Beyond Single-PR Issues: Understanding Multi-PR Issue Resolution in Game Development

Nimmi Weeraddana
nimmi.weeraddana@ucalgary.ca
University of Calgary, Canada

Abstract

Video games are a special type of software that differ from other types of software systems in that they aim to deliver immersive experiences to users. As a result, issue reports (e.g., bug reports) in video game projects may require iterative fixes that span multiple pull requests (PRs). A PR is a mechanism used in platforms such as GitHub, which are built on distributed version control systems (e.g., Git), through which developers propose code changes for review, discussion, and eventual integration into the main codebase. We refer to an issue report that is linked to multiple PRs as a *multi-PR issue*. Despite their potential impact on development efforts and coordination, such multi-PR issues remain largely unexplored.

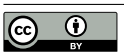
In this paper, we present a large-scale empirical study of multi-PR issues in open-source game development. We analyze their prevalence, derive a taxonomy explaining why multiple PRs are associated with a single issue, and compare their resolution timelines and development effort against *single-PR issues*, i.e., issues resolved through exactly one PR. Our results show that approximately 5% of issues are resolved through multiple PRs. Most of these cases arise from iterative refinement. Quantitatively, multi-PR issues take substantially longer to resolve, with a median resolution time four times higher than that of single-PR issues. They are also associated with moderately higher commit-level activity. Our findings have practical implications for both practitioners and researchers. For practitioners, multi-PR issues require more careful planning, as they involve higher commit-level effort and are more likely to experience prolonged resolution times, potentially threatening release schedules. Moreover, when multiple contributors independently submit PRs for the same issue, teams risk duplicated implementation effort and increased review overhead, underscoring the importance of early coordination mechanisms (e.g., ownership assignment or consolidation of parallel fixes). For researchers, our results highlight the need for empirical models and tools that explicitly account for iterative resolution patterns rather than assuming single-PR fixes. We further encourage researchers to extend our taxonomy to other domains, such as web and mobile applications.

CCS Concepts

• **Software and its engineering → Open source model**.

Keywords

pull requests, open source, video game development



FSE Companion '26, Montreal, QC, Canada

ACM ISBN 979-8-4007-2636-1/26/07
https://doi.org/10.1145/3803437.3806367

ACM Reference Format:
Nimmi Weeraddana. 2026. Beyond Single-PR Issues: Understanding Multi-PR Issue Resolution in Game Development. In *34th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (FSE Companion '26), July 5–9, 2026, Montreal, QC, Canada.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3803437.3806367

1 Introduction

Open-source game development has grown into a vibrant ecosystem encompassing playable games, game engines, and supporting tools [17, 21, 30]. Unlike other types of software projects, game development involves complex interactions between gameplay logic, performance constraints, real-time responsiveness, and immersive experience for users [5, 14, 20, 23]. Thus, defects in games often manifest as emergent behaviors that are difficult to diagnose and resolve through isolated code changes. Understanding how such defects are addressed in practice is essential for advancing empirical software engineering research in the game development domain.

To address bug reports or issue reports, developers prepare change sets and submit them to the main codebase via pull requests (PRs). While there have been several empirical studies analyzing PRs and their resolution times, they typically model issues and PRs as a one-to-one relationship or collapse multiple PRs into a single resolution event [3], thereby obscuring iterative fixing behavior and the additional coordination effort it entails. However, little evidence suggests that some issue reports (e.g., bug reports [31, 33], crash reports [34], and/or feature requests [13]), particularly in complex domains such as game development, require multiple, iterative PRs before being fully resolved [24]. Such *multi-PR issues* challenge the notion of single-PR issues and point to a more incremental and exploratory development process.

Despite their potential importance, multi-PR issues remain largely unexplored in empirical research [24]. As a consequence, little is known about how frequent multi-PR issues are, what characteristics distinguish them from single-PR issues, and how they impact resolution time, collaboration, and code change complexity—particularly within game development projects. In this paper, we present an empirical study of multi-PR issues in open-source game development. We begin with OSSGameBench [16], an existing dataset of 970 open-source game-related repositories with explicit mappings between issue reports, pull requests, and commits. By using this dataset, we systematically identify issues that are resolved through multiple PRs, i.e., multi-PR issues, and analyze the reasons for their existence and their resolution dynamics. Specifically, we strive to answer the following research questions (RQs):

RQ1: How common are multi-PR issues in open-source game development?

Motivation. We quantify the prevalence of multi-PR issues in open source video game-related projects and examine their distribution across projects to determine whether having multiple PRs linked to a single issue report is an occasional phenomenon or a systematic practice concentrated in specific projects.
Results. Across 51,366 issues linked[1] to at least one pull request, 2,387 issues (4.8%) are resolved through multiple pull requests. While multi-PR issues are relatively uncommon overall, their prevalence varies substantially across projects, with a non-trivial subset exhibiting high rates of multi-PR resolution.

RQ2: What are the reasons for multi-PR issues?
Motivation. To understand the reasons why multi-PR issues primarily exist in open source video game projects, we analyze discussions linked to issue reports and corresponding PRs to uncover the underlying reasons that lead to multiple PRs being associated with a single issue.
Results. Our qualitative analysis reveals a taxonomy of seven reasons why multi-PR issues occur in open-source video game projects. One of the popular reasons is that most multi-PR issues stem from iterative improvement, with incomplete or buggy initial pull requests and follow-up extensions accounting for the majority of cases.

RQ3: How do resolution timelines and efforts differ between single-PR and multi-PR issues?
Motivation. While several studies [12, 15] have investigated methods to predict PR/issue resolution times, none of them explicitly focused on Multi-PR issues. If multi-PR issues reflect a distinct resolution process, they may also exhibit systematically different resolution timelines compared to single-PR issues. We therefore compare resolution times to assess the temporal impact of iterative, multi-PR issues vs. single-PR issues. We further compare the commit counts across the two groups of issues.
Results. In the context of video games, we find that multi-PR issues take substantially longer to resolve than single-PR issues, with a median resolution time four times higher and a greater likelihood of becoming long-running. We find that the number of commits linked to multi-PR issues is also substantially larger than that for single-PR issues.

Together, the results of our RQs provide a comprehensive picture of a previously underexplored phenomenon in game development. This work makes three primary contributions.

- We provide the first large-scale empirical characterization of multi-PR issues.
- We identify key factors associated with iterative issue resolution specifically in open-source game development, highlighting differences in collaboration, complexity, and resolution time compared to single-PR issues.
- We offer empirical insights that inform researchers and practitioners about more nuanced models of defect resolution in game development, accounting for multi-PR issues.

By revealing the prevalence and characteristics of multi-PR issues, our findings advance the understanding of defect resolution processes in games and lay the groundwork for future research on issue prioritization, effort estimation, and predictive models that explicitly account for iterative fixing behavior.

Data Availability. The dataset and the scripts that we used for our analyses are available online.[2]

2 Software Development and Game Development in OSS

2.1 Issue Resolution and Pull Requests in Open-Source Development

In open-source software platforms such as GitHub, issue reports can be created to keep track of defects, feature requests, or design concerns, while pull requests, i.e., PRs, have become the popular way of proposing, reviewing, and integrating code changes that are intended to address issue reports [36]. Issues and PRs can be explicitly linked, enabling traceability between issue reports and their corresponding fixes.[3]

Some prior studies [4, 27] on issue reports and pull requests implicitly model issue resolution as a one-to-one process. For example, Ren et al. [27] associated each issue with the pull request that closes the issue, and analyzed characteristics based on that closing PR. Such designs abstract away scenarios where multiple pull requests iteratively contribute to resolving the same issue. While this abstraction simplifies modeling and measurement, it does not necessarily reflect how developers iteratively diagnose and fix complex problems in practice.

In practice, issue resolution can be iterative. For example, Alshara et al. [2] observed that 32% of issue reports in Android projects in GitHub are associated with multiple pull requests, indicating that a substantial portion of issues require staged or incremental fixes. Besides, issue resolution in open-source projects can be complicated due to their collaborative and volunteer-driven nature. Contributors may join and leave sporadically, and fixes may be developed incrementally as understanding of the problem evolves.

Despite these implications, multi-PR issues have received limited explicit attention in prior research. Existing studies often aggregate PR-level metrics without distinguishing whether a PR represents a complete or partial resolution. As a result, the prevalence, characteristics, and impact of iterative issue resolution remain underexplored, which we systematically investigate in this study. Also, previous studies that discussed multi-PR issues have not investigated in depth the underlying reasons that lead to multiple PRs per issue. In our study, we investigate the prevalence, characteristics, and reasons behind multi-PR issues.

2.2 Characteristics of Game Development Relevant to Issue Resolution

While iterative development is common across software domains, games are particularly sensitive to small behavioral changes: a minor modification in physics, rendering, or gameplay logic can unintentionally affect balance, user experience, or performance.

[1] A PR is considered to be linked to an issue according to the definition provided by the OSSGameBench dataset [17].

[2] https://zenodo.org/records/18764556
[3] https://docs.github.com/en/issues/tracking-your-work-with-issues/using-issues/linking-a-pull-request-to-an-issue

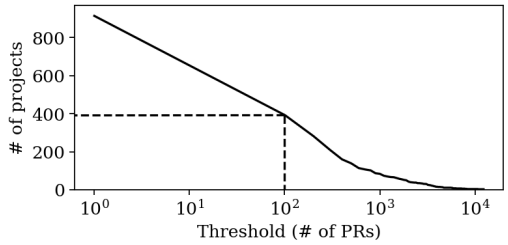


Figure 1: Threshold of PRs.

Consequently, fixes frequently require follow-up adjustments spanning multiple pull requests [25, 29, 34]. Not only that, gameplay-related defects may only manifest under specific interactions or usage patterns, making them difficult to reproduce and fix deterministically [26].

Beyond technical complexity, video games are often designed to sustain fun and long-term engagement through reward mechanisms (e.g., achievements, badges, unlockable levels, contests, and performance statistics) [6, 7, 10, 22, 35]. For example, Némery et al. [22] evaluated persuasive characteristics across multiple types of interfaces, including video games, by proposing and validating a criteria-based checklist. In their framework, persuasion refers to the capacity of an interface to shape, reinforce, or change users' attitudes and behaviors through design elements, such as progressive commitment through action sequences, reward mechanisms, and immersive engagement [22]. In practice, this emphasis on engagement can result in staged refinements to the source code of video games, where a single bug report is addressed through multiple successive PRs, each incrementally adjusting mechanics, balance, or user experience rather than delivering a complete fix in one PR.

Besides, many game repositories include not only playable games but also supporting engines and tools, introducing heterogeneity in development practices and resolution strategies [16]. These characteristics suggest that iterative, multi-PR issue resolution may be particularly prevalent in game development. However, without systematic empirical investigation, it remains unclear how common such issues are, how they differ from single-PR issues, and what implications they have for collaboration and development effort.

Understanding multi-PR issues is essential for building accurate empirical models of issue resolution in game development. Treating all issues as single-shot fixes risks underestimating development effort, mischaracterizing collaboration patterns, and biasing predictive models of resolution time or PR success. By explicitly identifying and analyzing multi-PR issues, researchers can better capture the iterative nature of real-world development and derive insights that are more faithful to developer practice. In this work, we investigate how iterative issue resolution unfolds in open-source game development.

3 Study Design

In this section, we present the design of our study. We begin with an existing dataset, OSSGameBench [16] by Marsad et al., which consists of 950 open-source repositories on GitHub. The dataset includes both playable video games and essential game development tools (e.g., game development frameworks). It contains issues, pull requests, comments, and commits from these projects, along with explicit mappings among these entities. The projects in this dataset have already been filtered to ensure that they have a reasonable development history. In fact, the author ensured each project contains a minimum threshold of 87 GitHub issues per project.

	Min	Median	Max	Total
Number of issues	11	214	12,483	280,952
Number of PRs	100	336	14315	358,857

Table 1: Statistics of the dataset

To ensure that the selected projects contain a sufficient number of PRs, we plot the threshold number of PRs against the number of candidate projects that satisfy each threshold (see Figure 1). Following Marsad et al. [16], we select the threshold that lies near the "knee" of the curve while avoiding overly conservative or overly restrictive cutoffs. Based on this criterion, we choose a threshold of 100 PRs. This selection results in 392 projects, which constitutes a sufficiently large sample to draw conclusions with over 95% confidence. Statistics of our dataset are shown in Table 1.

4 Study Results

In this section, we provide the approach and results for each research question.

4.1 RQ1: How common are multi-PR issues in open-source game development?

Approach. To quantify the prevalence of multi-PR issues, we group issues by project and issue number (an identifier to uniquely identify an issue report within a project). Then, we count the number of pull requests linked to each issue report. We classify an issue as a *multi-PR issue* if it is associated with more than one distinct pull request. To better understand the prevalence of multi-PR issues, we conduct the following analyses:

- **Cross-project analysis.** We examine the prevalence of multi-PR issues across the entire dataset by comparing the number of issues resolved by a single PR with those resolved by multiple PRs.
- **Project-level analysis.** We analyze how the prevalence of multi-PR issues varies across projects by computing, for each project, the proportion of issues that are associated with multiple PRs.

Results. Our cross-project analysis revealed that across the dataset, we identify 51,366 issues that are linked to at least one pull request. Among these, 2,387 issues (4.8%) are associated with multiple pull requests. While multi-PR issues represent a minority of cases, they constitute a non-trivial number of issues in absolute terms and reflect a distinct resolution pattern in which an issue is addressed through multiple pull requests rather than a single change. Such issues may involve iterative fixes, partial resolutions, or coordination across components, motivating a closer examination of their characteristics and resolution dynamics in our subsequent analyses.

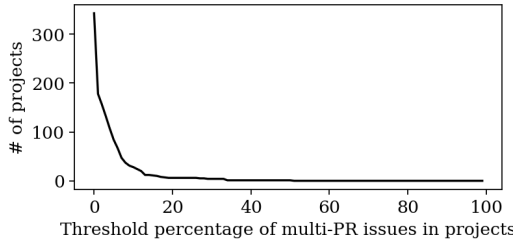


Figure 2: Number of projects exceeding thresholds of multi-PR issue prevalence.

At the project level, the median project exhibits a multi-PR issue rate of 1.4%. Figure 2 shows the number of projects whose proportion of multi-PR issues exceeds a given threshold percentage of multi-PR issues in projects. From this figure, we observe substantial heterogeneity in the prevalence of multi-PR issues. While most projects have relatively low proportions of multi-PR issues, a non-trivial proportion of projects shows much higher rates: 28 projects have more than 10% of their issues resolved through multiple pull requests, and an extreme subset of six projects exhibit multi-PR rates exceeding 20%. This variation indicates that iterative issue resolution is unevenly distributed across projects rather than being driven by a small number of isolated cases.

> Approximately one in twenty issues is resolved through multiple pull requests. While most projects exhibit modest rates of multi-PR issues, a non-trivial subset of projects shows highly iterative issue resolution.

4.2 RQ2: What are the reasons for multi-PR issues?

Approach. To understand why a single issue is addressed through multiple PRs, we construct a taxonomy of "multi-PR reasons" using 60 randomly selected issues that are each linked to more than one PR. These 60 issues contain 1,605 issue comments and 878 PRs.

For each issue report, we compile a structured evidence bundle which includes the issue title and description, discussion comments of the issue, and all linked PR artifacts (PR titles and descriptions, merge status, and PR discussion comments). To create this taxonomy, the author of the paper manually inspected each evidence bundle and produced an initial explanation for why a given issue is associated with multiple PRs. To improve reliability, an additional coder independently reviewed these explanations, and disagreements were resolved through discussion with the author. In parallel, we prompt an OpenAI Large Language Model (LLM) with the same evidence bundle and a fixed instruction template to produce an explanation of (i) why multiple PRs exist, (ii) a simplified "mental model" of the resolution process (including whether all PRs were merged, and why or why not), and (iii) any special observations grounded in the discussion. Next, the author, with another coder, jointly reviews the human- and LLM-generated explanations for all 60 issues and performs an open card sorting process to group similar rationales. This iterative sorting results in a taxonomy of seven distinct reasons that capture the main reasons behind multi-PR issue resolution. To support transparency and replication, we include all prompts, intermediate outputs (LLM explanations and category assignments), and the final taxonomy and labels in our replication package. This enables reproducing the taxonomy construction process.

Table 2: The extracted themes for multi-PR issues.

ID	Theme	Frequency (%)
(C1)	Incomplete/ buggy PR	29 (48.33%)
(C2)	Extended PR	12 (20.00%)
(C3)	Crowded PR	5 (8.33%)
(C4)	Abandoned PR	3 (5.00%)
(C5)	Accidental PR	3 (5.00%)
(C6)	Other	4 (6.67%)
(C7)	Unknown	4 (6.67%)

Results. Our open card sorting yields seven categories, as shown in Table 2. Each category in our taxonomy describes a reason for having multiple PRs linked to an issue.

(C1) Incomplete or buggy PR. The initial PR attempts to resolve the issue but is functionally incomplete, of poor quality, or introduces new bugs. Consequently, the PR is closed without being merged, and a subsequent PR is opened to provide a corrected or improved solution. For example, in the project `ppy/osu`, issue #30160 was addressed through multiple PRs. The first PR was closed without merging due to failing test cases, while another contributor independently developed a second PR that fully fixed the issue, which was later merged into the main codebase.

(C2) Extended PR. The issue is initially resolved by a PR that addresses the core problem (i.e., the PR is merged), but follow-up PRs are created to extend or refine the solution. These extensions may include additional features, edge-case handling, performance improvements, or quality improvements that go beyond the original fix. For instance, in the project `GameServerManagers/LinuxGSM`, issue #3422 requested support for faster multi-threaded backup compression. While the initial change added support for parallel compression, subsequent pull requests further extended the backup functionality by adding support for additional compression methods and improving dependency handling. Thus, rather than being resolved by a single self-contained fix, the issue evolved through multiple PRs that progressively expanded and refined the backup mechanism.

(C3) Crowded PR. A pull request attempts to address multiple issues in the PR. Because the PR bundles fixes for multiple issue numbers, it may complicate review and integration, leading to closing without merging. As a result, subsequent PRs are required to separately and cleanly resolve the original issue. For example, in issue #3269 from project `azerothcore/azerothcore-wotlk`, one PR introduced a large set of changes spanning several issues, prompting reviewers to suggest incremental fixes (e.g., "instead of doing everything at once, do it bit by bit"). This PR was ultimately closed without being merged, despite partially addressing the issue. A later

PR that focused exclusively on the focal issue was subsequently accepted and merged.

(C4) Abandoned PR. A pull request is created to address the issue, but is later abandoned without being merged, often due to lack of follow-up, unaddressed review feedback, contributor disengagement, or discovery of new information, which means this PR is no longer useful. The issue may or may not eventually be resolved through a subsequent PR. For example, in project `cuttle-cards/cuttle`, the first PR in issue #674 has a lot of review comments, and no changes were made to address them after several rounds, abandoning that PR.

(C5) Accidental PR. The comments acknowledge that a pull request in this issue was created accidentally or unintentionally (e.g., wrong branch or mistaken submission). This PR may be superseded by a follow-up PR. For example, in issue #1908 in project `quaver/quaver`, a developer noted, "Whoops, accidentally had commits of other branch in this." The initial PR was closed on the same day that the corrected PR was opened.

(C6) Other. This category covers the multi-PR issues that do not clearly fit into any of the five categories. These cases often involve atypical workflows, mixed motivations, or project-specific practices that cannot be classified under a single dominant reason. For example, in project `wolvenkit/wolvenkit`, issue #2129 involved multiple PRs with differing relevance to the issue: one PR was unrelated to the focal issue and therefore not merged, while another PR directly addressed the issue and was merged.

(C7) Unknown. Cases in which the reason a pull request was closed cannot be determined from available artifacts (e.g., issue discussions, PR comments, or commit history).

> Most multi-PR issues are driven by iterative improvement rather than exceptional breakdowns. Nearly half of the cases stem from an initial PR that is incomplete or buggy, while another fifth arise from follow-up PRs that extend or refine an existing solution. Only a handful of multi-PR issues are attributable to coordination challenges, such as crowded PRs that bundle multiple issues, abandoned PRs, or accidental submissions.

4.3 RQ3: How do resolution timelines and efforts differ between single-PR and multi-PR issues?

Approach. Building on our finding that multi-PR issues largely reflect iterative development rather than exceptional failures, we next examine how multi-PR issues affect issue resolution timelines. To this end, we examine whether issues resolved after the submission of multiple pull requests differ compared to single-PR issues in their resolution timelines and the resolution efforts in terms of the number of commits involved.

Resolution Timeline: We first define issue resolution time as the elapsed time (in days) between issue creation and issue closure. Because resolution time distributions are highly skewed and non-normal, we apply non-parametric statistical testing. Specifically, we use a one-sided Mann–Whitney U test to assess whether multi-PR issues tend to take longer to resolve than single-PR issues. The one-sided formulation reflects our directional hypothesis that iterative, multi-PR resolution processes may incur additional coordination and rework costs, resulting in longer resolution times.

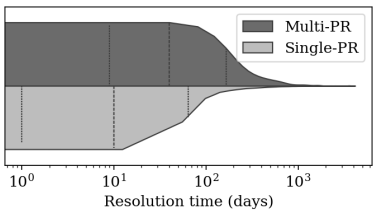


Figure 3: Distribution of issue resolution time in days.

To quantify the magnitude of observed differences, we report Cliff's delta (δ) as an effect size measure, which estimates the probability that a randomly selected multi-PR issue has a longer resolution time than a randomly selected single-PR issue. Following established guidelines, we interpret effect sizes as negligible, small, medium, or large. We visualize resolution time distributions for both groups using comparative plots to support interpretation and to illustrate distributional differences beyond summary statistics.

Resolution Effort. We define resolution effort as the total number of commits associated with all pull requests linked to an issue. As commit counts are similarly skewed, we apply the same non-parametric testing framework to compare the two groups (multi-PR issues and single-PR issues). Using a one-sided Mann–Whitney U test, we assess whether multi-PR issues involve more commits than single-PR issues, consistent with our expectation that iterative resolution processes require additional revisions and incremental refinements. Similar to our resolution-time analysis, we report Cliff's δ to quantify effect size, estimating the probability that a randomly selected multi-PR issue requires more commits than a randomly selected single-PR issue. We visualize commit count distributions for both groups using comparative plots to support interpretation and to illustrate distributional differences beyond summary statistics.

Results. Figure 3 shows the issue resolution times between single-PR and multi-PR issues. Overall, issues resolved through multiple pull requests exhibit longer resolution timelines than those resolved through a single pull request.

Resolution Timeline: The median resolution time for multi-PR issues is 40 days, substantially higher than that of single-PR issues (10 days). Moreover, the resolution-time distribution for multi-PR issues exhibits a heavier right tail, indicating a greater prevalence of long-running issues. A one-sided Mann–Whitney U test confirms that the difference in resolution times between the two groups is statistically significant ($p << \alpha = 0.05$). Although the corresponding Cliff's δ, 0.293, suggests a small effect size, this reflects the considerable overlap between the two distributions rather than the absence of a practically meaningful difference. In practice, multi-PR issues are systematically more likely to experience prolonged resolution timelines. Indeed, 33% of multi-PR issues require more than 100 days to be resolved, compared to only 20% of single-PR issues which require more than 100 days. A concrete example is issue #479 in project `Vocaluxe/Vocaluxe`. This issue remained open for

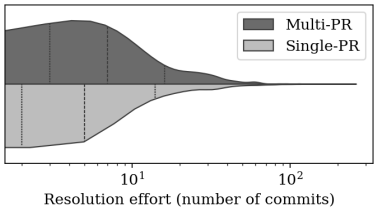


Figure 4: Distribution of the number of commits.

an extended period because an initial workaround was required to prevent system crashes, while the complete solution depended on coordinated changes across multiple subsystems (e.g., engine logic and user interface), unresolved design decisions regarding scoring semantics, and intermittent contributor availability. As a result, the issue only stabilized sufficiently for merging after being scoped down to basic support and aligned with a scheduled release milestone.

Resolution Effort: Figure 4 shows the distribution of commits across single-PR issues and multi-PR issues; the figure reveals noticeable differences in the number of commits associated with single-PR and multi-PR issues. Single-PR issues have a median of five commits (mean = 13.72), whereas multi-PR issues exhibit a higher median of seven commits (mean = 12.58). Although the means appear comparable, the distribution of single-PR issues shows substantially higher variability (standard deviation = 23.49) and a much longer right tail (max = 255 commits) compared to multi-PR issues (standard deviation = 14.47, max = 114 commits). A one-sided Mann–Whitney U test indicates that multi-PR issues tend to involve significantly more commits than single-PR issues ($p << 0.001$). The corresponding Cliff's δ, 0.166, indicates a small but non-negligible effect size, suggesting a systematic—though moderate—increase in commit-level activity for multi-PR issues.

One might expect multi-PR issues to accumulate more commits because they span multiple pull requests. However, the observed difference is moderate rather than extreme. The median increases from five to seven commits, and the effect size remains small but non-negligible. This suggests that multi-PR issues are not characterized by uncontrolled growth in development effort. Instead, additional pull requests appear to reflect structured iteration—incremental adjustments and refinements—rather than large bursts of additional code changes. In other words, multi-PR resolution does not necessarily imply disproportionately large implementation effort, but rather a more fragmented or staged integration process.

> Multi-PR issues represent a structurally different resolution process. They are substantially more likely to become long-running, yet their increased commit activity is moderate rather than explosive. Together, these findings suggest that multi-PR issues are characterized not by greater coding effort, but by prolonged, staged iteration involving refinement, coordination, and cross-component adjustments.

5 Implications

Below, we distill the implications of this study.

5.1 Implications for Developers and Project Maintainers

Developers should treat multi-PR issues as a distinct type of issue that requires multiple iterations of work rather than as ordinary bug-fix tasks. Although multi-PR issues account for only about 5% of all issues overall, their much higher prevalence in certain projects (exceeding 10–20%) suggests that they often arise from complex bugs, architectural coupling, or coordination across subsystems [32]. In the context of video game development, such issues frequently span engine logic, gameplay mechanics, asset pipelines, or user interface components [1], where fixes are difficult to isolate within a single change. Explicitly flagging and tracking such issues (e.g., using labels indicating iterative or multi-step resolution) can help teams recognize early that these issues are likely to require multiple iterations, enabling more realistic planning, better expectation management, and improved coordination during issue resolution.

Developers may need to consider the risk of duplicated or wasted effort when a single issue is addressed independently through multiple pull requests. In game development projects—where contributors may work on overlapping gameplay features, platform-specific code paths, or asset-related fixes—multi-PR issues can reflect parallel but partially redundant attempts to address the same underlying problem. This can lead to duplicated implementation effort, increased review overhead, or abandoned partial fixes that never reach production. Recognizing this pattern early and introducing lightweight coordination mechanisms, such as assigning ownership, consolidating efforts, or clarifying resolution strategies, can help reduce unnecessary duplication and improve the efficiency of issue resolution.

Multi-PR issues represent a disproportionate source of long-running issues that can threaten release schedules. Although multi-PR issues are relatively rare, a substantial fraction persist for extended periods, with one-third remaining unresolved for more than 100 days. In video game development, where releases are often tied to fixed launch windows, content updates, or seasonal events, such prolonged resolution timelines can pose a particular risk. Multi-PR issues may therefore contribute disproportionately to backlog aging and long-term maintenance burden. For developers and project maintainers, explicitly monitoring such issues can support earlier intervention—such as scope reduction, ownership assignment, or deferral to a future release cycle—to mitigate downstream delays.

5.2 Implications for Researchers

For researchers and tool builders, multi-PR issues highlight the limitations of single-shot resolution models in game development. The systematic difference in resolution timelines suggests that treating issue resolution as a one-PR process obscures important temporal and coordination dynamics, especially in game projects that involve tightly coupled systems, frequent iteration, and evolving design requirements. Future tools and empirical models for game development should explicitly account for iterative

resolution patterns, for example, by aggregating resolution effort across multiple pull requests, modeling cross-component dependencies, or identifying early signals of issues likely to enter prolonged multi-PR stages.

Researchers may apply our taxonomy beyond video games and extend it to other software domains. Although this study focuses on multi-PR issues in game development projects, the taxonomy we derive is not inherently game-specific. Many of the underlying mechanisms, such as iterative refinement, partial fixes, dependency-related adjustments, coordination across contributors, or evolving requirements, are common across software systems. Therefore, researchers can apply and validate our taxonomy in other domains (e.g., games for education [11, 18, 28], Augmented Reality applications [9], virtual reality applications [19], web applications, mobile apps [8], infrastructure tools, or AI systems) to examine whether similar multi-PR resolution patterns emerge [7, 22]. Such cross-domain replication studies would help assess the generalizability of our findings and identify domain-specific nuances. Ultimately, comparative analyses may reveal whether multi-PR issues are structurally tied to certain development contexts (e.g., asset-heavy systems, real-time constraints) or represent a broader phenomenon in collaborative software engineering.

6 Threats to Validity

In this section, we disclose the threats to the validity of our study.

6.1 Construct Validity

We rely on the existing OSSGameBench dataset [16] for our analysis. As a result, any limitations of the dataset may also affect the results of our analysis. For example, we identify multi-PR issues based on explicit links between issues and pull requests recorded in the OSSGameBench dataset [16]. Note that this dataset retrieves links between issue reports and PRs by using the GitHub API as well as mentions with keywords like "fixes" and "closes," followed by an issue number as suggested in prior work [2]. If pull requests are not correctly linked to issues (e.g., missing references, informal mentions in comments, or cross-repository links), some multi-PR issues may be misclassified as single-PR issues. Such cases would lead to an underestimation of the prevalence of multi-PR resolution. Therefore, the reported proportion of multi-PR issues should be interpreted as a conservative lower bound rather than an exact estimate. That said, OSSGameBench represents the most comprehensive dataset currently available for open-source video game development activities, containing explicit mappings among issues, pull requests, and commits. Given its scale (392 active repositories) and structured linking methodology, we believe that reliance on this dataset does not pose a substantial threat to the overall validity of our conclusions.

6.2 Internal Validity

Our taxonomy of reasons for multi-PR issues is derived from a manually analyzed sample of 60 issues. Although this may appear to be a small sample in terms of issue count, our analysis covers a substantial volume of qualitative data, including 3,601 pull request comments and 269 issue comments across these issues. While we deliberately select information-rich cases and employ a structured evidence bundle, the interpretation of issue discussions and pull request artifacts remains subject to researcher judgment. To mitigate this threat, we complement our manual analysis with LLM-generated explanations and iterative refinements.

Our study treats both games and game-related frameworks as equivalent, which may obscure differences in development practices and characteristics. Future studies may analyze these categories separately. Lastly, the resolution time is measured as the duration between issue creation and closure. This operationalization may not fully capture periods of inactivity or external delays unrelated to development effort, such as contributor availability, release scheduling, or project-specific workflow policies. As a result, longer resolution times for multi-PR issues may partially reflect contextual or organizational factors rather than technical complexity alone. To mitigate any biases, we complement the analysis of resolution time with another analysis on the number of commits. By analyzing both temporal and commit-level metrics, we triangulate our findings and characterize multi-PR resolution dynamics.

6.3 External Validity

We select projects with more than 100 PRs to ensure that our analysis focuses on projects with sufficient development activity to meaningfully study multi-PR resolution patterns. This selection results in a dataset of 392 projects. While this dataset represents a subset of the entire population of video game-related GitHub projects, and may not generalize to all such projects (e.g., smaller or less active ones), the inclusion of 392 projects constitutes a substantial sample of actively maintained repositories. As such, our findings provide robust insights into multi-PR issue resolution practices in open-source video game development.

7 Conclusion

This study is the first large-scale empirical study of multi-PR issue resolution in open-source game development. In fact, we analyze a dataset of 392 active game-related repositories, and systematically examined the prevalence, underlying causes, and resolution dynamics of issues that are resolved through multiple PRs.

Our results show that, while multi-PR issues constitute a minority (4.8%) of issues overall, they represent a non-trivial and unevenly distributed phenomenon across projects. A subset of repositories exhibits highly iterative resolution practices, with more than 10–20% of issues addressed through multiple PRs. Our qualitative analysis reveals that most multi-PR issues arise from iterative refinement, such as incomplete or buggy initial PRs and follow-up extensions. We further find that multi-PR issues exhibit substantially longer resolution times and systematically higher commit-level activity compared to single-PR issues, reflecting more iterative and effort-intensive resolution processes.

At least within the context of game development, issue resolution frequently unfolds as an incremental, multi-stage process spanning multiple development cycles. Abstracting this process as a single-PR resolution event risks underestimating the true engineering effort involved, obscuring coordination and collaboration dynamics, and biasing predictive models of defect resolution that assume a one-to-one issue-PR relationship.

References

[1] Vartika Agrahari, Shriram Shanbhag, Sridhar Chimalakonda, and A Eashaan Rao. 2023. A catalogue of game-specific anti-patterns based on GitHub and Game Development Stack Exchange. *Journal of Systems and Software* 204 (2023), 111789.

[2] Zakarea Alshara, Anas Shatnawi, Hamzeh Eyal-Salman, Abdelhak-Djamel Seriai, and Maad Shatnawi. 2022. Pi-link: A ground-truth dataset of links between pull-requests and issues in github. *IEEE Access* 11 (2022), 697–710.

[3] Usman Ashraf, Christoph Mayr-Dorn, Alexander Egyed, and Sebastiano Panichella. 2020. A mixed graph-relational dataset of socio-technical interactions in open source systems. In *Proceedings of the 17th International Conference on Mining Software Repositories*. 538–542.

[4] Peter Bludau and Alexander Pretschner. 2022. PR-SZZ: How pull requests can support the tracing of defects in software repositories. In *2022 IEEE international conference on software analysis, evolution and reengineering (SANER)*. IEEE, 1–12.

[5] Claudio Di Sipio, Andrea D'Angelo, Riccardo Rubei, and Cristiano Politowski. 2025. On the Need for Reproducibility Guidelines for Open-Source Games: A itch. io Case Study. *Conference on Games* (2025).

[6] Lucas Ferreira, Leonardo Pereira, and Claudio Toledo. 2014. A multi-population genetic algorithm for procedural generation of levels for platform games. In *Proceedings of the Companion Publication of the 2014 Annual Conference on Genetic and Evolutionary Computation*. 45–46.

[7] Carlo A Furia and Andrea Mocci. 2025. What Makes a Level Hard in Super Mario Maker 2?. In *2025 IEEE Conference on Games (CoG)*. IEEE, 1–8.

[8] Matthaios Gerakis and Christina Volioti. 2021. A mobile educational application for enhancing cognitive and language skills of children with disabilities. In *Interactive Mobile Communication, Technologies and Learning*. Springer, 431–442.

[9] Ruizhen Gu, José Miguel Rojas, and Donghwan Shin. 2025. Software testing for extended reality applications: A systematic mapping study. *Automated Software Engineering* 32, 2 (2025), 56.

[10] Emanuela Guglielmi, Gabriele Bavota, Nicole Novielli, Rocco Oliveto, and Simone Scalabrino. 2025. Is it Really Fun? Detecting Low Engagement Events in Video Games. In *2025 IEEE/ACM 22nd International Conference on Mining Software Repositories (MSR)*. IEEE, 564–575.

[11] Hana Hadzifejzovic, Colt Morell, Tom Scholberg, Barbara Perry, Ahmed Jama, David Obrien, Kim McCalpin, Andrew McKenzie, Alvaro Uribe-Quevedo, Andrew Hogue, et al. 2023. Development of a Safe Gaming and Anti-Hate Serious Game. In *2023 IEEE Gaming, Entertainment, and Media Conference (GEM)*. IEEE, 1–6.

[12] David Kavaler, Sasha Sirovica, Vincent Hellendoorn, Raul Aranovich, and Vladimir Filkov. 2017. Perceived language complexity in GitHub issue discussions and their effect on issue resolution. In *2017 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 72–83.

[13] Haitao Li, Haiyang Wang, Jiangchuan Liu, and Ke Xu. 2013. Video requests from online social networks: Characterization, analysis and generation. In *2013 Proceedings IEEE INFOCOM*. IEEE, 50–54.

[14] Gabriel Lovreto, Andre T Endo, Paulo Nardi, and Vinicius HS Durelli. 2018. Automated tests for mobile games: An experience report. In *2018 17th Brazilian Symposium on Computer Games and Digital Entertainment (SBGames)*. IEEE, 48–488.

[15] Chandra Maddila, Chetan Bansal, and Nachiappan Nagappan. 2019. Predicting pull request completion time: a case study on large scale cloud services. In *Proceedings of the 2019 27th acm joint meeting on european software engineering conference and symposium on the foundations of software engineering*. 874–882.

[16] Faiz Marsad and Nimmi Weeraddana. 2026. OSSGameBench: A Large-Scale Dataset of Contributor Activities in the Open-Source Video Games. In *Proc. of the International Conference on Mining Software Repositories (MSR)*.

[17] Faiz Marsad and Nimmi Weeraddana. 2026. OSSGameBench: A Large-Scale Dataset of Development Activities in Open-Source Video Games. (2026).

[18] Michael A Miljanovic and Jeremy S Bradbury. 2023. Engineering Adaptive Serious Games Using Machine Learning. In *Software Engineering for Games in Serious Contexts: Theories, Methods, Tools, and Experiences*. Springer, 117–134.

[19] Joss Kingdom Moo-Young, Andrew Hogue, and Veronika Szkudlarek. 2021. Virtual materiality: Realistic clay sculpting in vr. In *Extended Abstracts of the 2021 Annual Symposium on Computer-Human Interaction in Play*. 105–110.

[20] Emerson Murphy-Hill, Thomas Zimmermann, and Nachiappan Nagappan. 2014. Cowboys, ankle sprains, and keepers of quality: How is video game development different from software development? *International Conference on Software Engineering* (2014).

[21] Andres Navarro, Juan Vicente Pradilla, and Octavio Rios. 2012. Open source 3D game engines for serious games modeling. In *Modeling and Simulation in Engineering*. Vol. 6. IntechOpen.

[22] Alexandra Némery and Eric Brangier. 2014. Set of Guidelines for Persuasive Interfaces: Organization and Validation of the Criteria. *Journal of Usability Studies* 9, 3 (2014).

[23] Ciprian Paduraru. 2026. A state-aware, hierarchical deep learning framework for automated visual glitch detection in games. *Engineering Applications of Artificial Intelligence* 166 (2026), 113497.

[24] Jihun Park, Miryung Kim, Baishakhi Ray, and Doo-Hwan Bae. 2012. An empirical study of supplementary bug fixes. In *2012 9th IEEE Working Conference on Mining Software Repositories (MSR)*. IEEE, 40–49.

[25] Cristiano Politowski, Lisandra M Fontoura, Fabio Petrillo, and Yann-Gaël Guéhéneuc. 2018. Learning from the past: A process recommendation system for video game projects using postmortems experiences. *Information and Software Technology* 100 (2018), 103–118.

[26] Cristiano Politowski, Fabio Petrillo, and Yann-Gaël Guéhéneuc. 2021. A survey of video game testing. In *2021 IEEE/ACM International Conference on Automation of Software Test (AST)*. IEEE, 90–99.

[27] Yang Ren, Gregory Gay, Christian Kästner, and Pooyan Jamshidi. 2020. Understanding the nature of system-related issues in machine learning frameworks: An exploratory study. *arXiv preprint arXiv:2005.06091* (2020).

[28] Maria Andréia F Rodrigues, Thiago RC de Oliveira, Daniel L de Figueiredo, Edilson O Maia Neto, Alexandre AA Akao, Guilherme HM de Lima, Vitoria LN Silva, and Anderson L Karl. 2022. An interactive story decision-making game for mental health awareness. In *2022 IEEE 10th International Conference on Serious Games and Applications for Health (SeGAH)*. IEEE, 1–8.

[29] Walt Scacchi and Kendra M Cooper. 2015. Research challenges at the intersection of computer games and software engineering. In *Conference on Foundations of Digital Games*.

[30] Mohammad Reza Taesiri, Finlay Macklon, Sarra Habchi, and Cor-Paul Bezemer. 2024. Searching bug instances in gameplay video repositories. *IEEE Transactions on Games* 16, 3 (2024), 697–710.

[31] Andrew Truelove, Eduardo Santana de Almeida, and Iftekhar Ahmed. 2021. We'll fix it in post: What do bug fixes in video game update notes tell us?. In *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*. IEEE, 736–747.

[32] Gabriel C Ullmann, Cristiano Politowski, Yann-Gaël Guéhéneuc, Fabio Petrillo, and João Eduardo Montandon. 2022. Video game project management anti-patterns. In *Proceedings of the 6th International ICSE Workshop on Games and Software Engineering: Engineering Fun, Inspiration, and Motivation*. 9–15.

[33] Simon Varvaressos, Kim Lavoie, Sébastien Gaboury, and Sylvain Hallé. 2017. Automated bug finding in video games: A case study for runtime monitoring. *Computers in Entertainment (CIE)* 15, 1 (2017), 1–28.

[34] Nimmi Rashinika Weeraddana, Sarra Habchi, and Shane McIntosh. 2025. Crash report prioritization for large-scale scheduled launches. In *2025 IEEE/ACM 47th International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*. IEEE, 389–400.

[35] Kaijie Xu and Clark Verbrugge. 2025. A database-driven framework for 3D level generation with LLMs. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, Vol. 21. 367–376.

[36] Xunhui Zhang, Yue Yu, Georgios Gousios, and Ayushi Rastogi. 2022. Pull request decisions explained: An empirical overview. *IEEE Transactions on Software Engineering* 49, 2 (2022), 849–871.