针对安卓应用缺陷报告复现的实证研究

Jack Johnson*、Junayed Mahmud†、Tyler Wendland*、Kevin Moran†、Julia Rubin‡、Mattia Fazzini*
*明尼苏达大学，明尼苏达州，美国; joh19267@umn.edu, wendl155@umn.edu, mfazzini@umn.edu
†乔治·梅森大学，弗吉尼亚州，美国; jmahmud@gmu.edu, kpmoran@gmu.edu
‡英属哥伦比亚大学，不列颠哥伦比亚省，加拿大; mjulia@ece.ubc.ca

arXiv:2301.01235v1 [cs.SE] 3 Jan 2023

摘要——**确保移动应用质量的一项关键工作是缺陷报告的提交、管理与解决。因此，研究人员投入了大量资源，致力于将移动应用缺陷管理流程中的各项任务自动化，例如复现和分类分诊。然而，这些自动化方法能否成功，很大程度上取决于其所处理缺陷报告的特征和属性。因此，理解移动应用缺陷报告对于推动报告管理技术的持续进步至关重要。尽管以往研究考察了大规模报告集合的高层次统计数据，但目前我们仍缺乏深入研究，尚不清楚移动应用问题跟踪器中通常报告的信息与复现底层故障通常所需的具体细节有何关联。**

本文对从 GitHub 上的 Android 应用中系统性挖掘出的 180 份可复现缺陷报告进行了深入分析，并探究报告中包含的信息与复现所述缺陷这一任务之间的关系。在分析中，我们重点关注三类信息：复现缺陷报告所需的环境、复现步骤（S2R）以及观察到的行为。围绕这些信息，我们对故障类型进行分类，识别报告信息所采用的模态，并评估报告中信息的质量。我们发现，缺陷报告采用多模态形式，报告中并非总会提供环境信息，而且 S2R 往往缺少信息或信息不够具体。这些发现对自动化缺陷复现技术以及更广泛的自动化缺陷报告管理方法都有重要启示。

I. 引言

近年来，随着智能手机和平板电脑日益融入用户的日常生活，移动应用程序（俗称应用）的质量变得愈发重要。应用发布给用户后，其质量主要依靠持续开展维护活动来保障；研究表明，这些活动会耗费大量工程投入 [1]。这些重要的维护活动通常围绕缺陷报告管理 展开，包括理解、复现和解决缺陷报告等相关工作。

移动应用相关的诸多独特开发约束，例如频繁发布的压力 [2], [3],应对不断演变的平台 API 的需求 [4]–[5]，大量用户反馈 [6], [7], [8], [9]–[10], 以及测试挑战 [11] ，使缺陷报告管理流程更加复杂。软件工程研究人员已认识到这些特定领域的挑战，并致力于为多种缺陷报告管理活动提供自动化解决方案，包括移动应用的缺陷报告质量评估 [12],复现 [13], [14], 分类分诊 [15],和缺陷定位 [16], [17]。

这些不同自动化解决方案的一个共同点是，它们直接处理缺陷报告中包含的信息，因此会直接受到报告各组成部分的特征和质量影响，例如环境信息（如设备、软件版本）、复现步骤（S2R）以及观察到的行为（OB）。因此，研究人员和从业者需要坚实的实证基础，明确移动应用缺陷报告的常见特征，以构建有效的自动化技术。

在先前的工作中，研究人员考察了大量缺陷报告的高层统计数据（如报告的数量和类型、修复率、修复时间）。例如，Battacharya等人 [18] 针对提交至安卓平台、涉及 24 款广泛使用的开源应用的缺陷开展了实证研究。另有研究比较了移动应用与桌面应用的高层缺陷特征 [19]。然而，据我们所知，尚无研究深入分析移动应用缺陷报告中包含的信息会如何影响缺陷复现任务。以往研究未考察这种关联，一个可能的原因是这需要手动复现真实缺陷报告，而这项工作既耗时又困难。尽管这项分析难度很大，理解这些信息仍至关重要，因为开发人员和自动化缺陷分析技术可能都需要：（i）理解报告的故障类型；（ii）理解多种信息模态，例如文本、图像或屏幕录像；以及（iii）识别或推断报告中含糊不清或缺失的信息。简而言之，对移动应用缺陷报告所含信息的特征和质量进行实证分析，对于推动移动应用缺陷报告管理的实践与科学发展都至关重要。

本文对 Android 应用的可复现缺陷报告进行了深入表征。为此，我们大幅扩展了 ANDROR2 [20] ——一个针对 Android 应用的可复现缺陷报告数据集，其中包含代表多种故障类型的缺陷。我们从托管于 GitHub [21]且可在 Google Play 商店 [22], 获取的开源 Android 应用中，补充了额外的、经人工验证且已完全复现的缺陷报告，从而得到一个包含 180 份缺陷报告的数据集。在这项工作中，我们关注 Android 应用的缺陷报告

，因为 Android 是移动应用最广泛使用的操作系统 [23]。据我们所知，我们的数据集规模最大，其中包含：(i) 已完全复现的 Android 应用缺陷报告；(ii) 用户提交和开发者提交的报告；(iii) 与相关工作不同，关注应用崩溃以外的各种故障。基于该数据集，我们对三个信息来源进行了深入分析：复现缺陷报告所需环境的描述、复现步骤以及观察到的行为。

鉴于我们研究的报告被认为是完全可复现的，我们对报告的特征（包括故障类型以及报告信息的模态）和报告信息的质量 进行了深入分析。对于报告信息的质量，我们着重考察三个方面：缺失信息的类型及其普遍程度、报告讨论串是否包含有助于复现报告的信息，以及报告信息的具体程度（即考察报告信息能否直接用于复现报告）。尽管这些方面只是描述报告信息质量的部分维度，我们相信，对这些方面的分析能为缺陷报告的复现提供有益见解，因此将重点放在这些方面。

我们的分析表明：（i）报告的故障可分为四种类型，其中三种尚未纳入现有自动化复现技术的考量；（ii）人们使用不同的信息模态来报告与环境、复现步骤和观察到的行为有关的细节；（iii）大量报告（74%）至少有一个复现步骤需要在应用中执行多个操作，这表明针对该步骤提供的信息并非总是足够具体；（iv）绝大多数报告（92%）至少缺少一个复现步骤，这说明复现报告所需的操作往往必须通过推断得出；（v）在某些情况下（19%），缺陷报告讨论能够提供有助于复现报告的附加信息。最后，我们讨论了研究结果的意义，这些结果有助于指导未来有关缺陷报告自动化复现的研究，以及更广泛的缺陷报告管理工作。

总而言之，本文的主要贡献如下：

- 我们手动挖掘并复现了大量缺陷报告，共 180 份，涵盖多种故障类型的用户提交和开发者提交的错误报告。
- 我们研究了可复现移动应用缺陷报告中的缺陷特征和信息质量。这项研究推进了先前的研究，后者并未手动验证和收集可复现的缺陷报告。
- 我们讨论了研究发现的启示，阐明了未来研究非崩溃预言机、报告信息的多模态理解、模拟环境，以及缺失和不具体的复现步骤的必要性。
- 一个复现实验包 [24] ，其中包含我们的缺陷报告数据集、数据分析报告以及用于执行的脚本



图 1：GNUCASH 应用的缺陷报告。

对研究的分析，这有助于今后复现和扩展这项工作。

II. 背景与术语

对于描述应用故障的缺陷报告，我们使用术语报告者来指提交缺陷报告的人。报告者可以是用户 或开发者。在本研究中，我们将从未为应用源代码作出贡献的人视为用户，其他所有报告者均视为开发人员。

我们从概念上将缺陷报告中的信息划分为多个部分，每个部分都详细说明报告的某个特定方面。本研究关注的部分和方面，是那些提供如何复现报告中所述故障的细节的内容。这些方面包括：环境、复现步骤 （S2R）和观察到的行为（OB）。环境包含复现报告中所述故障所需的软件和硬件信息。这部分可能包含应用版本、操作系统（OS）版本，以及发生故障的设备等信息。S2R详细说明了为复现故障而应在设备上执行的操作。我们交替使用图形用户界面操作 （或简称操作）和图形用户界面交互（或简称交互这两个术语，来指代在设备图形用户界面上执行的操作。一个S2R（构成S2R的信息单元）可以对应一个或多个图形用户界面操作。OB描述故障，可用于检查是否成功复现了故障。实际上，这些概念部分的信息可能交错出现在缺陷报告的各个段落和章节中。缺陷报告还可能包含讨论串。讨论串包含讨论消息，这些消息可以提供

有关报告所涉及的环境、S2R和OB的附加信息。

图1展示了一份用户提交的缺陷报告示例 [25]。该缺陷报告取自 GNUCash 的报告管理系统。GNUCash 是一款用于财务跟踪的应用程序，此报告经过了少量修改以便展示。缺陷报告包含与环境、S2R 和观察到的行为相关的信息，分别位于报告的软件规格、复现行为的步骤和实际行为部分。

为触发该缺陷，用户导航至交易界面，开始添加一笔新交易，然后长按表示交易金额的图形用户界面元素。故障表现为向用户显示了错误的屏幕：显示的是带有键盘视图的屏幕，而非上下文菜单。描述该故障的观察到的行为通过文本（标题中）和图像（在实际行为部分中）报告。我们将报告某条信息的方式称为报告模态（简称模态），报告者可以使用不同模态多次提供相同的信息。由于用户未能到达预期屏幕，我们将此故障认定为导航故障。我们交替使用故障类型和故障类别这两个术语来指代故障的分类。

该报告包含五个复现步骤（复现行为的步骤 部分中的编号条目），复现故障需要执行 13 个图形用户界面操作。图形用户界面操作的一个示例是在应用的交易界面中点击添加按钮，如 *2.* 点击添加按钮所示。一个复现步骤可以对应一个或多个图形用户界面操作。在此示例中，第一个复现步骤（*1.* 导航至交易界面）对应三个图形用户界面操作。我们将对应多个图形用户界面操作的复现步骤称为非具体复现步骤。其余四个复现步骤中，三个分别对应一个图形用户界面操作，另一个是可选复现步骤（*3.* 输入描述（可选））。这个可选复现步骤不计入复现故障所需的 13 个图形用户界面操作。在此示例中，有七个（13 - 3 - 3）图形用户界面操作未在任何复现步骤中描述。我们将此类图形用户界面操作称为未映射的图形用户界面操作，并称其对应于缺失的复现步骤。我们将其余操作称为已映射的图形用户界面操作。如果某个未映射的图形用户界面操作发生在第一个已映射的图形用户界面操作之前，我们就将对应于该未映射操作的缺失复现步骤称为缺失的上下文复现步骤，这表示缺陷报告缺少一些上下文信息。反之，如果某个缺失的复现步骤对应于发生在第一个已映射的图形用户界面操作之后的图形用户界面操作，我们就将该复现步骤称为缺失的内联复现步骤。

III. 方法

为了刻画可复现缺陷报告，为自动化缺陷复现研究提供信息，并更广泛地为缺陷报告管理研究提供见解，我们提出并回答了以下研究问题（RQs）：

- **研究问题$_1$：可复现缺陷报告涉及哪些故障类型？**在这个研究问题中，我们分析并归类了与可复现缺陷报告 ts. 基于此e findings from this RQ we aim to inf为自动故障识别研究提供信息。
- **研究问题$_2$：报告可复现缺陷报告中所含信息时使用了哪些信息模态？**本研究问题对用于报告环境、复现步骤和 OB 信息的模态进行分类。该问题的研究结果旨在为缺陷分类、报告复现和报告质量评估研究提供信息。
- **研究问题$_3$：可复现缺陷报告是否存在信息缺失？**我们通过分析可复现缺陷报告中有关复现报告所述故障所需操作的信息来回答这一问题。本研究问题旨在指导相关研究工作，以识别和推断缺陷报告中的缺失信息，这些信息是缺陷报告复现所必需的。
- **研究问题$_4$：可复现缺陷报告的讨论串是否包含有助于复现报告的信息？**在这一研究问题中，我们分析了解读缺陷报告讨论后获得的信息增益。此研究问题旨在评估是否需要采用结合缺陷报告及其讨论内容的方法。
- **研究问题$_5$：可复现缺陷报告中所报告的信息有多具体？**在这一研究问题中，我们探究了可复现缺陷报告所含的信息能否直接映射到复现报告所需的操作。此研究问题旨在深入了解如何利用缺陷报告中的信息来复现故障。

图2概述了我们用于回答这些研究问题的方法。简而言之，我们首先整理出一个包含可复现缺陷报告的数据集，然后通过定性和定量分析研究这些缺陷报告的特征。接下来，我们将详细介绍这些步骤。

A. 数据集创建

图2中的数据集创建部分概述了我们的数据收集工作流程，该流程分为两个阶段：错误报告筛选和故障复现。

1) 错误报告筛选：此阶段的目标是确定一组我们可以尝试复现并最终纳入数据集的错误报告。在本研究中，我们关注可复现且描述不同类型故障的用户提交和开发者提交的错误报告。据我们所知，ANDROR2 [20] 是最大的安卓应用可复现缺陷报告数据集，且并非只关注崩溃。该数据集包含 90 份用户提交的缺陷报告，这些报告与 Google Play 商店 [22] 中提供、托管于 GitHub [21]上的应用相关联。这 90 份缺陷报告是 GitHub 问题 [26], 并关联有 ANDROR2 作者编写的复现脚本。这组 90 份缺陷报告是从通过系统方式从 GitHub 挖掘出的 6,365 个问题中提取的。这 6,365 个问题包括：(i) 属于使用 Java 的代码仓库；(ii) 带有“bug”标签；(iii) 所属代码仓库包含 `AndroidManifest.xml` 文件（因为安卓应用需要此文件】【。

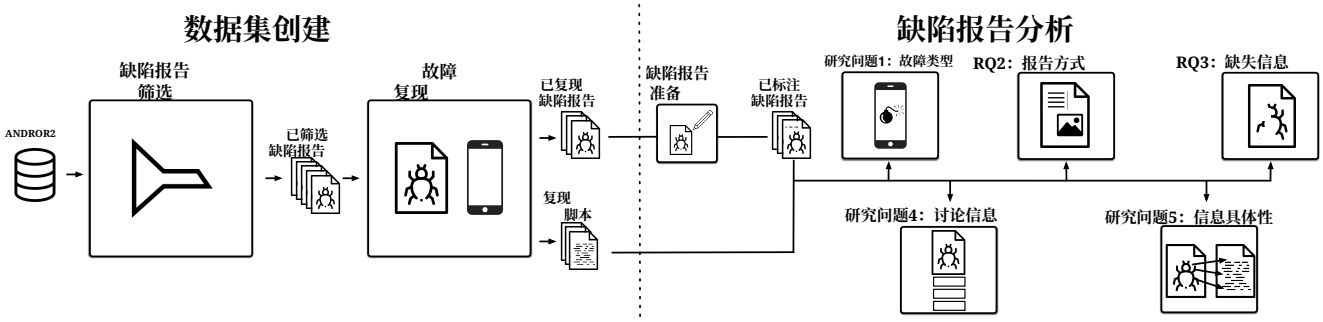


图 2：研究中所用方法概览。

文件，以便正确编译 [27]），（iv）包含“step”一词，并且（v）与同样可在 Google Play 商店下载的应用相关联。

由于我们也关注开发者提交的错误报告，因此从 ANDROR2 提供的 6,365 条 GitHub 问题中着手，找出了 90 条可复现的开发者提交的缺陷报告（与现有用户提交的缺陷报告数量相同）。为找出这 90 条开发者提交的缺陷报告，我们采用了与 ANDROR2 类似的方法。具体而言，首先将 6,365 条问题筛选为仅包含由曾为相关代码仓库做出贡献的 GitHub 用户创建的问题，最终得到 2,523 条问题。其次，我们选取了问题被采集时（2020年11月）已关闭的问题，以便更容易判断这些问题最初是否也由开发者复现过。筛选后得到 2,045 条报告。第三，分析这些问题后，我们发现某些代码仓库的问题数量远多于其他仓库。为避免缺陷报告数据集过度偏向某个特定应用，我们规定每个代码仓库最多纳入十条问题。对于问题超过十条的代码仓库，我们从中随机选取十条，最终得到涵盖 164 个应用的 645 条缺陷报告。

2）故障复现阶段：在数据集创建过程的第二阶段，我们从 645 份开发者提交的错误报告中随机选取报告，直到成功复现其中 90 份。在此过程中，我们忽略了那些很容易复现的缺陷报告，即只需打开应用即可复现的报告。

两位作者尝试复现缺陷报告中描述的故障。为复现故障，作者按照缺陷报告中包含的复现步骤，将各步骤映射为在运行相关应用的设备屏幕上执行的图形用户界面操作。如果报告缺失复现步骤，作者则通过试错法手动探索应用的功能，以确定能够补足这些缺失步骤的最短图形用户界面操作序列。当其中一位作者成功复现某份缺陷报告时，另一位作者也会尝试复现同一份报告，以确保复现出的故障与报告中描述的故障相同。对于全部 90 份缺陷报告，作者还使用 UIAutomator 框架将图形用户界面操作编码为复现脚本 [28]。

为了验证用户提交的缺陷报告是否仍可复现，我们运行了与这些报告相关的脚本（这些报告来自 ANDROR2 数据集）。由于相关应用的服务器已停止运行，其中四份报告无法复现。为了替换这些缺陷报告，我们从 ANDROR2 提供的 6,365 个 GitHub 问题中，找出了另外四份用户提交的报告并成功复现。完成这一过程后，我们得到了一组包含 90 份用户提交的可复现缺陷报告和 90 份开发者提交的可复现缺陷报告的数据，并在后续研究中对其进行了分析。

B. 缺陷报告分析

本节介绍我们为刻画与安卓缺陷报告可复现性相关的方面而开展的分析。图2中的缺陷报告分析创建部分总结了我们进行的分析。分析由论文的两位作者负责，并且每次只进行一项，以减轻认知负担。

1) 缺陷报告准备：在开展与各项研究问题（RQ）相关的分析之前，我们对缺陷报告及其讨论串中的信息进行了标注，以识别每份报告中提供有关环境、复现步骤和观察到的行为信息的部分。这一步由两位作者共同完成，分多个阶段进行；作者将报告文本描述中的每个句子，以及每个链接、图像、录屏和执行日志，都与其指定用途相关联：描述环境、复现步骤和观察到的行为。有些内容被赋予了多个标注，例如，一个句子可能同时提供复现步骤和观察到的行为信息。

2) RQ 分析$_1$ （与可复现缺陷报告相关的故障类型有哪些？）：为回答 RQ$_1$，，我们进行了结合归纳编码和主轴编码的定性分析 [29], [30]。归纳编码是一种通过手动对数据进行编码（即标记数据）来对数据进行分类的系统方法。主轴编码将各个编码相互关联，并找出能够代表原始编码抽象含义的更高层级编码。在我们的分析中，编码是用于对故障类型进行分类的标签，我们将该编码分配给描述该故障的缺陷报告。

分析由两位评审员完成。他们分析缺陷报告中对故障的描述，并使用复现脚本观察故障的具体表现。分析分为两部分。在第一部分中，两位评审员分析了一组缺陷报告样本，以制定分析编码手册——一份详细说明如何为故障分配特定编码的文件。对于每种编码，规则集

规定了为故障分配编码所需具备的特征。

这部分分析分六次迭代完成。在每次迭代中，两名评审员分别独立分析 18 份缺陷报告（占研究中所考虑报告的 10%）。两名评审员处理的是同一组缺陷报告，该报告集从尚未分析的缺陷报告中随机选取。在每次迭代结束时，两名评审员采用协商一致 [31] 来解决已创建和分配的编码之间的不一致，并确保编码过程的可靠性。我们采用这种方法，是因为它具有适用于我们这类研究的优势，即首要目标是产生新的洞见 [32]。由于我们采用了协商一致，因此评审者间一致性等指标不适用于我们的情境。为解决分歧，两名评审员共同复现故障，然后确定最终分类。例如，对于研究中考虑的一份缺陷报告 [33], , 其中一名评审员将该故障归类为崩溃，另一名评审员则将其归类为导航问题。两名评审员会面后讨论了这一分歧，并决定将该故障归类为崩溃，因为应用在将用户带回另一个界面之前显示了一个异常。

在第六次迭代中，评审员没有创建新的编码，并且已为所有报告分配了相同的编码。从那时起，评审员将剩余的 72 份缺陷报告平均分配，并独立对这些缺陷报告进行编码。在编码过程结束时，评审员还进行了主轴编码。这一步得出了四个主要的故障类别，我们将在第 IV 节中介绍这些类别。

3) RQ 分析$_2$ （回答 *RQ*：可复现缺陷报告中用于报告所包含细节的信息模态有哪些？）：用于回答 RQ_2 的分析同样基于归纳编码和主轴编码。两名评审员分析了缺陷报告准备步骤中所标注的环境、S2R 和 OB 信息。评审员通过两次迭代创建了分析编码手册，每次迭代都分析 18 份缺陷报告的样本（占全部缺陷报告的 10%）。评审员采用协商一致的方式来处理编码过程的可靠性问题。在最终确定编码手册后，作者将剩余的 144 份缺陷报告平均分配，并独立对其进行编码。

评审员在编码过程结束时进行了主轴编码。该过程得出了六种主要报告模态，详见第 IV 节。

4) RQ 分析$_3$ （可复现缺陷报告是否缺少信息？）：为回答 RQ_3, , 我们进行了两类分析。首先，我们利用在缺陷报告准备阶段创建的标注，识别报告是否完全缺少环境、复现步骤和 OB 信息。其次，在提供了复现步骤信息的情况下，我们对复现步骤进行了深入分析。具体而言，对于每份缺陷报告，我们将其中的复现步骤信息与复现脚本中的图形用户界面操作进行比较，以识别缺失的复现步骤。识别出缺失的复现步骤后，我们将其分为缺失的上下文复现步骤和缺失的内联复现步骤（定义见第 II 节）。两位作者分别独立分析每份缺陷报告，随后会面讨论并最终确定分类。

5) RQ 分析$_4$ （可复现缺陷报告的讨论串是否包含有助于复现报告的信息？）：在 RQ_4, 中，两位作者手动分析了缺陷报告讨论中的消息，以确定这些消息是否补充了有助于理解和复现缺陷报告的信息。作者利用缺陷报告准备阶段的标注，重点关注提供环境、复现步骤和 OB 信息的消息。作者分别分析每份缺陷报告，并将讨论消息中提供附加信息的数据标记为附加。在本研究中，两位作者同样会面讨论并确定最终分类。

6) RQ 分析$_5$ （可复现缺陷报告中的信息有多具体？）：为回答 RQ_5, , 我们分析了缺陷报告中提供的信息是否可直接用于复现缺陷。对于环境相关信息，两位作者检查了所提供的信息是否足以确定复现故障的环境。如果无需附加信息，我们就认为所提供的信息是具体的（否则则为非具体的）。对于 S2R，两位作者将缺陷报告中列出的每个 S2R 映射到复现脚本中相应的图形用户界面操作。如果某个 S2R 映射到多个图形用户界面操作，我们就将该 S2R 标记为非特定 S2R。我们认为其他 S2R 是具体的。对于 OB 信息，作者检查了这些信息是否足以验证故障。如果无需附加信息（即无需查看讨论消息），我们就认为所提供的信息是具体的（否则则为非具体的）。

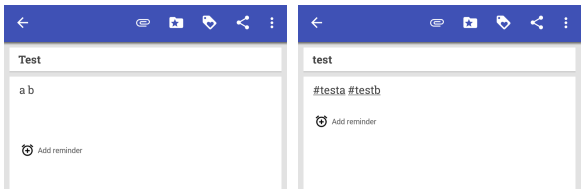
IV. 结果

本节介绍我们对可复现的安卓缺陷报告进行分析和特征描述的研究结果。

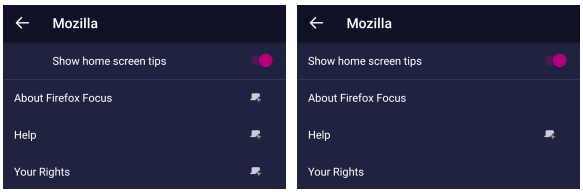
A. RQ_1：与可复现缺陷报告相关的故障类型有哪些？

我们的分析确定了四种故障类型：输出、外观、导航和崩溃。输出故障揭示了应用提供的输出中存在的问题。外观故障指出应用中不影响其功能的问题。导航故障会向用户显示错误的屏幕。崩溃会突然终止应用的执行。在所考察的缺陷报告中，我们发现33%的报告描述了输出故障，31%的报告描述了外观故障，8%的报告描述了导航故障，28%的报告描述了崩溃。这一发现值得注意，因为当前许多缺陷报告分析技术都只关注崩溃。我们将在第五节进一步讨论这些发现的影响。

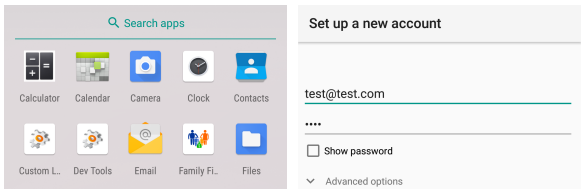
这一分布显示，输出故障、外观故障和崩溃这几类的数量大致相当，而导航故障的数量则显著较低。开发者和用户提交的缺陷报告中，这一分布相似。具体而言，在用户提交的缺陷报告中，输出故障占 33%，外观故障占 31%，导航故障占 7%，崩溃占 28%。在开发者提交的错误报告中，

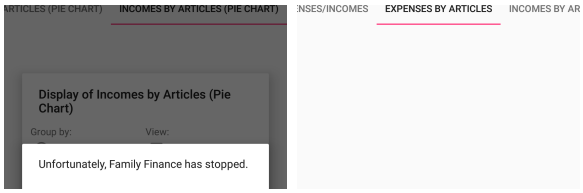


（a）左侧为输出故障示例，右侧为修复后的效果。



（b）左侧为外观故障示例，右侧为修复后的效果。





（c）左侧为导航故障示例，右侧为修复后的效果。（d）左侧为崩溃故障示例，右侧为修复后的效果。

Fig图3：所分析的缺陷报告中提及的四种故障类型的屏幕截图示例。

输出故障占 32%，外观故障占 29%，导航故障占 9%，崩溃占 29%。

我们的分析将这60个输出故障分为两个子类别：输出错误 （32个）和输出缺失 （28个）。输出错误 指应用程序的某些计算结果显示不正确，或未能正确保存到文件中的故障；输出缺失 则指某些计算结果未显示或未保存到文件中的故障。这些情况绝大多数影响应用程序的图形用户界面（56例），少数影响生成文件（4例）。

图3a左侧的屏幕截图展示了一个输出错误子类别中的故障示例。该示例摘自 OMNI NOTES（一款笔记应用）的缺陷报告 [34]。应用程序存在故障，无法正确显示与应用中笔记关联的标签值。

作为分析的一部分，我们进一步将 55 个外观故障划分为八个子类别：颜色错误 (10)、光标位置错误 (3)、内容截断 (3)、图像渲染问题(4)、图形用户界面元素缺失 (9)、方向错误 (2)、位置错误 (4)以及文本错误 (18)。我们在在线附录 [24]中详细介绍了这些子类别。图 3b 左侧的截图展示了一个位置错误 类外观故障的示例。该示例来自一份针对浏览器应用 FIREFOX FOCUS 提交的报告 [35]。在这个示例中，文本显示主屏幕提示 相较于屏幕上的其他文本元素（例如关于 Firefox Focus ）周围多出了内边距。

我们对导航故障的分析未能进一步划分出任何子类别。图3c 左侧的截图展示了一个导航故障的示例。该故障由用户针对电子邮件客户端应用 K-9 MAIL 报告 [36]。在这个示例中，用户开始设置新的电子邮件账户，进入手动配置设置后，按下返回按钮，结果退出了应用，而不是返回之前的应用界面。图3c 右侧的截图展示了应用的正确行为：用户按下返回按钮后会转到注册界面。

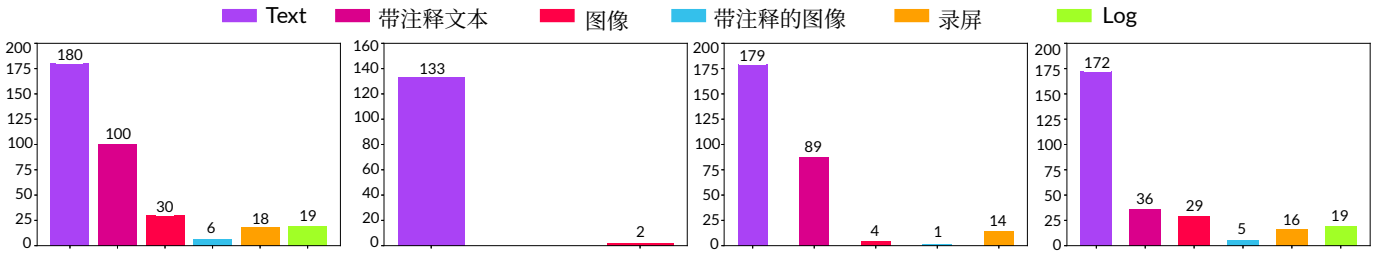
在导致崩溃的 50 个故障中，我们确定了两个主要子类别：立即崩溃 (46) 和应用卡死(4)。立即崩溃指在应用中执行某项操作后，应用便立即崩溃的故障。应用卡死指在应用中执行某项操作后，应用先变得无响应，随后经过一段时间才出现崩溃的故障。图3d左侧的截图展示了一个报告给家庭理财应用 FAMILYFINANCE 的立即崩溃故障示例 [37]。图3d右侧展示了应用中的错误修复后的界面。

研究问题$_1$ 答案：我们的分类确定了四种故障类型：输出（33%）、外观（31%）、导航（8%）和崩溃（28%）。我们还为输出（2 个）、外观（8 个）和崩溃（2 个）确定了子类别。最后，分别考察用户和开发者提交的报告时，故障分布并无明显差异。

B. 研究问题 $_2$：报告可复现缺陷报告中所含详情时，会使用哪些信息模态?

在对研究问题 2 的分析中，我们确定了六种用于报告错误信息的模态：文本、带注释文本、图像、带注释的图像、录屏和日志。文本指以纯文本形式报告的信息。带注释文本是指包含引号内文本，或包含大小写形式的文本的句子 [38]，这些文本代表应用输入或图形用户界面元素。图像指设备截图。带注释的图像是指经过编辑、突出显示内容部分的设备截图。录屏指任何能够录制设备屏幕的动画图像或视频。最后，日志指报告者提供的堆栈跟踪，这些跟踪提取自应用日志或系统日志。图 4 展示了各模态的分布情况，包括整体报告（图 4-a）、环境（图 4-b）、复现步骤（图 4-c）和观察到的行为（图 4-d）。

正如预期，文本 是最常用的模态，全部 180 份缺陷报告都使用文本来传达某些信息。带注释文本 是第二常见的模态，出现在 100 份缺陷报告中。在分析中，我们还将带注释文本模态进一步分为标注的图形用户界面文本 和 标注输入文本。标注的图形用户界面文本指报告者使用引号中的文本或大小写形式来标识相关应用图形用户界面中某个元素的缺陷报告。



a) 缺陷报告的模态。 b) 环境的模态。 c) S2R 的模态。 d) OB 的模态。

图 4：缺陷报告及其组成部分的报告模态。

例如，图 3b 所示的缺陷报告中，用户在报告里写道“*Showhomescreentips” isindented* ，以描述报告中的观察到的行为。标注输入文本这一子类别包含报告者使用引号中的文本提供应用文本输入的情况。例如，在一份关于 K-9 MAIL 的缺陷报告 [39] 中，报告者将*Add new e mailaccount with “foo@b.2”* 列为报告中的一项复现步骤。总计而言，我们发现有 100 份缺陷报告包含带注释文本（85 份包含标注的图形用户界面文本，6 份包含标注输入文本，另有 9 份同时包含这两类文本）。其余模态虽然较少见，但仍然存在，而且在许多情况下能够提供原本更难传达的信息。在所考察的缺陷报告中，报告者分别在 30、6、18 和 19 份报告中使用了图像、带注释的图像、录屏和日志模态。此外，基于图像的模态（即图像、带注释的图像和录屏）在用户提交的缺陷报告（35 份）中出现得比开发者提交的错误报告（14 份）更频繁。最后，我们注意到图像数据的使用略有逐年增加的趋势：包含基于图像信息的报告占比从 2016 年的 14% 上升至 2019 年的 36%。

图 4-b、图 4-c 和图 4-d 展示了缺陷报告各特定部分所使用的模态。图 4-b 展示了环境部分所使用的模态。绝大多数报告（133 份）使用文本模态报告环境信息，只有少数使用图像模态（2 份）。图 4-c 展示了复现步骤所使用的模态。文本是最常用的模态（出现在 179 份缺陷报告中）。带注释文本也出现在相当数量的缺陷报告中（89 份）。其余模态虽然较少见，但为复现缺陷报告提供了相关信息。19 份缺陷报告使用了文本或带注释文本以外的多种复现步骤模态，其中 16 份由用户提交，3 份由开发者提交。这 19 份缺陷报告还使用了录屏（14 份）、图像（4 份）和带注释的图像（1 份）模态。最后，图 4-d 展示了观察到的行为部分所使用的模态。文本模态再次成为最常见的模态（172 例）。然而，对于观察到的行为，与环境部分和复现步骤相比，图像和录屏模态的使用频率更高（分别为 29 例和 16 例）。60 份缺陷报告使用了文本或带注释文本以外的多种观察到的行为模态，其中 38 份由用户提交，22 份由开发者提交。这 60 份缺陷报告还使用了图像（29 份）、带注释的图像（5 份）、录屏（16 份）和日志（19 份）模态（部分缺陷报告使用了多种模态）。总体而言，用户提交的缺陷报告比开发者提交的缺陷报告更频繁地使用文本以外的报告模态。

考察报告模态与故障类型之间的关系后，我们发现，与输出故障和崩溃故障相比，描述外观故障和导航故障的缺陷报告中，使用基于图像的模态报告信息的比例更高。具体而言，描述外观故障的缺陷报告中有 45%、讨论导航故障的缺陷报告中有 43% 使用基于图像的模态；而在描述崩溃故障和输出故障的缺陷报告中，这类模态的占比分别仅为 16% 和 18%。聚焦于缺陷报告的特定部分，我们在 OB 描述中也发现了类似结果。此外，日志模态仅用于报告描述崩溃的缺陷报告中的 OB。这些结果表明，某些模态可能更适合特定的故障类型。

研究问题$_2$ 答案：我们的分类确定了六种主要的报告模态。总体而言，文本和带注释文本是最常见的模态。对于特定故障类型，某些模态出现得更频繁，例如，图像常用于外观故障和导航故障。

C. 研究问题$_3$：可复现缺陷报告是否存在缺失信息？

我们的分析发现，54 份缺陷报告未包含任何环境信息，1 份缺陷报告没有任何 S2R，另有 4 份缺陷报告未包含 OB 信息。（缺失信息是根据最初提交的缺陷报告计算的，不考虑其讨论中包含的信息，因为这正是研究问题4的重点。）

尽管只有一份缺陷报告没有任何复现步骤，但 92.2% 的缺陷报告至少缺少一个复现步骤。如第 II 节所述，缺失的复现步骤包括缺失的上下文复现步骤和缺失的内联复现步骤。88.3% 的缺陷报告至少缺少一个上下文复现步骤，37.7% 的缺陷报告至少缺少一个内联复现步骤。图 5 将缺失的复现步骤与未映射的图形用户界面操作联系起来。更准确地说，对于每份缺陷报告，该图给出了未映射的图形用户界面操作占复现该报告所需图形用户界面操作总数的百分比。该图分别给出了缺失的复现步骤、缺失的上下文复现步骤和缺失的内联复现步骤所对应的百分比。图中显示，75% 的缺陷报告因缺失复现步骤而至少有 20% 的图形用户界面操作未映射。在所有

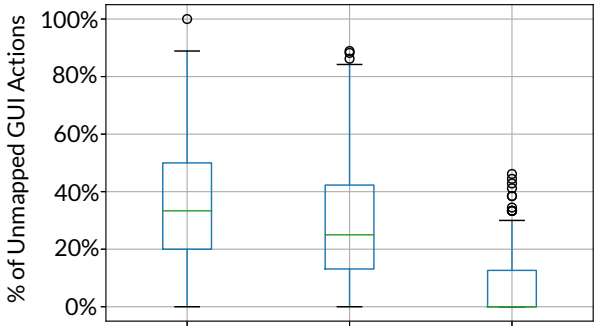


图 5：因缺失复现步骤而未映射的图形用户界面操作百分比。

缺陷报告中，缺失的复现步骤导致 43.2% 的图形用户界面操作未映射。未映射的图形用户界面操作中，33.4% 是由于缺失上下文复现步骤，9.8% 是由于缺失内联复现步骤。这些结果表明，复现缺陷报告还需要推断大量未在缺陷报告描述中明确说明的图形用户界面操作。

与开发者提交的缺陷报告中缺失的复现步骤相比，用户提交的报告中因缺失上下文 S2R 而导致的未匹配图形用户界面操作占比更低（22.5%），而开发者提交的报告中这一比例为 43.5%。但对于因缺失内联 S2R 而导致的未匹配图形用户界面操作，这种差异并不存在（用户提交的缺陷报告为 9.5%，开发者提交的缺陷报告为 10%）。我们没有观察到不同故障类型之间缺失信息的差异。

RQ_3 回答： 在所考察的各个部分中，缺陷报告的环境部分最有可能在提交的报告中缺失。很大比例的缺陷报告（92%）至少缺少一个复现步骤。缺失复现步骤导致 43.2% 的图形用户界面操作未被映射，而这些操作是复现报告中所述故障所必需的。

D. RQ_4：可复现缺陷报告的讨论串是否包含有助于复现这些缺陷报告的信息？

为回答这一 RQ，我们分析了数据集中缺陷报告相关的讨论，并找出了对话中补充的、与复现缺陷相关的信息。共有 35 份缺陷报告包含了额外信息，具体涉及环境、S2R 或缺陷报告中的观察到的行为。在这 35 份缺陷报告中，25 份由用户提交，10 份由开发者提交。此外，在其中 22 份缺陷报告中，开发者明确要求在讨论中补充相关信息。

在讨论中，有 20 次补充了环境信息，11 次补充了 S2R，9 次补充了 OB。这些数字的总和高于包含附加信息的缺陷报告总数，因为有些讨论（共五次，其中四次包含两条消息，一次包含三条消息）包含多条提供附加信息的消息。虽然补充信息的情况并不多，但这些结果表明

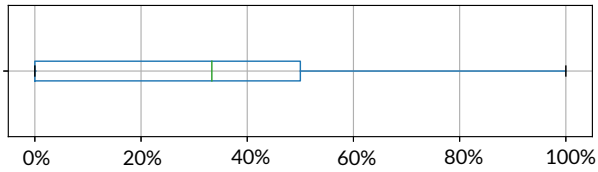


图 6：各缺陷报告中非特定 S2R 的比例。

后续对话可用于复现报告中的缺陷。此外，考虑到 RQ_3，中有大量报告缺少环境信息，且其中的图形用户界面操作无法匹配，自动化技术可以尝试通过迭代式或交互式错误复现方法识别并自动获取这些信息。

从不同故障类型来看，描述输出故障的缺陷报告在讨论中补充的信息最多。在包含补充信息的 35 份缺陷报告中，17 份描述了输出故障，15 份报告了崩溃，2 份描述了外观故障，1 份讨论了导航故障。

RQ_4 回答： 在所考察的缺陷报告中，有 35 份通过后续跟进中的基于消息的讨论获得了与复现报告相关的附加信息。在 22 份报告中，开发人员明确请求了这些信息。最后，在这 35 份报告中，有 20 份补充了环境信息，11 份补充了 S2R，10 份补充了 OB。

E. RQ_5：可复现缺陷报告中所报告的信息有多具体？

报告环境信息时，这些信息可以直接映射为复现故障的操作。也就是说，可以选择正确的应用版本、Android 版本和设备来复现故障。对于 OB，我们必须查看六份报告的讨论，以便更好地理解所报告故障相关的问题；这意味着在我们的分析中，这些缺陷报告中描述的 OB 不够具体，无法复现故障。至于 S2R，73.9% 的缺陷报告至少包含一条所报告的 S2R，无法直接映射为单个图形用户界面操作，而是需要多个图形用户界面操作。根据第 II 节中定义的术语，这意味着这些缺陷报告至少包含一条非特定 S2R。图 6 展示了我们数据集中每份缺陷报告中非特定 S2R 的百分比。在所有报告中，S2R 部分的 S2R 平均有 36% 属于非特定 S2R。结果表明，复现报告时，需要弥补将 S2R 映射到相应图形用户界面操作这一环节的不足。

从故障类型来看，描述导航故障的缺陷报告中，非特定 S2R 的平均占比最高（40%），而描述输出故障的缺陷报告中占比最低（34%）。这一结果表明，不同报告故障类型的 S2R 具体程度略有差异。用户报告的非特定 S2R 平均占比（34.6%）与开发人员报告的占比（35.8%）也相差不大。

研究问题$_5$ 答案：在绝大多数情况下，环境和OB信息足够具体，能够复现所报告的故障。很大比例的报告（73.9%）至少包含一个非特定 S2R，所有报告中非特定 S2R 的平均占比为36%。

V. 讨论与启示

1) 需要用于理解非崩溃预言机的新型自动化技术。 现有的大多数移动应用自动化缺陷复现方法，都侧重于复现导致崩溃的缺陷 [13], [14]。这可能是因为与崩溃相关的故障更容易识别，例如可以通过检测崩溃对话框来判断，从而确定导致崩溃的缺陷是否已被复现。然而，我们的分析表明，超过 70% 的缺陷报告描述的是崩溃以外的故障，因此需要更复杂的预言机定义和检测方法。例如，缺陷报告复现的自动化技术可以借助利用计算机视觉定义视觉预言机的技术，例如通过颜色直方图分析检测错误的配色主题。同样，导航故障可能需要分析静态计算得到的程序状态图，以确定导航路径的可行性。通过推导程序不变量来定义预言机的近期工作（例如， [40]）也可以进一步促进预言机构建。

2) 有必要对缺陷报告信息进行自动化多模态理解。 我们的分析表明，缺陷报告可以混合多种信息模态，包括文本、图像和录屏，以捕捉有关特定缺陷的不同信息。然而，近期大多数关于自动化缺陷报告复现与分析的研究仅考虑文本模态 [12], [13], [14]。鉴于我们的 RQ_3,分析揭示了缺失信息普遍存在，即使在可复现的报告中也是如此，自动化报告分析应力求分析已报告信息的所有 类型，以实现更稳健、更全面的分析。因此，我们需要用于多模态理解缺陷的新技术。例如，可以利用连接图像与自然语言的深度学习技术（如密集图像字幕生成 [41]），将文本信息与视觉信息关联起来，从而更全面地分析报告。此外，对于复现步骤，自动化技术还需要识别如何恰当地排列信息，而这可以通过利用从应用中静态或动态计算得出的窗口转换图来实现 [42], [43]。

3）推断和模拟应用环境的技术至关重要。 长期以来，安卓应用开发者一直难以解决平台和设备生态系统碎片化所带来的问题。这些问题也会出现在缺陷报告中。正如在分析所考察的缺陷报告时所发现的，缺陷可能会在设备和平台版本的特定组合下出现。考虑到提交的缺陷报告中并不总是包含这些信息（30% 的案例中缺失），因此需要能够推断并确定环境设置（例如设备和平台版本）优先级的技术，以推动更先进的自动化移动缺陷报告分析技术的研究。此外，考虑到应用发布频繁，[44], [45]而且缺陷报告并不总是包含相关的应用版本，自动推断与缺陷报告对应的应用版本将大有裨益。这项任务可以通过将缺陷报告中的信息自动映射到应用中的图形用户界面组件或代码实体来完成。

3) 推理缺失的复现步骤是必要的。我们的分析表明，所研究的缺陷报告中绝大多数（92%）至少缺少一个复现步骤。这给自动化报告分析技术带来了显著挑战，因为这类技术很可能需要推断缺失信息，才能提供可靠的分析。现有技术确实提供了先进的解决方案（即基于随机探索），以帮助填补某些缺失部分 [13], [14], [12]。不过，可能还需要其他技术，以便更细致地推断缺失步骤。例如，未来的技术可以检查现有的缺陷报告语料库（例如与本研究相关的材料），并尝试通过从完整缺陷报告语料库中学习到的模式来推断缺失步骤。

4) 处理缺陷报告数据中的非特定 ***S2R***是一项重大挑战。除了复现步骤缺失现象普遍之外，我们的分析还发现，36%的复现步骤对应多个图形用户界面操作。这些复现步骤描述的是“高层次”操作，其中并未明确指出具体操作或目标图形用户界面元素。这给自动化复现和报告分析技术带来了棘手的推理问题。现有技术尝试通过本体匹配 [13]或文本的神经表示 [12]，以及随机探索，来消除这类歧义。不过，可能还需要其他技术来映射非特定操作或目标。例如，未来的技术或许可以利用多模态图像描述模型推断应用程序控件或功能的描述，从而更好地将文本映射到应用程序运行时信息。根据从规范的复现步骤集合中学习到的模式，自动“修复”含糊不清的缺陷报告步骤，也可能是一个值得探索的方向。此外，还可以从现有测试用例中的图形用户界面操作序列提取复现步骤描述，并将其映射到缺陷报告中的复现步骤，以便复现缺陷。

总之，本文进行的分析揭示了若干值得关注的启示，这些启示将影响自动化缺陷报告复现、报告撰写、分析和管理方面的未来工作。我们相信，未来的工作将受益于这些发现，以及它们所指向的潜在新研究方向。

VI. 有效性威胁

尽管我们在收集、分析和报告结果时遵循了系统化的方法，但讨论本研究的有效性威胁，对于全面呈现研究发现至关重要。就外部有效性而言，我们的结果

可能无法推广到其他 Android 应用中的缺陷。不过，考虑到所研究应用及报告的数量、类型多样性和受欢迎程度，我们认为所研究的报告应能合理代表整体 Android缺陷报告。我们考察了最新的可复现缺陷报告数据集（其中包含非崩溃错误），并扩展该数据集，将开发者提交的错误报告也纳入其中。该数据集涵盖了规模和类别各不相同的应用。另一个潜在威胁是我们只使用了开源应用。不过，评估中包含 FIREFOX FOCUS 和 Simplenote 等应用，它们功能复杂，安装量达数百万。就构念效度而言，我们的结果可能受到用于分析的工具中错误的影响。为减轻这一威胁，我们对工具进行了充分测试，并由多位作者手动检查结果。最后，我们还进行了定性分析，这可能会受到评估者之间理解差异的影响。为减轻这一威胁，我们采用基于协商一致的开放式编码 [31]。

VII. 相关工作

A. 错误报告的一般研究

相关工作研究了错误报告属性，以便更好地理解错误报告管理流程中的多项活动 [46], [47], [48], [49], [50], [51], [52], [53]‚[54]。在不同主题中，这一研究方向分析了错误报告内容、开发者和用户参与缺陷报告讨论的情况、分类分诊以及错误修复。Bettenburg 等人开展的一项重要研究 [46] 确定了错误报告中应包含的理想要素。在后续研究中，Bettenburg 等人 [47] 还表明，重复错误报告包含一些有助于缺陷分类的额外信息。Sahoo 等人 [48] 通过开展实证研究，确定了缺陷复现所需的主要组成部分。一些先前的研究主要聚焦于用户提交的缺陷报告。这一研究方向考察了用户通常如何描述软件问题 [51], 、高级用户所提供信息的实用性 [52]„ 以及用户社区的期望 [55]。本文研究了与用户提交和开发者提交的安卓缺陷报告相关的关键方面。此外，我们还聚焦于与缺陷报告复现相关的方面，具体考察缺陷报告中的信息与复现报告所需信息之间的关系。

B. 移动应用错误报告研究

最初的大多数缺陷报告研究都聚焦于桌面应用。不过，由于智能手机应用在过去十年间日益普及、易用且广受欢迎，研究人员也开始关注移动应用缺陷报告的特征。Zhou 等人 [19] 开展了一项研究，以了解桌面软件与移动软件之间的缺陷管理。Bhattacharya 等人 [18]研究了移动缺陷报告及其错误修复流程。Aljedaani 等人 [56]比较了安卓与 iOS 的缺陷报告。Zhang 等人 [57] 研究了移动应用的缺陷报告，并为这些报告添加标签，并计算其与先前标注报告之间的相似度。在我们的研究中，我们复现了缺陷报告，描述了与这些报告相关的故障，分析了报告所提供信息的实用性，并对报告模态进行了分类。以往的研究也生成了包含相关缺陷报告的安卓缺陷数据集。Wendland 等人 [20] 创建了一个包含可复现的用户提交缺陷报告的数据集。Su 等人 [58] 基于 GitHub 问题创建了一个崩溃缺陷数据集。Fazzini 等人 [13]和 Zhao 等人 [14]也为其关于缺陷报告自动化复现的研究整理了一个崩溃缺陷数据集。据我们所知，与这些数据集相比，本文首次创建并在研究中采用了一个非崩溃且可复现的缺陷报告数据集，其中同时包含用户提交和开发者提交的报告。

八、结论

我们开展了一项实证研究，对可复现的安卓缺陷报告进行了归类。具体而言，我们人工复现了从 GitHub 上的安卓应用中系统挖掘出的 180 份缺陷报告，并调查了缺陷报告中包含的信息与复现这些报告的任务之间的关系。我们的分析发现，报告中的故障可分为四类，其中三类尚未被现有自动化复现技术纳入考量；报告者会采用不同模态来报告与复现故障相关的信息；大量报告（74%）至少包含一个非特定 S2R（即执行 S2R 所述操作需要多个图形用户界面操作）；绝大多数报告（92%）未提供复现报告所需的全部 S2R；并且在某些情况下（19%），缺陷报告中的讨论能够提供有助于复现报告的附加信息。

未来工作中，我们计划先向安卓开发者介绍研究发现，随后开发辅助自动化复现缺陷报告的技术。为支持缺陷报告的自动化复现，我们计划首先定义一种方法，利用自然语言处理和计算机视觉技术，将 OB 信息自动编码为测试预言机，从而辅助复现输出、外观和导航故障。其次，我们计划定义一种技术，整合通过不同模态报告的 S2R 信息。第三，我们计划定义一种技术，利用现有测试用例中包含的信息，帮助将非特定 S2R 映射到相应的图形用户界面操作。最后，我们认为有必要进一步研究其他领域软件的缺陷报告复现，而这些研究也有望为这些领域的缺陷报告管理技术提供参考。

致谢

本研究部分由 Facebook 资助，并获得美国国家科学基金会 CCF-2007246 和 CCF-1955853 项目资助。本文所表达的任何观点、研究结果和结论均为作者所有，并不一定反映资助方的意见。

参考文献

[1] G. Tassey，《软件测试基础设施不足的经济影响》，美国国家标准与技术研究院，技术报告，2002 年。[2] G. Hu、X. Yuan、Y. Tang 和 J. Yang，《借助 appdoctor 高效且有效地检测移动应用程序缺陷》，载于第九届欧洲计算机系统会议论文集，丛书 EuroSys'14，纽约州纽约市，2014 年，第 18:1–18:15 页。[3] N. Jones，《优化移动测试工作的七项最佳实践》，Gartner，技术报告 G00248240，2013 年。[4] D. Han、C. Zhang、X. Fan、A. Hindle、K. Wong 和 E. Stroulia，《通过对厂商特定缺陷进行主题分析，理解安卓碎片化》，载于*2012* 年第 *19* 届逆向工程工作会议论文集，丛书 WCRE '12。美国华盛顿州华盛顿：IEEE计算机学会，2012 年，第 83–92 页。 [在线]。网址：http://dx.doi.org/ 10 1109 2012 18 。/WCRE。。《安卓碎片化统计[5] 数据 http://opensignal.com/reports/2014/android-fragmentation/》，2014 年。[6] A. Ciurumelea、A. Schaufelb¨uhl、S. Panichella 和 H. Gall，《分析移动应用的评论和代码，以改进发布规划》，载于*IEEE* 第 *24* 届软件分析、演化与再工程国际会议论文集，丛书 SANER'17，2017 年 2 月，第 91–102 页。[7] A. Di Sorbo、S. Panichella、C. V. Alexandru、J. Shimagaki、C. A. Visag- gio、G. Canfora 和 H. C. Gall，《用户会对我的应用做哪些改变？总结应用评论以推荐软件变更》，载于第 *24* 届 *ACM SIGSOFT*软件工程基础国际研讨会论文集，丛书 FSE'16，美国华盛顿州西雅图，2016 年，第 499–510 页。[8] F. Palomba、P. Salza、A. Ciuru melea、S. Panichella、H. Gall、F. Ferrucci 和 A. De Lucia，《基于用户评论为移动应用推荐并定位变更请求》，载于第 *39* 届国际软件工程会议论文集，丛书 ICSE'17，美国新泽西州皮斯卡塔韦，2017 年，第 106–117 页。[9] F. Palomba、M. Linares-V´asquez、G. Bavota、R. Oliveto、M. D. Penta、D. Poshyvanyk 和 A. D. Lucia，《众包用户评论以支持移动应用的演化》，《系统与软件杂志》，第 143–162 页，2018 年。[10] F. Palomba、M. Linares-Vasquez、G. Bavota、R. Oliveto、M. D. Penta、D. Poshyvanyk 和 A. D. Lucia，《用户评论很重要！追踪众包评论以支持成功应用的演化》，载于*IEEE* 软件维护与演化国际会议论文集，丛书 ICSME'15，2015 年 9 月，第 291–300 页。[11] S. Choudhary、A. Gorla 和 A. Orso，《安卓自动化测试输入生成：我们准备好了吗？（e）》，载于 *2015* 年第 *30* 届 *IEEE/ACM* 自动化软件工程国际会议（*ASE*）。美国加利福尼亚州洛斯阿拉米托斯：IEEE计算机学会，2015 年 11 月，第 429–440 页。 [在线]。网址：https://doi.ieeecomputersociety.org/10.1109/ASE.2015.89[12] O. Chaparro、C. Bernal-C´ardenas、J. Lu、K. Moran、A. Marcus、M. Di Penta、D. Poshyvanyk 和 V. Ng，《评估缺陷报告中复现步骤的质量》，载于*2019* 年第 *27* 届 *ACM* 欧洲软件工程会议与软件工程基础研讨会联合会议论文集。计算机协会，2019 年，第 86–96 页。[13] M. Fazzini、M. Prammer、M. d'Amorim 和 A. Orso，《自动将缺陷报告转换为移动应用的测试用例》，载于第 *27* 届 *ACM SIGSOFT*软件测试与分析国际研讨会（*ISSTA*）论文集，2018 年，第 141–152 页。[14] Y. Zhao、T. Yu、T. Su、Y. Liu、W. Zheng、J. Zhang 和 W. G. J. Halfond，《ReCDroid：根据缺陷报告自动复现 Android 应用程序崩溃》，载于第 *41* 届国际软件工程会议（*ICSE*）论文集，2019 年，第 128–139 页。[15] X. Xia、D. Lo、Y. Ding、J. M. Al-Kofahi、T. N. Nguyen 和 X. Wang，《借助专用主题模型改进自动化缺陷分类》，《*IEEE*软件工程汇刊》，第 43 卷，第 03 期，第 272–297 页，2017 年 3 月。[16] M. Pradel、V. Murali、R. Qian、M. Machalica、E. Meijer 和 S. Chandra，《Scaffle：在数百万个文件中定位缺陷》，载于第 *29* 届 *ACM SIGSOFT*软件测试与分析国际研讨会论文集。美国纽约州纽约市：计算机协会，2020 年，第 225–236 页。[17] T. Zhang、W. Hu、X. Luo 和 X. Ma，《一种基于提交消息的安卓应用程序缺陷定位方法》，《国际软件杂志》工程与知识工程，第 29 卷，487，2019 年 4 月，第 457 页。[18] P. Bhattacharya、L. Ulanova、I. Neamtiu 和 S. C. Koduru，"开源安卓应用中的缺陷报告与错误修复实证分析"，载于《软件维护与再工程》（CSMR），2013 年第 17 届欧洲会议，2013 年，第 133–143 页。[19] B. Zhou、I. Neamtiu 和 R. Gupta，"开源项目中缺陷及其修复的跨平台分析：桌面端、安卓与 iOS"，《第 19 届软件工程评估与评价国际会议论文集》，2015 年。[20] T. Wendland、J. Sun、J. Mahmud、S. M. H. Mansur、S. Huang、K. Moran、J. Rubin 和 M. Fazzini，"Andror2：一个针对 Android 应用人工复现的缺陷报告数据集"，2021 IEEE/ACM 第 18 届软件仓库挖掘国际会议（MSR），第 600–604 页，2021 年。[21]（2021 年 10 月）Github。 [在线]。网址：https://github.com [22]（2021 年 10 月）Google Play。 [在线]。网址：https://play.google.com [23] "全球移动操作系统市场份额"，StatCounter，技术报告，2021 年。[24] A. Authors，"在线附录 https://sites.google.com/view/ 2021bugreportingstudy/home"，2021 年。[25]（2021 年 10 月）缺陷：长按金额输入框会弹出 QWERTY 键盘。 [在线]。网址：https://github.com/codinguser/ gnucash-android/issues/ 489 [26] 2021， [（1 月）关于问题。在线]。网址：https://docs.github.com/ en/github/managing-your-work-on-github/about-issues [27]（2021 年 1 月）应用清单概览。 [在线]。网址：https: // developer.android.com/guide/topics/manifest/manifest-intro [28]（2021 年 1 月）UIAutomator。 [在线]。网址：https://developer.android. com/training/testing/ui-automator [29] J. Corbin 和 A. Strauss，《定性研究基础：扎根理论的技术与程序》。Sage 出版社，2014 年。[30] M. B. Miles、A. M. Huberman 和 J. Salda˜na，《定性数据分析：方法资料手册》。Sage 出版社，2018 年。[31] J. L. Campbell、C. Quincy、J. Osserman 和 O. K. Pedersen，"深度半结构化访谈的编码：单元划分、编码者间信度与一致性问题"，《社会学方法与研究》，第 42 卷，第 3 期，第 294–320 页，2013 年。[32] E. R. Morrissey，"问卷数据编码中的误差来源"，《社会学方法与研究》，第 3 卷，第 2 期，第 209–232 页，1974 年。[33]（2018 年 8 月）无法打开添加/移除药品库存对话框。 [ ] 134 在线。网址：[34] 2021， [https://github.com/citiususc/calendula/issues/（10 月）使用复选框移除标签文本时出现缺陷。在线]。网址：https://github.com/federicoiosue/Omni-Notes/issues/ 634 [35] 2021， [（10 月）主屏幕提示开关缩进不正确。在线]。网址：https://github.com/mozilla-mobile/focus-android/issues/ 3304 [36] 2021， （10 月）手动设置时按下返回按钮会导致应用关闭。 [ ] 9 9 3971 在线。网址：[37] 2021， https://github.com/k mail/k- /issues/（10 月）在报告部分切换视图时应用崩溃。 [在线]。网址：https://github.com/zwieback/FamilyFinance/issues/ [38] 2021， （10 月）用户体验文案中的标题式大小写与句式大小写。[在线]。网址：https://uxdesign.cc/ title-case-vs-sentence-case-in-ux-writing-2120871922 61 [39] 2021， （10 月）电子邮件地址格式无效时发生崩溃。 [在线]。网址：https://github.com/k 9 9 3255 mail/k- /issues/ T. Su、Y. [40] Yan、J. Wang、J. Sun、Y. Xiong、G. Pu、K. Wang 和 Z. Su，"用于检测非崩溃逻辑错误的 Android 应用全自动功能模糊测试"，《ACM 程序设计语言》，第 5 卷，OOPSLA 专刊，10 月。 2021 [ ] 10 1145 3485533 。在线。网址：[41] https://doi.org/ 。/ J. Johnson、A. Karpathy 和 L. Fei-Fei，"DenseCap：用于密集图像描述的全卷积定位网络"，载于《IEEE 计算机视觉与模式识别会议论文集》，2016 年。[42] D. Lai 和 J. Rubin，"面向 Android 应用的目标驱动探索"，载于 2019 年第 34 届 IEEE/ACM 自动化软件工程国际会议（ASE），2019 年，第 115–127 页。[43] S. Yang、H. Wu、H. Zhang、Y. Wang、C. Swaminathan、D. Yan 和 A. Rountev，"Android 的静态窗口转换图"，《国际自动化软件工程期刊》，第 25 卷，第 4 期，第 833–873 页，2018 年 12 月。[44] J. Gao、L. Li、T. F. Bissyand´e 和 J. Klein，"移动应用复杂度的演变"，载于 2019 年第 24 届复杂计算机系统工程国际会议（ICECCS），2019 年，第 200–209 页。[45] S. McIlroy、N. Ali 和 A. E. Hassan，"新鲜应用：Google Play 商店中频繁更新的移动应用实证研究"，《实证软件工程》，第 21 卷，第 3 期，第 1346–1370 页，2016 年。

[46] N. Bettenburg、S. Just、A. Schr¨oter、C. Weiss、R. Premraj 和 T. Zim- mermann，“什么样的缺陷报告才算好？”，载于《第 *16* 届 *ACM SIGSOFT* 软件工程基础国际研讨会论文集》。美国纽约州纽约市：ACM，2008 年，第 308–318 页。[47] N. Bettenburg、R. Premraj、T. Zimmermann 和 S. Kim，“重复缺陷报告有害……真的吗？”，载于《国际软件维护会议论文集》，ICSM’08 系列，2008 年，第 337–345 页。[48] S. K. Sahoo、J. Criswell 和 V. Adve，“服务器软件报告缺陷的实证研究及其对自动化缺陷诊断的启示”，载于《第 *32* 届 *ACM/IEEE* 国际软件工程会议论文集——第 *1* 卷》，ICSE ’10 系列。美国纽约州纽约市：计算机协会，2010 年，第 485–494 页。 [在线]。网址：https://doi.org/10 1145 1806799 1806870 。/ 。S. Breu、R. Premraj、J. Sillito 和 T. Zimmermann，“缺陷报告中的信息需求：改善开发者与用户之间的协作”，载于《计算机支持的协同工作会议论文集（*CSCW’10*）》，2010 年，第 301–310 页。[50] S. Davies 和 M. Roper，“缺陷报告里有什么？”，载于《第 *8* 届 *ACM/IEEE* 实证软件工程与度量国际研讨会论文集》，ESEM ’14 系列。美国纽约州纽约市：计算机协会，2014 年。 [在线]。网址：https://doi.org/10 1145 2652524 2652541 。/ 。A. J. Ko、B. A. Myers 和 D. H. Chau，“人们如何描述软件问题的语言学分析”，载于《视觉语言与以人为中心的计算研讨会论文集（*VL/HCC’06*）》，2006 年，第 127–134 页。[52] A. J. Ko 和 P. K. Chilana，“高级用户如何帮助并阻碍开源缺陷报告，”载于*SIGCHI*计算系统中的人为因素会议论文集，系列：CHI ’10。美国纽约州纽约：计算机协会，2010年，第1665–1674页。 [在线]。可访问：https://doi.org/10 1145 1753326 1753576 。/。F. Thung、D. Lo 和 L. Jiang，“自动缺陷分类”，载于逆向工程工作会议论文集（*WCRE’12*），2012年，第205–214页。[54] P. J. Guo、T. Zimmermann、N. Nagappan 和 B. Murphy，“刻画并预测哪些缺陷会得到修复：微软 Windows 的一项实证研究”，载于第*32*届 *ACM/IEEE* 国际软件工程会议论文集——第*1*卷，系列：ICSE ’10。美国纽约州纽约：计算机协会，2010年，第495–504页。 [在线]。可访问：https://doi.org/10.1145/1806799。1806871[55] P. K. Chilana、A. J. Ko 和 J. O. Wobbrock，“理解开放式缺陷报告中对不受欢迎行为的表达”，载于*2010*年 *IEEE*视觉语言与以人为中心的计算研讨会，2010年，第203–206页。[56] W. Aljedaani、M. Nagappan、B. Adams 和 M. Godfrey，“比较两个开源跨平台浏览器应用在 iOS 与安卓平台上的缺陷”，载于*2019*年 *IEEE/ACM* 第*6*届移动软件工程与系统国际会议（*MOBILESoft*），2019年，第76–86页。[57] T. Zhang、H. Li、Z. Xu、J. Liu、R. Huang 和 Y. Shen，“为移动应用的问题报告添加标签”，*IET* 软件，第13卷，第528–542页，2019年。[58] T. Su、J. Wang 和 Z. Su，“针对现实世界中的缺陷对安卓自动化图形用户界面测试进行基准评估。”美国纽约州纽约：计算机协会，2021年，第119–130页。