该满足哪些方面的质量要求？

RQ1 旨在探索缺陷标题质量有哪些影响因素，这可以帮助区分不同质量水平的缺陷标题和识别缺陷标题存在哪些质量问题，以及指导报告者改进缺陷标题。此外，了解缺陷标题质量的要求有助于揭示一些研究方向。

RQ2　开源社区中，缺陷标题在多大程度上存在质量问题？

RQ2 旨在建立起对缺陷标题质量分布现状的基本认识，以揭示在现实场景中缺陷标题的各类质量问题分别有多严峻。

本文首先采用 Google 搜索引擎搜集了 190 个与缺陷标题质量相关的在线文档，并对其使用主题分析[4]，得到了 10 类质量要求，其中开发者主要关注 4 个方面的质量要求：简洁（110，57.89%）、清楚（65，34.21%）、提供期望信息（157，82.63%）和提供具体描述（67，35.26%）。然后，本文采用 GQM 范式（Goal-Question-Metric paradigm，目标-提问-指标范式）[5]构建了针对缺陷标题质量的度量模型，并基于此对 5 个开源项目的缺陷标题的质量进行人工度量和定量分析。分析结果表明，缺陷标题的质量问题分布具有普遍性，平均有 70.07% 的缺陷标题存在质量问题，但是 62.86% 的缺陷标题只存在 1～2 个质量问题。此外，缺乏期望信息和描述不具体是最常见的两类质量问题，平均有 41.96% 的缺陷标题缺乏期望信息，平均有 23.84% 的缺陷标题需要补充具体描述。

本文的主要研究贡献如下：

1）总结了开发者对缺陷标题的 10 类质量要求，可帮助区分不同质量水平的缺陷标题和指导报告者拟写高质量的缺陷标题；

2）提出了评估缺陷标题质量的度量模型，可帮助定量分析缺陷标题存在的质量问题并指导报告者改进缺陷标题；

3）发现缺陷标题的质量问题是普遍存在的，其中缺乏期望信息和描述不具体是最常见的两类质量问题，需要开发者及研究人员重点关注和改进。

本文第 2 章介绍了整个研究工作的方法设计；第 3 章展示了针对两个研究问题的研究结果；第 4 章讨论了研究结果给开发者和研究者带来的启发；第 5 章梳理了该研究所面临的有效性威胁；第 6 章梳理了相关研究工作，讨论了已有研究工作的优势和不足，并描述了本文工作的亮点；最后总结全文。

2　方法设计

本文的研究方法如图 1 所示。首先通过对 190 个与缺陷标题质量相关的在线文档做主题分析[4]来了解开发者对缺陷标题的质量要求，然后采取 GQM 范式[5]来构建针对缺陷标题质量的度量模型，最后使用该度量模型对来自开源社区中 5 个开源项目的 1 804 个缺陷标题进行人工度量和定量分析。相关数据和模型可在文献[6]中找到。

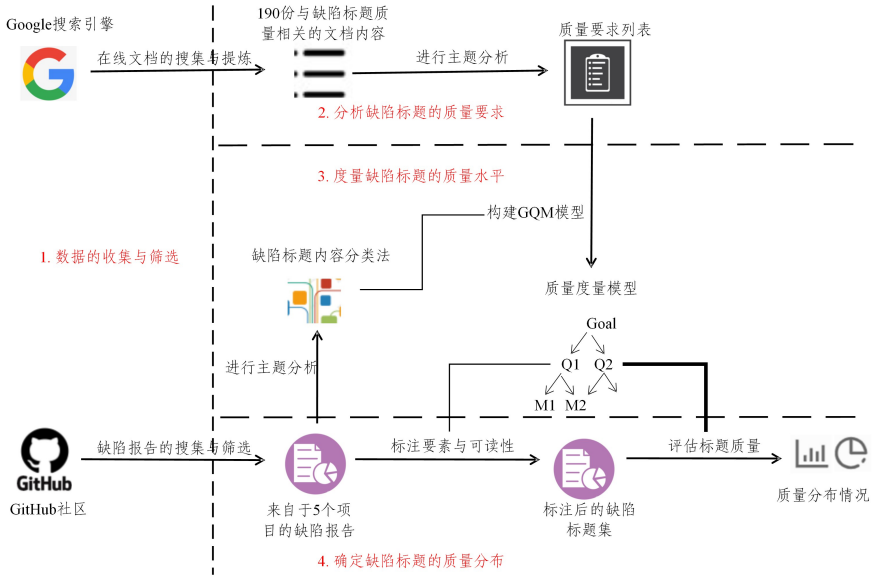

图 1　研究方法框架

Fig. 1　Framework of research approach

1) https://developer.android.com/studio/report-bugs

2.1 数据的收集与筛选

1)在线文档的搜集与提炼。鉴于先前的研究者们使用谷歌搜索引擎来获取实践现状且取得了不少有价值的发现[7-8],本文采取在谷歌搜索引擎上使用关键词检索的方法来搜集包含有关缺陷标题质量的开发者观点的在线文档,并对其进行提炼。

(1)检索策略。本文在谷歌搜索引擎上使用一组关键词进行检索,并对按相关性排序的前 200 个检索结果进行审核筛选。在这个过程中,本文会递归审核在线文档中的嵌入链接以扩展检索范围,该过程将在所有嵌入链接都不符合审核要求或当前在线文档不存在嵌入链接时停止。受分析能力的限制,只能搜集尽可能多的在线文档用于分析,而考虑到相关性排名在 200 名以后的检索结果很少符合审核标准,这意味着继续往后搜集在线文档需要花费相当大的精力却只能得到很有限的收益,因此此处将审核和筛选范围限制在前 200 个检索结果。

(2)检索关键词。为了检索有关缺陷标题质量的在线文档,本文首先审核和筛选了关键词"good bug report"和"good bug report title"下的检索结果,发现有不少在线文档采用"summary"而不是"title"来指代缺陷标题。为此,本文进一步补充了"good bug report summary"来扩大检索范围。

(3)文档审核与筛选。考虑到本文的研究目的是调查开发者对缺陷标题的质量要求,本文只保留包含缺陷标题质量要求相关论述的在线文档。此外,为了尽可能确保搜集到的在线文档可以真实地反映开发者的观点,本文只保留属于论坛讨论、个人技术博客、在线书籍和项目贡献指南这 4 个类别的在线文档,这 4 类文档常被用于发表个人观点。

(4)文档提炼。最终,有 190 份在线文档被筛选出来,搜集截止日期为 2022-12-24。这些在线文档的详情见文献[6]。考虑到这些文档只有部分内容与本文研究相关,因此有必要做进一步提炼。具体地,本文对所有在线文档的内容进行筛选,只留下用于表达对缺陷标题质量的期望以及对拟写高质量标题的建议的句子,这些句子反映了开发者对缺陷标题质量的要求。如果有几个句子在语义上是连贯的,那就将它们合并为一个复合句子。最终有 444 个句子被搜集到,平均每个文档有 2.34 个句子符合我们的要求。将这些句子作为提取开发者质量要求的数据来源。此外,开发者在文档中用来区分质量不同的缺陷标题的例子也被提炼出来,其可以帮助验证和完善 2.3 节提出的质量度量模型。

2)缺陷报告的搜集与筛选。为了调查缺陷标题质量的分布,考虑到 GitHub 平台上托管了大量采用问题追踪系统来管理软件缺陷的开源项目,本文选取 GitHub 平台作为调查项目来源,并确定了 5 个面向不同领域、基于不同语言、来自不同社区的大型开源项目。这些项目的基本信息如表 1 所列,相关统计数据截止到 2022-12-17。选择这些项目的理由如下:(1)这些项目关联着不同规模和不同知识背景的报告者和开发人员群体,这意味着这些项目的缺陷标题在表达和内容上存在差异,而从这些项目中抽取缺陷标题进行分析得出结论具备一定的泛化性;(2)这些项目的 star 数均超过 1000,且在近 3 个月内持续接收到问题报告,这代表它们是足够流行且足够活跃的,本文假设这样的项目会有尽可能多的高质量缺陷标题。本文从这些项目中随机采样了 1804 个问题报告(置信度 95%,误差范围 5%),平均每个项目 360.8 个问题报告。对于抽样得到的问题报告,本文通过手动检查问题类型、正文和评论来排除不是缺陷报告的问题报告(如功能请求),因为本文只关注软件缺陷报告的标题。同时,我们通过自动化检查排除了标题或正文存在非英文表述的问题报告。对于这些被排除掉的问题报告,本文将重新随机采样相同数目的问题报告并重复上述检查过程,直到没有新的问题报告被排除为止。

表 1 5 个开源项目的基本信息(截止到 2022-12-17)

Table 1 Basic information of five open source projects(until 2022-12-17)

项目名称	面向领域	开发语言	所属社区	最近 3 个月的问题报告提交个数	问题报告总数
moby	容器技术	go	docker	9,30,57	21652
moveit	运动规划	c++,python	ros	32,13,12	1381
pandas	数据分析	python	pandas	74,84,113	23590
ansible	软件维护	python	redhat	10,13,45	31013
WordPress-Android	网络服务	kotlin,java	wordpress	37,25,38	7409

2.2 分析缺陷标题的质量要求

本节采用主题分析方法[4]来确定开发者对缺陷标题的质量要求,具体步骤如下:1)仔细阅读 2.1 节中提炼得到的每个句子,以了解开发者表达了何种质量要求;2)基于阅读所得见解,为每个句子分配一些简短的描述性的编码来表征该句子所表达的质量要求类别;3)在完成初始编码后,将有相似含义的编码归类到同一初始主题下;4)回顾各个初始主题,尝试将相似的初始主题合并为一个新主题,或将一些初始主题作为其他主题的子主题;5)最后,得到最终的主题集。为了限制主观性,本节中的所有手动分析,皆由前两位作者共同参与完成。在创建编码和主题构建的过程中,如果两人意见不一致,那么将请第三位作者作为仲裁者来主持会议以讨论和解决冲突,如果会议中两人仍然无法达成一致,则由仲裁者决定最终结果,该过程一直持续到两人在所有句子上都达成一致为止。

2.3 度量缺陷标题的质量水平

考虑到已有的缺陷标题质量的度量方法[1]缺乏对语义层面质量的衡量,在调查缺陷标题质量分布情况之前,基于 2.2 节所得质量要求分类,本节确定了缺陷标题质量的度量模型和度量标准。

1)缺陷标题内容的分类。为了度量开发者在缺陷标题内容方面的质量要求,有必要了解报告者会在缺陷标题中具体传达哪些类别的信息。本文使用术语"表达类别"来代指缺陷标题所传达的信息的类别,使用术语"表达要素"来代指缺陷

标题中用于传达某类信息的部分。对于一个缺陷标题而言，其可以属于多个表达类别和包含多个表达要素。考虑到开发者对缺陷标题内容的见解往往和他们对缺陷标题内容方面的质量要求紧密关联，而这些见解有助于帮助刻画内容方面的质量要求，本文首先通过对 2.1 节搜集的所有句子进行主题分析[4]来了解开发者对内容的一些看法，并从中提炼出一组初始编码。然后，使用这些初始编码对在线文档中的某些标题实例进行试点编码和组织完善，并形成初始编码本。在得到初始编码本后，从随机抽取得到的 1 804 个软件缺陷报告中选取 400 个缺陷标题进行主题分析[4]，具体过程与 2.2 节一致，以确定具体存在哪些表达类别。这里选择 400 个缺陷标题用于表达类别提取，原因是更多的缺陷标题也不能提供新的见解。事实上，在分析完 250 个左右的缺陷标题后就不再出现新的编码，这意味着已经达到了饱和[9]。表 2 列出了缺陷标题内容的分类情况，其中圆括号表示主题或表达类别的出现次数与频率，大括号表示标题示例所属软件缺陷报告在编码结果列表中的序号，斜体文字代表当前表达类别对应的表达要素。具体的编码结果和编码准则可以在文献[6]中找到。

表 2　缺陷标题内容分类结果

Table 2　Taxonomy of content of bug titles

主题	表达类别	类别解释	表达要素示例
缺陷效应（306，35.53%）	软件失效（173，20.10%）	可以直接观察到的异常软件行为，其代表的是缺陷效应的直观外在表现	“*pandas. io. data doesn't work with* hyphenated ticker names”{52}
	软件故障（54，6.27%）	报告者通过软件调试所发现的异常软件状态，其代表的是缺陷效应的隐含内在表现	“*Deadlock* on getting swarm info”{80}
	操作失败（61，7.08%）	对用户失败操作的描述	“*Unable to TFTP IOS Image to Cisco ASR router*”{5}
	非法操作（3，0.34%）	对不合法但是却执行成功的用户操作的描述	“*User can delete Homepage and break their site*”{70}
	期望结果（15，1.74%）	报告者期望发生的情况	“*Reader should check attachments* for suitable featured image”{134}
缺陷上下文（421，48.90%）	缺陷方位（173，20.10%）	缺陷的大致位置，其可以进一步被细分为代码行、函数、文件、组件和软件特征	“memory leak *in libmoveit_move_group_default_capabilities*”{180}
	缺陷触发前的操作（90，10.45%）	缺陷被触发前报告者对软件进行的操作	“*Clicking on a commenting user* does not open a site when it's missing ”{128]
	缺陷触发时的状况（158，18.35%）	对缺陷触发时任意可能与缺陷相关的客观事实的描述，如软件版本、软件配置情况等	“At theend of the login flow，“Continue” button is not working *when “Don't keep activities” is enabled*”{172}
报告者想法（102，11.85%）	缺陷解释（70，8.13%）	报告者对缺陷的理解，即报告者对缺陷是什么或缺陷为什么会被触发的回答	“Build problem *due to linker problems* with ‘shapes:: createMeshFromVertices’ and ‘ros:: NodeHandle:: NodeHandle’. ”{3}
	修复提议（16，1.86%）	报告者针对缺陷修复所给出的建议，即报告者对如何修复缺陷的回答	“*allow omitted spaces* for nested loop dictionaries”{19}
	缺陷分类（15，1.74%）	报告者对缺陷类别的推测，即报告者对哪一类缺陷被触发了的回答	“*Build problem* on OSX”{367}
	效应推论（1，0.12%）	报告者对未观测到但可能发生的异常软件行为或异常软件状态的推测，即报告者对缺陷会导致哪些异常情况的回答	“netscaler_nitro_request. py mandates resource name in all instances，*breaking some functions*”{104}
缺陷无关内容（32，3.72%）	问题报告间关系（1，0.12%）	当前缺陷报告与其他问题报告之间的关系，此处的问题报告不局限于缺陷报告，也可以是特征申请、方案讨论等其他类别的问题报告	“*Same issue as* 20221 but not resolved”{4}
	非描述性短语（6，0.70%）	非描述性的一些请求、提问等	“The link still not work in my environment，*need help*”{4}
	不可读文本（18，2.09%）	含义难以理解的文本	“Kernel Panic - Centos 7. 2 3. 10. 0 *-with devicemapper LVM thinpool ext4*”{34}
	无法归类文本（7，0.81%）	无法被归类到之前任何一个表达类别中的文本	“*numpy vs pandas*：different estimation of covariance in presence of nan values”{73}

2）GQM 质量度量模型的构建。本文采用 GQM 范式[5]来构造质量度量模型，这是一种面向目标和基于提问的通用质量度量方法，其构建出的质量度量模型通常包含 3 个层次，分别为目标层、提问层和指标层。构造过程使用了如下定义。

表达要素集：缺陷标题中所有表达要素组成的集合。

有效要素：归于缺陷效应、缺陷上下文、报告者想法的表达要素。这些表达要素可以在一定程度上帮助开发者进行软件缺陷管理。

冗余要素：归于缺陷无关内容的表达要素。

泛化词：不传达任何有助于软件缺陷管理的信息的单词。

实义词：与泛化词相对。

待改进要素：内含实义词过少，导致无法传达丰富信息量的表达要素。

未阐述要素：在正文中没有被进一步阐述的表达要素。

信息期望：开发者对缺陷标题所含信息的普遍期望。考虑到后文 3.1 节中对开发者信息期望的分析，本文将该信息期望定义为“表达要素集中至少含有一个归于缺陷效应的表达要素和至少含有一个归于缺陷上下文的表达要素”。

本文将开发者对缺陷标题的不同质量要求总结归纳为缺陷标题的可用性[1)]，并将其设置为度量目标。考虑到度量难度

1) 可用性可以通过如下提问来刻画——“缺陷标题能否帮助开发者轻易快速地获取到可以被用于一系列缺陷管理任务的期望信息”。

和必要性，本文不为“易于检索”“使用标签文本”“符合项目风格”“突出问题重要性”等质量要求设置度量指标。具体质量度量模型如表3所列。此外，考虑到NELI指标的计算，在每个表达类别都设置了一个实义词数下限值，具体如表4所列。

表3 缺陷标题的GQM质量度量模型

Table 3 GQM model of bug titles

目标层	提问层	指标层	指标含义	指标要求	对应质量要求
从开发人员的角度来度量缺陷标题的可用性	缺陷标题的表达如何？	单词数 *NW*	缺陷标题所含单词的个数	*NW*<16	简洁
		可读性级别 *RL*	可取值-1,0,1,其中1代表易读,0代表偏难,-1代表难读[1)]	*RL*=1	清楚
	缺陷标题的内容如何？	最少待补充要素数 *NELA*	为了满足信息期望，最少需要增加有效要素的个数	NELA=0	提供期望信息
		最少待改进要素数 *NELI*	表达要素集中实义词数低于相应下限值的表达要素的总数，这些表达要素有待进一步改进	NELI=0	提供具体描述
		冗余要素数 *NRE*	标题中冗余要素的数目	NRE=0	避免冗余描述
		阐述率 *ER*	缺陷标题的要素集中在正文中得到进一步阐述的要素的占比	ER=1	具有概括性

表4 各表达类别对实义词数的下限要求

Table 4 Minimum number of meaningful words required by each expression category

表达类别	软件失效	软件故障	操作失败	非法操作	期望结果	缺陷方位	缺陷触发前的操作	缺陷触发时的状况	缺陷解释	修复提议	缺陷分类	效应推论
实义词数下限要求	2	2	2	3	2	1	1	2	2	2	1	2

3)质量度量标准的确定。为了从整体上反映缺陷标题质量水平，基于所得GQM度量模型，本节设置了一个质量度量标准，其将缺陷标题的质量分为3个等级，分别为好(表达或内容俱佳)、较好(表达上或内容上存在些许不足)、差(表达上或内容上存在很多不足)。该质量分级通过指标质量问题个数(Number of Quality Issues，NQI)来实现，其含义为缺陷标题的所有度量指标中不满足相应指标要求的度量指标的总数。NQI取值为0时代表“好”，取值为1或2代表“较好”，为其余取值时代表“差”。

2.4 确定缺陷标题的质量分布

为了了解不满足开发者质量要求的缺陷标题在多大程度上存在，本文首先对2.1节中搜集到的1804个缺陷标题分3个维度进行了标注，这3个维度分别为表达要素、可读性级别、未阐述要素。更具体地讲，本文对其中的表达要素进行实体标注，对其可读性级别进行类别标注，对在正文中没有被进一步阐述的表达要素进行类别标注(将其标注为未阐述类别)。在标注过程中，如果报告者在报告缺陷后的一小时内又提供了额外的补充评论，那么这些补充评论将被视为正文的一部分。如果表达要素并非在正文的文本部分被阐述，而是在相关图片、视频或超链接文档中被阐述，则该阐述关系被认可。以上过程将分5轮来实施，且分别由前两位作者独立完成(总共1804个缺陷报告，去除掉2.3节中用于发现表达类别的400个缺陷报告，还剩下1404个缺陷报告。在前4轮中每人分别标注300个缺陷报告，最后一轮每人分别标注204个缺陷报告)。在每轮标注完缺陷标题后，本文将对标注人员之间的一致性水平进行分析，并举行会议来解决出现的分歧。如果会议中两位标注人员未能就标注结果达成一致，则由第三位作者担任仲裁者。每轮标注结束后的一致性情况和分歧情况具体如表5所列，其中F1指标[10]用于衡量表达要素维度和未阐述要素维度的一致性水平，cohen's kappa系数[11]用于衡量可读性级别维度的一致性水平。

表5 人工标注过程统计数据

Table 5 Statistics of manually labelling process

轮次	表达要素维度下的F1值	可读性维度下的kappa值	未阐述要素维度下的F1值	不一致样本总数
1	0.74	0.88	0.86	172
2	0.83	0.91	0.82	132
3	0.70	0.90	0.80	173
4	0.69	0.95	0.80	167
5	0.70	0.85	0.85	112

在解决完所有分歧后，本文根据2.3节所提出的质量度量模型对来自5个项目的1804个缺陷标题进行质量度量和统计分析。具体来说，本文首先计算了每个缺陷标题在每个质量指标上的取值，然后统计并分析了5个项目缺陷标题的NQI指标分布，以及不合格质量指标的分布。

3 研究发现

本章将介绍本文实验结果，这些实验结果可以解决引言中提出的两个研究问题。3.1节展示了开发者对缺陷标题有哪些质量要求。3.2节展示了开源社区中缺陷标题在多大程度上存在质量问题。

3.1 RQ1的研究发现

本文最终确定了开发者对缺陷标题的10个质量要求，并将其归于2个维度。图2展示了这一发现，以及它们在190份在线文档中的出现次数和占比(在括号中表示)。下文对其进行了详细说明。

1) 易读指无需正文辅助便可理解标题含义；偏难指必须在阅读完正文中标题相关部分后才能够理解标题含义；难读指即使在阅读完正文中标题相关部分后还是无法理解标题含义或者是只能部分理解标题含义。

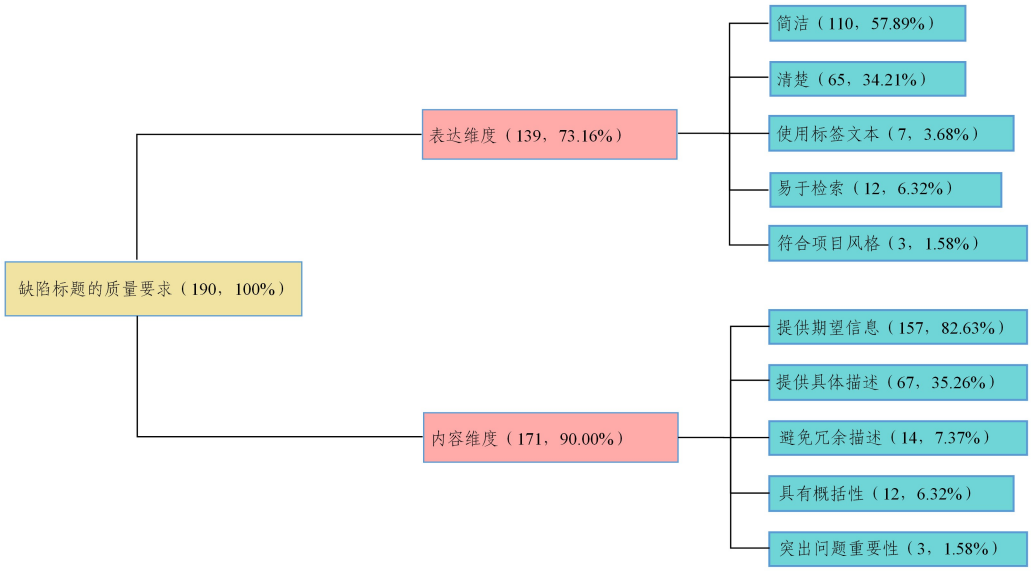

图 2　缺陷标题质量要求的分类结果

Fig. 2　Taxonomy of quality requirements of bug titles

1)表达维度(139,73.16%)。表达良好的缺陷标题往往可以快速被开发者定位、理解和消化，这可以有效提高软件缺陷管理的效率。表达维度包含以下 5 个质量要求：

(1)简洁(110,57.89%)。缺陷标题应当简短。具体地，一个缺陷标题不宜包含过多的字符或单词。过多的字符可能会导致缺陷标题在浏览列表中被截取掉，从而影响开发者的浏览体验[1)]。而过多的单词数会让开发者无法在短时间内消化标题的含义，并且很可能会产生冗余信息[2)]。

(2)清楚(65,34.21%)。为了让开发者在浏览时准确了解标题所表达的含义，缺陷标题需要易于理解[3)]。针对该质量要求，一些开发者建议在拟写缺陷标题时使用恰当的语法、符号、拼写以及确保句式的完整[4)]。

(3)使用标签文本(12,6.32%)。缺陷标题被期望在标题首部使用“[tag]”“tag:”等格式的标签文本，这有利于开发者进行软件缺陷报告的排序[5)]。

(4)易于检索(7,3.68%)。缺陷标题应易于检索，这可以帮助开发者快速定位相关软件缺陷报告，并且减少重复缺陷报告出现的概率，从而提高软件缺陷管理的效率。针对该质量要求，一些开发者建议报告者使用开发者常用的检索词来构造缺陷标题[6)]。

(5)符合项目风格(3,1.58%)。缺陷标题应该采用项目开发团队所熟悉的标题结构[7)]。一些开发者建议报告者严格按照项目规定来拟写缺陷标题[8)]。

2)内容维度(171,90.00%)。内容丰富的缺陷标题可以为开发者提供可以被用于一系列缺陷管理任务的期望信息。内容维度包含以下 5 个质量要求：

(1)提供期望信息(157,82.63%)。报告者需要在缺陷标题中提供开发者期望其提供的缺陷信息，否则开发者将被迫阅读报告正文来获取这些信息，这会加重开发者的阅读负担并降低软件缺陷管理效率。为了进一步了解开发者有哪些信息，前两名作者对与“提供期望信息”相关的句子进一步按 2.2 节中的步骤进行定性分析，旨在了解开发者有哪些常见信息期望。分析发现，开发者最常见的信息期望有“发生了什么”“何时发生”“何处发生”，表达这些信息期望的在线文档在“提供期望信息”相关在线文档中分别占比 56.05%，18.47%，28.66%。

(2)提供具体描述(67,35.26%)。缺陷标题需要准确具体地描述软件缺陷，而不应该进行泛化的描述。对软件缺陷的具体描述可以对当前软件缺陷报告和其他软件缺陷报告之间进行有效区分，同时也可以使开发者掌握足够多的缺陷细节信息。针对该质量要求，一些开发者建议报告者避免使用“not work”或“broken”之类的泛化表述[9)]。

(3)避免冗余描述(14,7.37%)。缺陷标题应避免提供无用或冗余的描述，这有助于开发者将注意力集中到可以提供有效帮助的信息上，从而更高效地进行软件缺陷管理。

(4)具有概括性(12,6.32%)。缺陷标题需要能够概括报

1) https://www.applause.com/blog/write-an-impactful-bug-report

2) https://testerwork.com/how-to-write-legendary-bug-reports/

3) https://zipboard.co/blog/bug-tracking/all-about-bug-reports/

4) https://qa.world/writing-good-bug-report-tips-tricks/

5) https://community.eveonline.com/support/test-servers/bug-reporting/

6) https://www.hikeqa.com/mobile-app-testing-services/how-to-write-a-bug-report-for-android-app-testing-services/

7) https://www.testmonitor.com/blog/how-to-write-a-bug-report-that-solves-issues-effectively

8) https://lucasdargis.com/5-ways-to-improve-your-bug-titles/

9) https://softwaretesting.news/write-effective-bug-report/

告正文，这一方面需要其能够概述报告正文中描述的缺陷情况，另一方面其包含的信息都在正文中得以进一步阐述。

(5)突出问题重要性(3，1.58%)。缺陷标题需要着重突出缺陷的严重程度，这有助于开发者进行软件缺陷报告的排序。

综上，本节确定了开发者对缺陷标题的 10 个质量要求，其中开发者主要关注 4 个方面的质量需求：简洁(110，57.89%)、清楚(65，34.21%)、提供期望信息(157，82.63%)和提供具体描述(67，35.26%)。

3.2 RQ2 的研究发现

5 个项目缺陷标题的 NQI 指标分布如图 3 所示。该分布结果表明，平均有 29.93%的缺陷标题的质量等级属于“好”，62.86%的缺陷标题的质量等级属于“较好”，而质量等级属于“差”的缺陷标题比较罕见，只占 7.21%。这一方面说明缺陷标题的质量水平普遍较高，另一方面也说明缺陷标题的质量问题是普遍存在的。为了了解不同质量问题的普遍性，本文进一步对不同质量维度下不合格质量指标的分布进行了调查。

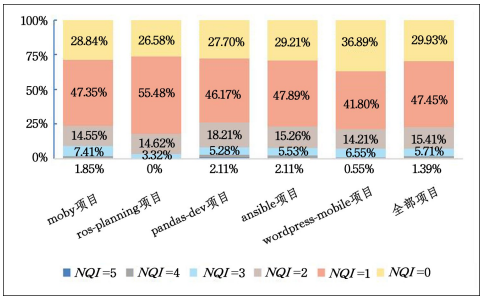

图 3 缺陷标题的 NQI 指标分布

Fig. 3 Distribution of NQI metrics of bug titles

表达维度下不合格质量指标的分布如图 4 所示。有 3.43%～6.08%的缺陷标题在 NW 指标上是不合格的，而在浏览这些缺陷标题时，开发者需要花费较长的时间来理解这些缺陷标题的含义，这会降低开发者处理缺陷报告的速度。该统计结果表明有一部分缺陷标题用了过长的篇幅来描述缺陷，但是这种现象并不常见。此外，有 6.03%～10.05%的缺陷标题在 RL 指标上是不合格的，但这种现象并不明显。这部分难以被理解或者是不可被理解的缺陷标题会阻碍开发者快速理解和消化其中的内容，从而影响开发者管理缺陷的效率。本文对在 RL 指标上不合格的 151 个缺陷标题进行了定性分析，并且总结归纳出了产生不易读缺陷标题的一些原因，具体如下：

1)省略了关键连词或关键实义词。一些不易读缺陷标题倾向于省略某个正文部分的一些关键连词或关键实义词，这直接导致缺陷标题信息不完整或缺陷标题结构不通顺，并最终降低了缺陷标题的可读性。例如，缺陷标题“Kernel Panic-Centos 7.2 3.10.0-with devicemapper LVM thinpool ext4”[1)]的后半部分“with devicemapper LVM thinpool ext4”让人难以理解。在阅读正文之后发现，进一步阐述该部分的正文部分为“We are leveraging devicemapper with a backing LVM thinpool using the ext4 filesystem”。通过对比可知，报告者省略了“leveraging”“with”“using”之类的连词及实义词。

2)引用文本信息但是省略了引号。一些不易读缺陷标题在引用报错信息或软件内容之类的文本信息时省略了引号，这给读者分析标题结构并正确理解标题内容带来了困难，因为读者无法及时把引用部分和其他部分区分开。例如，缺陷标题“Stack deploy failed with rejected status endpoint create on GW Network failed: endpoint with name gateway_stack_name already exists in network docker_gwbridge”[2)]引用了报错信息“endpoint create on GW Network failed: endpoint with name gateway_stack_name already exists in network docker_gwbridge”，但是它省略了引号，这会导致读者在阅读时自然地将这一部分和前面的自然语言描述连贯起来理解，而这种对标题结构的错误分析将阻碍读者对标题内容的正确理解。

3)堆砌关键内容而忽略了系统组织。一些不易读缺陷标题倾向于把不同正文部分的重要内容堆砌在一起而不考虑对这些内容的系统组织，这会带来语法混乱的问题并影响缺陷标题的可读性。例如，“dynamics_solver symbol lookup error: undefined KDL symbol”[3)]便是简单地把组件“dynamics_solver”和报错信息“symbol lookup error: undefined KDL symbol”堆砌在一起。

值得一提的是，上述原因可能同时出现在一个缺陷标题上。例如，“dynamics_solver symbol lookup error: undefined KDL symbol”便同时满足“引用文本信息但是省略了引号”和“堆砌关键内容而忽略了系统组织”。

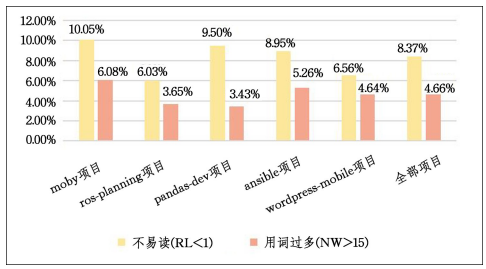

图 4 表达维度下不合格质量指标的分布

Fig. 4 Distribution of unqualified indicators of bug titles in expression dimension

内容维度下不合格质量指标的分布如图 5 所示。平均有 41.96%的缺陷标题在 NELA 指标上是不合格的，而在 ros-planning 项目中，甚至有将近一半的缺陷标题在 NELA 指标上是不合格的。这说明缺陷标题缺乏期望信息的现象十分严重。当开发者阅读这些标题时，他们往往无法从标题中获得他们所期望获取的全部信息，这会迫使他们花费额外的时间和精力来阅读正文以判断具体发生了什么以及了解一些上下文情况，并最终加重开发者的阅读负担。这向我们暴露了一个很严重的问题，即报告者可能根本不知道开发者在缺陷标题中需要哪些信息，这意味着开发者和报告者之间存在一道巨大的认知鸿沟。特别地，本文重点调查了没有传递出缺陷

1) https://github.com/moby/moby/issues/25504

2) https://github.com/moby/moby/issues/35281

3) https://github.com/ros-planning/moveit/issues/2420

效应情况和上下文情况的缺陷标题（即 NELA 值为 2 的缺陷标题）具体包含哪些类别的信息。如图 6 所示，在这一类缺陷标题中，属于报告者想法的表达要素的总占比达到了 72.88%。这表明报告者在不传达缺陷效应情况或上下文情况时，主要倾向于表达他们对缺陷的理解、分类以及为修复缺陷所提出的解决方案。

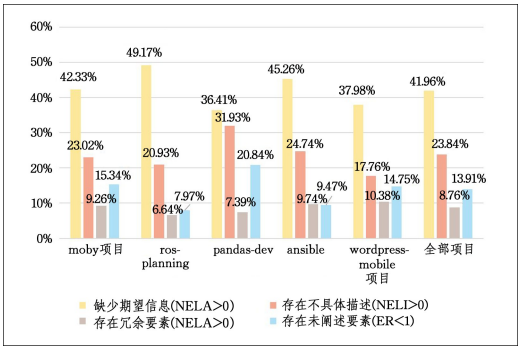

图 5　内容维度下不合格质量指标的分布

Fig. 5　Distribution of unqualified indicators of bug titles in content dimension

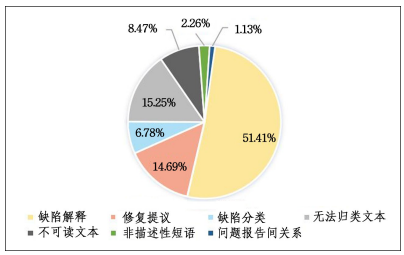

图 6　所有 NELA 值为 2 的缺陷标题下的表达类别分布

Fig. 6　Distribution of expression categories of bug titles with NELA equal to 2

此外，平均有 23.84% 的缺陷标题在 NELI 指标上是不合格的，这说明缺陷标题存在严重的描述不具体的现象。与“缺乏期望信息”不同的是，存在不具体的描述并不意味着不能为开发者带来信息收益，只是其可以带来的信息收益十分有限。为了加深对该现象的了解，本文进一步调查了待改进要素的表达类别分布。如图 7 所示，在所有待改进要素中，出现最频繁的表达类别是软件失效，其次是缺陷触发时的状况。此外，缺陷效应和缺陷上下文这两类主题出现占比之和甚至超过了 90%，这意味着待改进要素的出现往往会对开发者的阅读体验造成严重影响，因为这两类主题是开发者信息期望的重要组成部分。

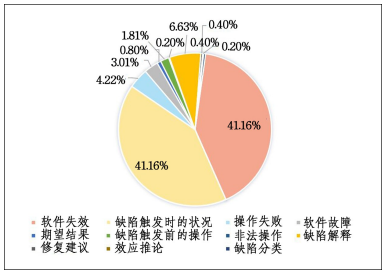

图 7　所有待改进要素下的表达类别分布

Fig. 7　Distribution of expression categories of improvement-needed bug titles

然后，有 6.64%～10.38% 的缺陷标题存在冗余要素，这些冗余要素给开发者带来的信息收益十分有限，但是却需要额外的时间来甄别和理解。此外，平均 13.91% 的缺陷标题含有未在正文中被进一步阐述的表达要素，这使得开发者无法通过阅读正文来了解标题所传达缺陷概况的相关细节。以上现象并不严重。

综上，缺陷标题的质量问题分布具有普遍性，平均有 70.07% 的缺陷标题存在质量问题，而 62.86% 的缺陷标题只存在 1～2 个质量问题。缺乏期望信息和描述不具体是最常见的两类质量问题，平均有 41.96% 的缺陷标题缺乏期望信息，平均有 23.84% 的缺陷标题需要补充具体描述。

4　讨论

1）对开发者的启发。本文对缺陷标题质量分布的调查显示，缺陷标题缺乏期望信息和缺乏细节描述的问题尤为严重，开发者需要重点关注这两类质量问题并提供相应的解决方案。例如，开发者可以在缺陷报告模板中告知报告者缺陷标题需要包含哪些信息和需要提供哪些细节描述。此外，由 3.2 节可以发现，一些报告者倾向于省略关键连词或关键实义词、引用文本信息但不加引号等，这可能会威胁到缺陷标题的可读性。为了避免出现该现象，开发者可以在缺陷报告模板中提供相应提示。最后，为了促使报告者提交高质量的缺陷标题，开发者有必要针对不同的质量要求向报告者提供相应拟写指导，例如提供统一风格的标题模板，或者是提供标签文本的书写规范等。

2）对研究者的启发。本文研究表明，绝大部分缺陷标题只存在 1～2 个质量问题，因此研究者与其关注于全面改进缺陷标题质量，不如侧重于改进某一个质量问题。例如，针对“不符合项目风格”这一质量问题，可以提供重写算法来将原标题复述为符合项目风格的形式。再者，针对“描述不具体”这一质量问题，可以将描述不具体的内容部分替换为更具体的描述。此外，评估软件缺陷标题存在的质量问题也是一个很有价值的研究方向，该工作一方面可以帮助缺陷标题生成工作筛选高质量的数据集，另一方面可以帮助报告者发现缺陷标题的质量问题并及时改进。

5　有效性威胁

5.1　内部有效性威胁

在搜集与准备数据的过程中，内部有效性威胁主要来源于有限的数据集规模。由于分析能力有限，本文仅搜集了 190 个与缺陷标题质量相关的在线文档用于分析开发者对缺陷标题的质量要求。此外，鉴于 5 个开源项目中缺陷报告数量巨大，本文仅仅采样了其中一部分缺陷报告用于实证分析，有限的实证规模可能会对我们实证结论的内部有效性造成威胁。为了让分析足够合理，在每个项目都设置了一个具有统计学意义的样本量（置信水平：95%，置信区间：5%）。

在分析质量要求和确定质量度量标准的过程中，内部有效性威胁主要来源于有限的分析规模和主观性。首先，有限的分析规模可能会威胁分类结果的多样性和全面性。然后，对质量要求的分析与对表达类别的提取面临着主观性的

威胁。为了最大限度减轻该威胁，本文严格遵守系统化的编码实践经验，旨在最小化主观性。此外，为了最大程度减少背景知识的局限以及编码时的固有个人倾向给表达类别提取带来的影响，本文选择从 2.2 节阅读过程中所得的见解出发进行编码。最后，为了缓解质量度量模型的主观性，本文严格遵守 GQM 范式[5]，并且通过 2.1 节中搜集的一系列好坏示例不断对质量度量模型进行验证与完善。

在确定缺陷标题质量分布的过程中，内部有效性威胁主要来源于标注主观性、认知局限、标注策略和泛化词集设置。为了最大限度减少标注主观性，本文确保每个缺陷标题都有两名作者进行标注，并同第三方仲裁者一同解决冲突。标注过程是持续进行的，编码本将在此过程中被不断迭代和完善，最终形成一组严格的编码标准。然后，在前两位作者对项目认知不足且正文中并未提供足够的相关阐述时，一些可分类文本可能会被标注为不可分类文本，这会给统计结果的可靠性带来威胁。其次，当标注过程中出现非连续实体时，本文仅将不连续实体中表达主要含义的部分进行标注，而舍弃另一部分。这可能会给质量分布结果带来威胁，但鉴于这种情况很少见且只是小概率影响 NELI 指标的计算，因此这种威胁是很小的而且是可以接受的。最后，计算 NELI 指标所使用的泛化词集是由 NLTK 的停用词集以及人工总结的一些泛化词组合而得，可能存在一些泛化词没有被包括在内的情况，这将导致一部分待改进的表达要素没有被检测出来，并给质量分布结果带来威胁。值得一提的是，泛化词集是在标注过程中不断补充的，这意味着本文所使用的泛化词集至少在当前缺陷报告数据集上是足够可信的，因此该威胁是足够小的。

5.2 外部有效性威胁

由于分析能力有限，本文仅使用了来自 5 个开源项目的缺陷报告，这意味着一些发现可能不能推广到其他软件项目上。鉴于其他软件项目关联着不同规模和不同知识背景的报告者和开发人员群体，且他们在缺陷报告管理任务上往往有着不同程度的管理经验，这些软件项目的缺陷标题在质量问题分布和表达形式上可能和当前研究的 5 个项目存在较大差异，这将对相关结论的外部有效性造成威胁。为了尽量保证所得结论的普遍性，本文搜集了 5 个面向不同领域的开源项目，它们来自于不同的软件社区，且基于不同的编程语言进行软件开发。

6 相关工作

6.1 缺陷报告质量的刻画

在软件维护过程中，所收集的缺陷报告往往充斥着各种质量问题，这些质量问题会给缺陷的修复效率和修复可能性带来威胁[12]。因此，如何刻画缺陷报告的质量（即如何认识缺陷报告需要满足哪些质量要求）成为了软件维护领域的一项重要课题，有大量研究工作侧重于此。

作为缺陷报告的主体部分，缺陷报告正文吸引了大量研究人员来对其质量进行刻画，并由此形成了大量研究工作，其中缺陷报告正文由缺陷描述、复现步骤、截图等要素构成。在这些研究工作中，大部分研究工作重点关注于缺陷报告正文在内容方面的质量问题。例如，Soltani 等[12]通过半结构化访谈和自动化标注来调查缺陷报告内容质量的现状，他们发现崩溃描述、复现步骤或测试用例、堆栈信息对开发者修复缺陷非常重要，然而平均有 70%左右的缺陷报告不包含这些信息内容。Yusop 等[13]通过发放调查问卷的方式来调查可用性缺陷报告内容质量的现状，他们发现报告者在可用性缺陷报告正文中极少提供与可用性缺陷相关的信息，如问题起因、UI 事件轨迹等。Garousi 等[14]通过调查问卷的方式来调查不同类别的内容的用途和重要性，他们发现，复现步骤和观测软件行为是最有用的两类信息，然而它们常出现内容不正确的情况。

除了对正文质量进行刻画外，还有一小部分研究工作侧重于对缺陷报告的其余组成部分的质量进行刻画。例如，通过对来自一些大型项目的用于指导缺陷报告提交的官方教程进行分析，Chen 等[1]发现开发者普遍要求报告者提供简洁、详实和概括性的缺陷标题。此外，他们提出了 3 个启发式来对这 3 个质量维度进行度量。然而，这项研究工作并没有考虑到可读性维度。与 Chen 等的研究工作不同的是，为了刻画缺陷标题的质量，本文进一步将调查范围扩大到关于缺陷标题质量的一系列教程、讨论、书籍和博客上，以获得更全面的关于高质量缺陷标题应该满足哪些要求的见解。

6.2 缺陷报告文本内容的分析

缺陷报告的文本内容包括缺陷报告的标题、正文和评论，是开发者在软件缺陷管理过程中的一项重要的信息来源。一些研究工作对缺陷报告文本内容的特点进行了分析，旨在了解相关软件维护活动的实践特点和实践现状。

为了了解安卓缺陷报告的内容与其可重现性之间的关系，Johnson 等[15]从 GitHub 上选取了 180 个可重现的安卓缺陷报告进行定性分析。他们发现，有 3 类常见缺陷不能被现有的自动重现技术有效重现，缺陷信息的表达方式存在多样化的特点，74%的缺陷报告存在描述不具体的重现步骤。本文重点调查了围绕缺陷标题进行内容分析的相关工作。为了了解报告者是如何描述缺陷的，Ko 等[2]展开了一项探索性调查，他们收集了来自 5 个大型开源项目的 20 万个缺陷标题并对其进行词性标注、信息分类和分布统计。他们发现缺陷报告的标题经常包含软件实体或软件行为、失效情况以及执行上下文，并且，缺陷标题的结构显示出足够的规律性。然而，该工作从报告者的表达习惯这一角度出发来对缺陷标题进行内容分析，且没有采取限制主观性的措施，这意味其研究结论并不能直接用于刻画和度量缺陷标题在内容方面的质量。此外，Sureka 等[3]进一步扩展了 Ko 等[2]的工作，他们调查了开源 Firefox 浏览器的 1 881 个缺陷报告，以探索不同严重性程度的缺陷标题之间的表达模式是否存在差异，该工作旨在探索可以用于严重性程度预测的标题特征。他们最终发现仅仅靠缺陷报告的标题无法精准预测缺陷报告的严重程度，但是存在一些具体的区分词可以用来直接推断缺陷报告的严重程度。该研究工作从缺陷严重性这一角度出发来对缺陷标题进行内容分析，侧重于讨论缺陷标题内容特征与缺陷严重性之间的关系，而不是缺陷标题内容特征与缺陷标题质量之间的关系，因此，其研究结论对刻画和度量缺陷标题在内容方面的质量并无帮助。

结束语　本文对 190 个与缺陷标题质量相关的在线文档进行主题分析[4]，并发现了 10 类针对缺陷标题的质量要求。通过使用 GQM 范式[5]，本文构建了针对缺陷标题的质量度量模型，并基于此对来自 GitHub 的 5 个开源项目的 1 804 缺陷标题进行质量分析，发现缺陷标题的质量问题是普遍存在的，而缺乏期望信息和描述不具体是最普遍的两类质量问题。基于研究发现，本文为开发者及研究者提供了一些启示。在未来工作中，本文将考虑把 AI 方法应用于缺陷标题质量的改进，例如自动检测缺陷标题存在的质量问题和为报告者提供具体的、针对性的质量问题改进指导。

参 考 文 献

[1] CHEN S, XIE X, YIN B, et al. Stay professional and efficient: Automatically generate titles for your bug reports [C]// International Conference on Automated Software Engineering (ASE). 2020: 385-397.

[2] KO A J, MYERS B A, CHAU D H. A linguistic analysis of how people describe software problems[C]// Visual Languages and Human-Centric Computing(VL/HCC'06). 2006: 127-134.

[3] SUREKA A, INDUKURI K V. Linguistic analysis of bug report titles with respect to the dimension of bug importance[C]// The Third Annual ACM Bangalore Conference. 2010: 1-6.

[4] CRUZES D S, DYBA T. Recommended steps for thematic synthesis in software engineering [C]// International Symposium on Empirical Software Engineering and Measurement. 2011: 275-284.

[5] VICTOR C R, GIANLUIGI B, ROMBACH H D. The goal question metric approach[M]// Encyclopedia of Software Engineering. 1994: 528-532.

[6] XU Y. A replication package for Examining the Quality of Bug Report Titles: An Empirical Study[EB/OL]. https://doi.org/10.5281/zenodo.7883398.

[7] TAN X, ZHOU M. How to communicate when submitting patches: An empirical study of the Linux kernel[J]. Human-Computer Interaction, 2019, 3(CSCW): 1-26.

[8] TIAN Y, ZHANG Y, STOL K J, et al. What makes a good commit message? [C]// International Conference on Software Engineering. 2022: 2389-2401.

[9] MASON M. Sample size and saturation in PhD studies using qualitative interviews[C]// Forum: Qualitative Social Research. 2010.

[10] TJONG KIM SANG E F, DE MEULDER F. Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition [C]// the Seventh Conference on Natural Language Learning at HLT-NAACL 2003. 2003: 142-147.

[11] COHEN J. A coefficient of agreement for nominal scales [J]. Educational and Psychological Measurement, 1960, 20(1): 37-46.

[12] SOLTANI M, HERMANS F, BÄCK T. The significance of bug report elements [J]. Empirical Software Engineering, 2020, 25(6): 5255-5294.

[13] YUSOP N S M, GRUNDY J, VASA R. Reporting usability defects: do reporters report what software developers need? [C]// International Conference on Evaluation And Assessment in Software Engineering. 2016: 1-10.

[14] GAROUSI V, ERGEZER E G, HERKILOČLU K. Usage, usefulness and quality of defect reports: An industrial case study [C]// International Conference on Evaluation and Assessment in Software Engineering. Limerick: ACM, 2016: 1-6.

[15] JOHNSON J, MAHMUD J, WENDLAND T, et al. An empirical investigation into the reproduction of bug reports for android apps[C]// International Conference on Software Analysis, Evolution and Reengineering. IEEE Computer Society, 2022.

XU Yong, born in 1998, postgraduate, is a student member of China Computer Federation. His main research interest is AI for software engineering.

MAO Xinjun, born in 1970, Ph.D, professor, is a member of China Computer Federation. His main research interests include software engineering, intelligent agent theory and technology, adaptive and self-organizing systems, etc.

（责任编辑：杨雪敏）