An Empirical Evaluation of Maintenance Practices in Open-Source Gaming Software: Analyzing the Correlation Between Update Frequency, Code Stability, and Player Sentiment

Connor Chapman
Colorado State University
Fort Collins, CO, USA
c.chapman@colostate.edu

Jocelyn Villegas
Colorado State University
Fort Collins, CO, USA
jsuzette@colostate.edu

Kyle Pham
Colorado State University
Fort Collins, CO, USA
phamky@colostate.edu

Ashley Gotch
Colorado State University
Fort Collins, CO, USA
agotch@colostate.edu

1 INTRODUCTION

Despite the $180 billion annual market for the video game industry, empirical research dedicated to software engineering for gaming software remains limited [3]. Open-source gaming projects are unique, combining entertainment value, real-time performance constraints, and community-driven development that distinguishes them from traditional software systems [14]. Understanding how maintenance practices relate to software quality and player engagement is critical to improving development in this growing ecosystem.

Current software engineering research focuses predominantly on conventional business applications, leaving a significant gap in understanding gaming-specific challenges and quality metrics [3]. Studies of update patterns in general open-source projects fail to account for gaming software's unique characteristics, such as the necessity of the "fun factor," real-time performance requirements, and community feedback loops prioritizing player satisfaction [14]. There are no frameworks for correlating technical repository metrics with player engagement indicators, despite gaming communities generating rich data streams, including user reviews and performance feedback.

This gap creates problems for open-source gaming projects because developers lack evidence-based guidance for optimizing update strategies. Without understanding the correlation between update frequency, software stability, and player engagement, projects risk alienating communities through poorly timed or rushed releases or losing player interest due to insufficient updates.

Previous research examined update frequency in mobile applications and general open source projects with mixed results on frequency and quality relationships [6]. Gaming studies have identified critical success factors and analyzed bug fix patterns in commercial games [1] [20]. However, these studies focused on proprietary games, analyzed general open-source projects without a gaming context, or examined gaming processes without correlating update patterns to user sentiment and code metrics.

This study conducts an empirical analysis of open source gaming repositories to investigate the relationships between update frequency, software quality metrics, and player engagement indicators. We examine repository data alongside community engagement measures to answer:

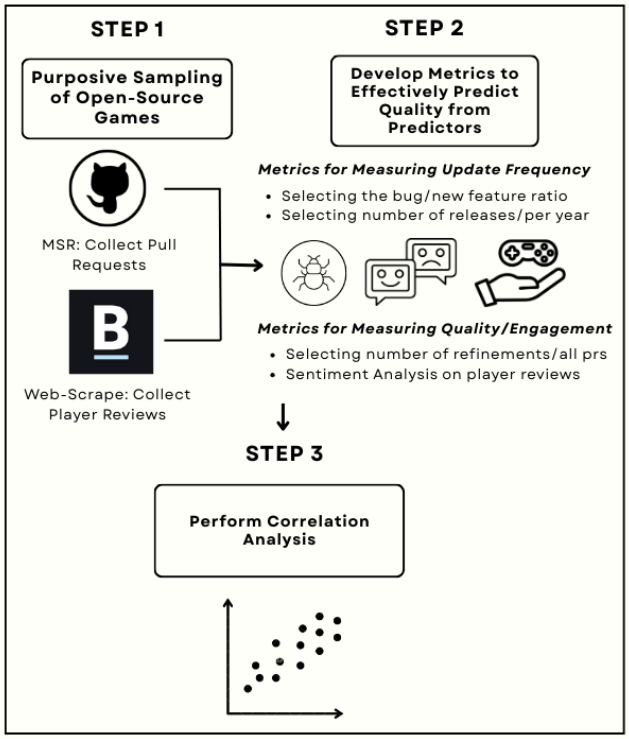

Figure 1: Research Design Process

RQ. *How does the frequency of game updates, technical quality metrics such as bug-to-feature ratios, overall code stability, and player sentiment correlate with each other?*

2 METHOD

Figure 1 presents an overview of the research design. We followed a multi-phase structure: data collection (sample filtering & data cleaning), metric analysis (update frequency & technical quality), and sentiment analysis (trained RoBERTa model on user reviews). Finally, we performed the Spearman correlation test on our derived

Game Title	Review
Daggerfall Unity	i can see why people would enjoy daggerfall, but it really is not all that great gameplay wise.
Battle for Wesnoth	Fantastic and surprisingly deep tactics game with great campaigns and cool user content. And it's free!
Luanti	As much as I want this game to succeed, it's doing something wrong, and I don't know what that is...
SuperTux	peak fiction
...	...

Table 1: Backloggd Player Reviews Preview

label	frequency	percent
Bugfix	2005	10.225939715407764
Feature	1306	6.660886418115979
Maintenance	1202	6.130463609935227
Trivial	1197	6.104962513388076
@ Client / Audiovisuals	1116	5.691844749324221
@ Documentation	587	2.9938287346355894
@ Server / Client / Env.	526	2.682715356760341
Rebase needed	407	2.0757892589381344
@ Build	373	1.9023818024175039
Performance	349	1.7799765389911766
Non-trivial	16	0.081603509
...	...	...

Table 2: Unclean PR Labels Preview (Luanti Dataset)

metrics and then used Kendall Tau correlation analysis to verify our findings.

2.1 Dataset Overview

This investigation utilizes data from two sources: GitHub repositories for code metrics (update frequency & technical quality) and the game review site Backloggd [4] for player sentiment. GitHub was chosen as it is the most widely used platform for open-source projects, and Backloggd for its focus on niche open-source game discussions. Utilizing both sources directly supports our RQ by comparing quantitative code metrics with user sentiment.

The dataset is a filtered list of seven open-source games. Each required more than one year of activity and more than 50 commits for sustained development. We mined a total of 74,438 pull requests (PRs) and 282 player reviews across the lifespans of the selected projects. Sampling focused on projects with an active codebase and a presence with user reviews on Backloggd.

This scope allows us to observe update patterns to evaluate correlations between our metrics, player sentiment, and focus on projects with sufficient activity that are within our domain focus. All data was collected in September 2025.

2.2 Data Collection Procedure

Data was collected directly from GitHub using its REST API and custom Python scripts, utilizing the PyGitHub library [15]. These scripts extracted all PR metadata (timestamps, authorship, labels, and release tags) to calculate update frequency and bug-fixing activity.

To capture sentiment, we collected all player reviews for the seven selected games from Backloggd [4]. A web-scraping script using Python libraries Selenium [16] and BeautifulSoup [5] was used to collect and format the review text. Game titles were manually matched between GitHub and the review site (Table 1).

This approach follows methods used in previous mining software repository studies [6, 20], and ensures that both technical quality measures and player sentiment are represented in the dataset.

2.3 Data Filtering

To ensure reliability, we used purposive sampling; only repositories with at least one year of activity and a minimum threshold of 50 commits were included, helping to filter out inactive or short-lived experimental projects. Forked repositories were excluded from the sample, while their corresponding upstream sources were retained. Repositories that lack essential metadata, such as commit timestamps or developer-generated issue labels, were not included to maintain data consistency and would not provide the information needed regarding code metrics. In addition, automated commits and bot-generated pull requests were flagged for awareness but not discarded, as they represent legitimate forms of project maintenance.

These filtering steps ensured that the final data set captured active and analyzable projects with consistent historical activity [6].

PR Label Standard	Description/Key Term Identification
Bug Fix	PRs that refer to bug fixes, patches to resolve bugs, or regression.
Feature	PRs that create new features or are adding some sort of new element.
Refine	PRs that are refining, refactoring, enhancing, or optimizing any current code.
Other	PRs that are doing none of the above.

Table 3: Mapping PR Label Data

Title	Bug Fix	Feature	Refine	Other	Total
Daggerfall Unity	107	81	14	128	330
Luanti	2279	1314	1028	14986	19607
SuperTux	211	147	14	1129	19937
OpenRA	976	5	271	7441	8693
Battle for Wesnoth	166	181	252	7037	7636
OpenTTD	548	0	109	955	16329
TripleA	2	0	0	1904	1906

Table 4: Cleaned PR Labels

2.4 Data Processing

2.4.1 Pull Request Data Cleaning. Project identifiers were standardized as ("owner/name"). The key cleaning step was the qualitative normalization of PR labels, as developer-created labels are unique to each codebase. This was done by manually categorizing the labels from our pull request mining results.

We sorted the original labels into standardized categories of "feature," "bug," and "refinement" and created a new data set from our Mining Software Repository (MSR) step as seen in Table 3. The process of mapping these qualitative traits by hand can be observed in the comparison between Table 2 and Table 4.

2.4.2 Review Data Preprocessing for Sentiment Analysis. Player reviews were preprocessed for sentiment analysis.

Data Cleaning involved removing null text, filtering entries shorter than 10 characters, and retaining all languages (compatible with RoBERTa's byte-level tokenization).

Text Normalization was minimal to preserve authentic player language. This included converting all text to UTF-8 encoding and removing non-printable characters. Unlike traditional Natural Language Processing, we preserved capitalization, punctuation, and special characters, as these features carry essential sentiment information (e.g., excessive exclamation marks can indicate a strong emotion).

Label Generation and Balancing Since Backloggd reviews lack sentiment labels, we used an existing annotated dataset of about 17, 494 Steam game reviews with binary sentiment labels (positive: 1 & negative: 0) to train our sentiment classification model [7]. The training data exhibited moderate class imbalance with 70.0% positive and 30.0% negative reviews, which reflected the natural distribution of gaming reviews. To address this imbalance during model training, we implemented weights in the loss function to assign more "weight" to negative reviews.

Splitting the Data The dataset was then divided into stratified subsets of 70% training, 15% validation, and 15% testing to maintain an even class distribution. This ensured that each subset maintained the original class distribution, providing reliable performance estimates during model evaluation.

Handling Missing & Noisy Data Reviews with ambiguous sentiment were retained in the training set, as they represent realistic examples the model would encounter. We also flagged but did not remove reviews containing common gaming slang, profanity, or sarcasm.

2.5 Dataset Characteristics

The dataset includes free, open-source games from multiple genres (RPG, action, racing, sandbox, &, strategy) that are actively maintained and have Backloggd reviews.

A bias exists towards projects with higher activity due to the filtering criteria (more than 50 commits), potentially comparing projects with vastly different developer team sizes. While they could both have an acceptable GitHub presence, a team of more developers will have more resources to update features than a smaller team. Additionally, the necessity of sufficient reviews for player sentiment analysis introduces a popularity bias, favoring more well-known open-source games.

Variable	Metric
x1 - Update Frequency	public releases (log-normalized)
x2 - Refinement Ratio	refinements/all prs
x3 - Bug Ratio	bug fix prs/all prs
x4 - Feature Ratio	feature prs/all prs

Table 5: Variable Metrics

Title	sentiment_label	sentiment	confidence
Daggerfall Unity	1	positive	0.74955714
Daggerfall Unity	0	negative	0.946025
Luanti	1	positive	0.9572593
Luanti	1	positive	0.98474103
OpenRA	1	positive	0.98531866
...	...	...	...

Table 6: Backloggd Player Sentiment Analysis Data Preview

2.6 Reproducibility

All datasets, analysis code, and repositories used in this study are publicly available to ensure reproducibility:

- Game Review Site:
  - BackLoggd [4]
- Open-Source Game Repositories:
  - DaggerFall Unity [8]
  - Luanti [11]
  - SuperTux [18]
  - OpenRA [12]
  - Battle For Wesnoth [21]
  - OpenTTD [13]
  - TripleA [19]
- Data Collection & Analysis Scripts:
  - Replication Package [7]

2.7 Data analysis

The analysis tests the various parallels between our created metrics that represent software maintenance properties, such as update frequency, refinements, bug fixes, and player sentiment.

Spearman correlation analysis was performed and validated with the Kendall Tau correlation analysis to demonstrate the interrelationships between our metrics. Details about our metrics developed to analyze these statistical associations can be seen in Table 5.

We decided on these montonic correlation analyses because they do not assume linearity between variables and emphasizes the strength of their correlation.

2.8 Sentiment Model Characteristics

2.8.1 RoBERTa Architecture and Pre-training. We used the RoBERTa-base (Robustly Optimized BERT Pretraining Approach) as the foundation for our sentiment analysis model. RoBERTa is a transformer-based language model consisting of 12 encoder layers with 124 million parameters, which features 768-dimensional hidden states and 12 attention heads that process sequences up to 512 tokens using byte-level Byte Pair Encoding tokenization [10].

RoBERTa was not originally pre-trained on user reviews or gaming-specific content. Therefore, fine-tuning was essential to adapt the model to gaming review sentiment analysis. We fine-tuned RoBERTa on our training dataset of 17,494 annotated Steam game reviews, adding a linear classification head for binary sentiment prediction (positive: 1, negative: 0).

Training Configuration We trained using the AdamW optimizer, a learning rate of 2×10^{-25}, batch size 16, weight decay of 0.01, and linear warmup over 500 steps. Training ran for up to 10 epochs with early stopping (patience of 3 evaluation steps based on validation of F1 score). The model was evaluated every 200 training steps, and then the checkpoint with the highest F1 validation was kept as the final model.

Tokenization Review text was tokenized using RoBERTa's byte-level BPE tokenizer with a maximum sequence length of 512 tokens. Reviews were truncated if they were longer and padded if slower, with special tokens ([CLS], [SEP]) added automatically.

2.8.2 Model Performance and Application. The fine-tuned model achieved strong performance on the test set of 2,625 reviews, with the median prediction confidence of 0.944, indicating well-calibrated sentiment classifications. We then applied this trained model to all 281 Backloggd reviews, processing them in batches of 32 and generating binary labels (positive/negative) and continuous probability (0-1 scales). The positive class probability was used as our continuous sentiment measure for regression analysis, providing a nuanced metric of player perception. By training our own RoBERTa model rather than using a pre-trained classifier, we ensured the model captured gaming-specific language patterns, terminology, and sentiment expressions unique to player reviews. A preview of these sentiment analysis results can be seen in Table 6.

3 ANALYSES AND RESULTS

To answer RQ1, we conducted two correlation analyses to examine how software development factors - specifically bug ratio, refinement ratio, feature ratio, and update frequency - correlate with one another.

3.1 Player Engagement and Satisfaction

To understand player engagement and satisfaction, we analyzed Backloggd user reviews for all seven open-source games in our sample. We collected a total of 281 player reviews across the repositories, with individual games receiving between 1 and 67 reviews each. We trained a RoBERTa transformer model specifically for sentiment analysis of video game reviews. The model was trained to quantify player sentiment on a scale from 0 (most negative) to 1 (most positive) whose average probability of positive sentiment served as a variable for correlation analysis.

Sentiment analysis revealed a substantial variation in player satisfaction in all games observed. The average sentiment scores ranged from 0.629 (Luanti) to 0.984 (TripleA), with an overall mean of 0.796 (SD = 0.225) across all reviews. This distribution indicates that while most open-source games in our sample received generally positive player feedback, there exists considerable variance in how players perceive these games. Overall, 248 (88.3%) were classified as positive, while 33 reviews (11.5%) were classified as negative, with a median confidence score of 0.944 in the sentiment predictions.

Game	Reviews	Avg Sentiment	Std Dev	% Pos.
TripleA	1	0.984	0.000	100.0%
Battle for Wesnoth	30	0.889	0.156	93.3%
SuperTux	24	0.831	0.194	95.8%
NetHack	31	0.817	0.226	87.1%
OpenTTD	67	0.813	0.221	91.0%
Daggerfall Unity	43	0.775	0.311	86.0%
Luanti	10	0.629	0.361	70.0%

Table 7: Per-Game Sentiment Analysis Results

Table 7 presents the sentiment statistics per game. Battle for Wesnoth achieved the highest average sentiment score (0.889, SD = 0.156) with 93.3% positive reviews and SuperTux (0.831, SD = 0.194) with 95.8% positive reviews. These games demonstrated consistent positive feedback regarding gameplay mechanics, community engagement, and overall entertainment value. Conversely, Luanti received the most moderate sentiment score (0.629, SD = 0.361) with only 70% positive reviews, with player reviews highlighting concerns about game stability, incomplete features, or inconsistent update quality.

There is a variation in standard deviations across games that is important to note. Luanti exhibited the highest variability (SD = 0.361), which suggests polarized player opinions, while SuperTux showed the most consistent sentiment (SD = 0.194). Games with larger review counts (OpenTTD: 67 reviews) maintained relatively stable sentiment scores, suggesting that their established player communities have reached consensus on long-term user sentiment.

Review content analysis revealed that players frequently mentioned specific technical aspects in their feedback. Among the negative reviews, common themes include difficult learning curves, technical bugs, and incomplete implementation of features. However, highly positive reviews praised innovative gameplay mechanics, active development communities, and continuous content updates. This correlation between technical development activities and player sentiment provides the basis for our correlation analysis in Section 3.3.

These technical metrics from our repository analysis provide variables for our correlation models, enabling us to examine how development practices are related to player sentiment, measured in Section 3.1.

3.2 Correlation Analysis Between Metrics and Player Sentiment

To evaluate the relationship between technical development metrics and player sentiment, we performed a Spearman correlation test in Figure 2 on our derived metrics to show the strength of the relationships in Table 5. To ensure the validity of the strengths in these relationships, another robust monotonic test is needed. A Kendall Tau test in figure 3 is performed on the same data to confirm the relationships in the Spearman test by offering less biased estimates within the small data sets, and is more stable to outliers since it is based on concordant and discordant pairs.

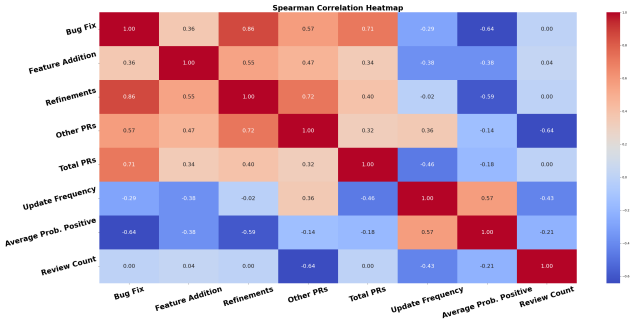

Figure 2: Spearman correlation heatmap

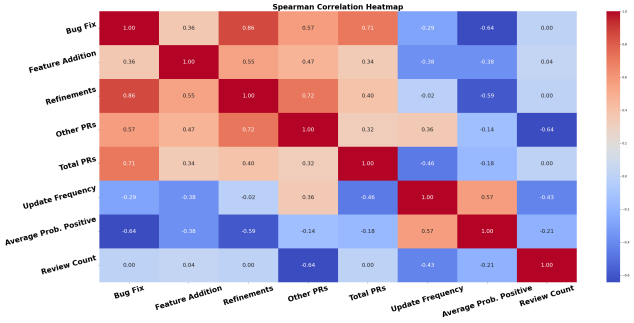

Figure 3: Kendal Tau correlation heatmap

The resulting tests show a strong association between technical metrics to themselves. This relationship shows that development teams focusing on technical variables observe positive correlations between them, particularly when the team prioritizes pull requests. Overall, a negative relationship is seen between technical metrics and the average probability of a positive player sentiment. These results show that games with a higher number of releases on GitHub (update frequency) have a stronger correlation with the average positive sentiment of user reviews. The remaining technical variables have an adverse relationship with the average sentiment of the players, indicating that more bug fixes, features, refinements, and general pull requests diminish the average probability of a positive sentiment.

4 DISCUSSION AND LIMITATIONS

4.1 Discussion of Findings

The amount of releases and player sentiment of a game are the strongest positively related attributes. This finding suggests that games with more frequent releases also tend to receive more positive player feedback. These results reinforce previous studies, emphasizing that players perceive quality through the visibility and consistency of updates. Truelove found that regular post-release updates and transparent communication through patch notes contribute to sustained user engagement and trust [20]. Similarly, Aleem found that active iteration and feature development align with success factors in game development, and can correlate with stronger sentiment [2].

In contrast, the refinement variable showed a moderately weak negative association with sentiment, which could reflect the tension between technical improvements and user sentiment value. While code refactoring and optimization are necessary for long-term maintainability, they often occur behind the scenes and do not produce immediate visible improvements for players. Stroggylos and Spinellis similarly noted that overemphasis on internal maintenance can divert developer attention away from user-facing progress, which can affect user satisfaction [17].

The bug-fix variable also exhibited a negative coefficient relationship when compared with player sentiment, and suggests that frequent bug-fix activity alone does not strongly correlate with user sentiment. Truelove found that while bug fixes are critical for stability, they often go unnoticed by players unless they resolve highly visible issues [20]. In our data, this pattern may explain why projects with higher proportions of bug-related commits did not necessarily achieve higher sentiment scores. Players appear to value progress and new experiences more than background maintenance.

Finally, our study investigates the use of user sentiment as a potential influence on software quality, reinforcing Leopairote's work on using opinion mining to compute a "quality in use score" from user reviews [9].

In summary, effective maintenance practices in open-source games must balance technical excellence with player-facing progress and community sentiment to sustain engagement and long-term success.

4.2 Threats to Validity

External Validity. Our sample of seven open-source games is small and limited to a single domain. Consequently, the results emphasize exploratory pattern recognition and are not generalizable to the broader population of all open-source games or commercial gaming development processes.

Internal Validity. The reliance on games reviewed on Backloggd and with sufficient GitHub activity introduces a selection bias. If an open-source game was not reviewed on Backloggd and had minimal GitHub activity, then the repository was omitted from our sample. Game popularity could be a confounding variable in this model since more well-known games would be more likely to meet both of the criteria.

Additionally, we developed a normalization of labels across all of the repositories mined. We are working with the assumption that developers routinely and correctly utilize the labels they created for issues and pull requests across their repository. If the labels are non-reflective of the true work, our technical metrics can become compromised.

Construct Validity. Because we are analyzing the correlation and relationships between technical code metrics and player sentiment, it is important to note that players' perceived quality or user satisfaction is not measured or observed directly. This could potentially introduce face validity where player sentiment can only appear to accurately represent the full domain of player satisfaction. A key threat to construct validity is construct underrepresentation. The full domain of user satisfaction encompasses numerous factors such

as long-term engagement, emotional connection, and more that are not directly observed by the sentiment expressed in reviews. Therefore, this metric only partially represents the full scope of player satisfaction.

Conclusion Validity. Because our study included only seven open-source games, the sample size is small, which limits the statistical power of our analysis. This smaller sample size thus reduces the ability to detect a true underlying association, even if one exists in the population. Although development activity metrics are associated with player sentiment in some ways, they do not generalize well enough to forecast user perception trends, and only demonstrate existing correlations. Therefore, our findings could be validated in larger datasets.

4.3 Implications for Practice

These results reinforce the significance of maintaining consistent and methodical development practices and the potential benefits of striking a balance between technical refinement and visible game enhancement.

Game developers can benefit from adopting frequent update practices that prioritize visible new features along with stable releases. This approach reflects good technical metrics, such as code quality and stability, but can also serve as a signal to the player community that the project is actively maintained and evolving, thus enhancing player sentiment.

Project managers can benefit from data-driven decision-making by proactively using correlation-based modeling as a benchmarking tool to analyze a game's development progress and priorities based on current player sentiment and technical code metrics.

4.4 Implications for Researchers

Overall, our findings contribute to the growing body of software engineering research suggesting that user sentiment can not be evaluated solely by traditional repository metrics.

Our method provides future researchers with a guideline for correlation-based analysis of open-source software using Spearman correlation, Kendall Tau validation, technical code metrics, and player sentiment.

REFERENCES

[1] Saiqa Aleem, Luiz Fernando Capretz, and Faheem Ahmed. 2018. Critical Success Factors to Improve the Game Development Process from a Developers Perspective. *arXiv preprint arXiv:1801.04293* (2018). https://doi.org/10.48550/arXiv.1801.04293

[2] Saiqa Aleem, Luiz Fernando Capretz, and Faheem Ahmed. 2021. User Requirements for Software Game Process: An Empirical Investigation. *arXiv preprint arXiv:2110.03764* (2021). https://doi.org/10.48550/arXiv.2110.03764

[3] Apostolos Ampatzoglou and Ioannis Stamelos. 2010. Software Engineering Research for Computer Games: A Systematic Review. *Information and Software Technology* 52, 9 (2010), 888–901. https://doi.org/10.1016/j.infsof.2010.05.004

[4] Backloggd. [n. d.]. Backloggd, Discover, Collect, and Analyze your Games. https://backloggd.com/ Accessed on October, 2025.

[5] BeautifulSoup. [n. d.]. BeautifulSoup. https://beautiful-soup-4.readthedocs.io/en/latest/.

[6] Fabio Calefato, Marco Aurelio Gerosa, Giuseppe Iaffaldano, Filippo Lanubile, and Igor Steinmacher. 2022. Will you come back to contribute? Investigating the inactivity of OSS core developers in GitHub. *Empirical Software Engineering* 27, 3 (2022), 76. https://arxiv.org/abs/2103.04656

[7] Connor Chapman, Jocelyn Villegas, and Kyle Pham. 2025. Repo Miner. https://github.com/ConnorChap/RepoMiner.

[8] Interkarma. [n. d.]. daggerfall-unity. https://github.com/Interkarma/daggerfall-unity Accessed on September, 2025.

[9] Warit Leopairote, Athasit Surarerks, and Nakornthip Prompoon. 2013. Evaluating software quality in use using user reviews mining. (2013). https://doi.org/10.1109/JCSSE.2013.6567355

[10] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv preprint arXiv:1907.11692* (2019). https://arxiv.org/abs/1907.11692

[11] luanti org. [n. d.]. luanti. https://github.com/luanti-org/luanti Accessed on September, 2025.

[12] OpenRA. [n. d.]. OpenRA. https://github.com/OpenRA/OpenRA Accessed on September, 2025.

[13] OpenTTD. [n. d.]. OpenTTD. https://github.com/OpenTTD/OpenTTD Accessed on September, 2025.

[14] Luca Pascarella, Fabio Palomba, Massimiliano Di Penta, and Alberto Bacchelli. 2018. How Is Video Game Development Different from Software Development in Open Source?. In *2018 IEEE/ACM 15th International Conference on Mining Software Repositories (MSR)*. IEEE, 392–402. https://ieeexplore.ieee.org/document/8595223

[15] PyGithub. [n. d.]. PyGithub. https://pygithub.readthedocs.io/en/latest/index.html.

[16] Selenium. [n. d.]. Selenium. https://selenium-python.readthedocs.io/.

[17] Konstantinos Stroggylos and Diomidis Spinellis. 2007. Refactoring–Does It Improve Software Quality? *Proceedings - ICSE 2007 Workshops: 5th International Workshop on Software Quality, WoSQ 2007* (2007). https://www.researchgate.net/publication/4262085_Refactoring--Does_It_Improve_Software_Quality

[18] SuperTux. [n. d.]. supertux. https://github.com/SuperTux/supertux Accessed on September, 2025.

[19] triplea game. [n. d.]. triplea. https://github.com/triplea-game/triplea Accessed on September, 2025.

[20] 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. https://doi.org/10.1109/ICSE43902.2021.00073

[21] wesnoth. [n. d.]. wesnoth. https://github.com/wesnoth/wesnoth Accessed on September, 2025.