Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Bias and Fairness in Code Review Systems: A Systematic Review and a Multidimensional AI Framework for Detection and Mitigation

Isah Muhammad Alhassan

isah.a@azmanuniversity.edu.ng

Azman University Kano https://orcid.org/0009-0009-9478-1541

Shamsu Sabo

shamsu.s@azmanuniversity.edu.ng

Azman University Kano

Abubakar Muhammad Umaru

Yusuf Maitama Sule University

Systematic Review

Keywords: Code Review, Software Engineering, Bias Detection, Fairness in AI, Fairness Metrics, Natural Language Processing, Machine Learning, Responsible AI

Posted Date: May 22nd, 2026

DOI: https://doi.org/10.21203/rs.3.rs-9771379/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: The authors declare no competing interests.

Abstract

Code review systems are vital in today's software engineering world because they can improve code quality, help foster collaboration among developers, and provide a medium for knowledge transfer through platforms like GitHub and GitLab. However, a new study has shown that the technical aspects of code review activities are not purely technical in nature; rather, code reviews are influenced by bias related to both human and social nature. These biases can impact the reviewers' decisions and cause the contributors being reviewed to be treated unfairly or to experience varied outcomes. The majority of tools available today which target code reviews do not account for these biases when they're created; instead, they focus primarily on identifying potential defects and improving the quality of the code being reviewed rather than addressing fair and unbiased reviews of the authored code. The present study conducts a Systematic Literature Review (SLR) of articles discussing bias and fairness associated with code review systems that were published between 2015 and 2025 to identify the source(s) of bias, methods used to measure fairness, and/or strategies used to mitigate bias in the context of software engineering and machine learning. From the results of a systematic search using multiple academic databases, 14 primary studies were identified. Based on the synthesized findings and an analysis of the findings from the primary studies, this paper identifies significant research gaps, including the lack of a unified approach to examining human, process, and algorithmic fairness in code review workflows. To address this gap, this paper proposes a Multidimensional Fairness Framework (MFF) that utilizes natural language processing (NLP), machine learning (ML), and fairness metrics (e.g., demographic parity or equalized odds) to detect, quantify, and mitigate bias in code review systems.

1. Introduction

Quality assurance via code review is a critical aspect of successful software engineering that is conducted over a collaboration platform such as GitHub, GitLab, and Bitbucket. Reviews help identify defects, transfer knowledge to others using the code, and allow teams to maintain the code collectively [1]. Traditionally, code review has been seen as a technical process; however, new research shows that social/human aspects are playing more of a role in determining outcomes during reviews [2]. Examples of bias in code review that can impact both feedback received through the review process and acceptance decisions include reviewer's subjectivity, different ways that reviewers and contributors communicate with each other, and types of people contributing to the project (this could include experience level of the contributor, gender cues, and whether or not the person has established a reputation)[3].

Automated tools for developing software show increasing support throughout the different stages of the development process by automating tasks related to defect discovery and code reviewer recommendations [4]. Yet, most AI-supported systems used for code review place an excessive emphasis on technical correctness and performance while largely ignoring fairness; this situation has paralleled the ongoing debates within the broader AI community regarding definitions of fairness and strategies for achieving fair outcomes, although applications of these concepts in practice are limited

[5]. Research findings in software and AI ethics indicate that fairness considerations are frequently neglected throughout the software development life cycle and that this failure has the potential to create social harm, reinforce stereotypes, and diminish trust in AI-assisted decision making [5].

The latest studies recommend that Fairness should be treated as a primary consideration across System Design, Data Handling, and Model Evaluation [5]. Studies within Software Engineering have started to explore methodologies for Fairness Testing; suggesting that there is now an increasing presence of Fairness practices being incorporated into automated workflows [6]. However, while some improvements have been made; there is still a lack of attention paid to Developing Bias Detection / Mitigation in Code Review Systems, as well as limited empirical evidence on how to achieve effective Interventions for Fairness.

In essence, this paper aims to fill those gaps by providing a systematic literature review focused on the issue of bias and fairness in code review systems. This includes an analysis of the existing literature on the sources of bias, the development of fairness metric(s), and documented strategies for mitigating bias. Additionally, the outcome of this review is the introduction of a Multidimensional Fairness Framework (MFF) that is designed to incorporate natural language processing (NLP), machine learning (ML), and fairness-aware metric(s) in order to identify and mitigate bias within code review workflows. The MFF will utilize a socio-technical perspective as well as the principles of Responsible AI to promote equitable, transparent, and inclusive code review on a global basis across all software development.

2. Systematic Review Methodology

This investigation employs a systematic literature review (SLR) method to evaluate bias and fairness in code review (c/r) systems. Relevant studies were identified across recognized academic databases, including Scopus, IEEE Xplore, SpringerLink, ScienceDirect, ACM Digital Library, and Google Scholar. The search utilized key terms such as "c/r bias," "fairness in SE," and "AI fairness".

2.1 The method of selecting studies

At first, the automated search yielded 165 records. After 47 duplicates were taken out of the total records, there were a total of 118 unique articles for a screening process. Articles were screened based on pre-defined inclusion and exclusion criteria, in order to identify those that were thematically relevant and academically of high quality.

· The inclusion criteria for the data collection comprised: Peer-reviewed journals or conference proceedings published between 2015-2025 and addressing Software Engineering (SE) (specifically the code review process) and ways that computing systems may be judged for fairness or bias.

· The exclusion criteria for collecting data from studies comprised: Being a non-English publication; being a non-academic source (e.g., blog, white paper); or being a study that had no direct relation to the core research area of the study.

2.2 The eligibility and data extraction process

Among the titles and abstracts reviewed to screen, 86 records were removed due to not meeting inclusion criteria. This left 32 full-text articles to be evaluated for eligibility. Following a full review, 18 articles were removed due to having no direct focus on workflow code reviews or not having enough depth of empirical/conceptual research on fairness.

Ultimately, 14 primary studies were found applicable for further evaluation. Each study was reviewed to analyze the methodology of the study, the key findings of the study, and the limitations of the study (see Table 1), and the identified research gaps generated the Multidimensional Fairness Framework (MFF). This structured and transparent approach to developing the framework and the underlying evidence-based factors help to ensure the framework is based on actual evidence of SE-based studies.

The PRISMA flow diagram (Figure 1) illustrates the screening and selection process and summarises the total number of records examined, included and excluded from the review.

3. Literature Review

3.1 Code Review in Software Engineering

Code reviews are widely regarded as a vital aspect of quality assurance in software development today. Code reviews employ reviewers to evaluate source code to find quality issues, inconsistencies, and adherence to coding standards [7]. Although code reviews have historically been viewed as strictly a technical process, recent studies demonstrate that social and cognitive factors play an important role in shaping the decision-making process for reviewers. An example of this is that a reviewer may make different decisions about the quality of source code based on how well they know the contributor, what they believe is the contributor's reputation, and how the code is being presented, rather than based on the technical quality of the code ([2], [4]). In addition, studies of open-source software development show that women and less-experienced developers are under-represented in code review, indicating an existing representation bias based on both gender and experience [3]. These findings point to a need to examine the code review process to ensure both technical considerations are considered as well as fairness considerations.

3.1.1 Human and Social Biases in Code Review

Studies have shown that when a reviewer does their review and cannot help but show a bit of subjectivity, it can impact how the contributors receive feedback from the contributor and build motivation in order to be able to continue working on their project[8]. [3] as well as evidence for bias relating to a code of acceptability, and the rate of participation for women in open-source communities. Specifically, women have more instances of their code waiting on comment than men do. The new findings also show structural inequities are in place that keep women out of these systems and that the challenges standing between women and their ability to contribute go beyond just technical ones. These

and other research experiences have taught us that in the future just being technically correct will not be enough, but also that the process of how and why decisions about reviews are made must also be taken into account, that is, evaluation of fairness.

3.2 AI-Assisted Code Review and Fairness

The integration of artificial intelligence (AI) and machine learning (ML) into code review workflows offers powerful capabilities, including automated defect detection, reviewer recommendation, and code summarization ([9], [10]). However, many AI systems are trained on historical data that reflect pre-existing biases, which machine learning models can inadvertently learn and perpetuate.

3.2.1 Fairness in ML-Based Code Reviewer Recommendation

Recent studies have found some problems with fairness in machine learning when it is used in software engineering. For example, [6]Did a study on how fair the machine learning systems are that suggest reviewers. The study found that reviewer recommendation systems disproportionately favour male reviewers. These findings indicate the presence of systematic bias within the model.

The study by [11] Also found that the ways we use to make the system fair work when everything is balanced. These methods do not work well when the data is not balanced. So, fairness-aware machine learning approaches must account for the challenges associated with real-world, imbalanced datasets.

3.3 Fairness in Machine Learning and Software Systems

The area of fairness in machine learning has become a lot better over the ten years. We now have different ways to define fairness measure it and make things fairer. People have carefully looked at all the ways to find bias in machine learning. They have come up with many different fairness metrics. For example, we have parity and equalized odds. There are also tools that can help reduce bias in many different areas, such as healthcare, hiring and systems that recommend things to people. Research conducted by [12] and [13] are examples of multiple studies that illustrate that fairness in ML is a complex issue that exists over several tasks. To achieve fairness, ML practitioners must look at a range of contexts in which fairness is needed and apply multiple methods to reduce bias within those contexts. Given all of these complexities, fairness in ML is an ongoing challenge and requires further research and methodology development for achieving fair outcomes.

3.3.1 Systematic Reviews of Fairness Metrics & Mitigation

Recent studies on bias and fairness in Machine Learning look at ways to make things fairer. They group these ways into three types: things we do before things we do during and things we do after we use Machine Learning. It is really important to pick the measures to make sure things are fair and work well. These studies found that even though we have made a lot of progress in making Machine Learning fairer it is still really hard to use these methods, in the world. They do not work well in all situations. Are often not used in the systems that are actually being used [12].

Recent studies have proposed methods for evaluating fairness in machine learning systems. They want to see if these systems treat people differently in situations. This means we need to make sure we are testing Machine Learning systems in a way that checks for fairness when we are building them [14]. Machine Learning systems need to be checked all the time to make sure they are being fair not once. We need to keep an eye on them to make sure they are treating people fairly.

3.4 Ethical and Socio-Technical Perspectives

Beyond purely algorithmic fairness, research in ethical and socio-organizational aspects of AI highlights gaps in developer understanding, organizational practices, and ethical maturity. Empirical studies report that practitioners often lack clear guidance on when and how to integrate fairness considerations during the software development lifecycle, underscoring the need for actionable frameworks and training [15].

3.4.1 Bias Auditing and Responsible AI in Software

Systematic reviews of bias auditing processes emphasize transparency and accountability in AI systems —extending beyond traditional software testing to include adversarial evaluation, participatory assessments, and fairness benchmarking. These reviews identify a range of techniques for auditing black-box models, but also highlight that auditing practices are not yet widely adopted in mainstream software engineering contexts [16].

3.5 Conceptual Overlap Diagram

To illustrate the interconnections between human biases, AI fairness, and software engineering practices, **Figure 2** presents a conceptual framework showing the multidimensional overlaps that inform code review fairness.

3.6 Critical Synthesis & Limitations

Comparative analysis of the literature shows that:

· Technical reviews mainly focus on how efficient and correct they are. They often leave out fairness considerations.

· Studies on bias show that there are unfair outcomes in reviews like biases against certain genders and people we don't know well. They rarely connect with technical solutions that can fix these problems.

· Research on fairness in machine learning provides ways to measure and fix unfairness. However, it rarely applies these methods to software engineering like code review.

· Studies on ethics and auditing highlight challenges for practitioners and organizations. Yet they do not provide solutions for code review workflows.

These limitations show a gap in research where human social bias and algorithmic fairness meet in code review systems. Each area provides insights. However few studies combine these perspectives into a framework that can be implemented. This gap leads to our proposal of a Multidimensional Fairness Framework. It unifies technical and algorithmic fairness considerations, for code review systems.

3.7 Conclusion of Review

Overall, the research shows that prejudice in AI and machine learning in general has received a lot of attention, and that social biases in code review procedures are becoming more widely recognised. However, there is still a lack of development in the integration of real-world deployment tactics, sociotechnical viewpoints, and fairness criteria in AI-assisted code review. The suggested multidimensional framework seeks to address the need for research that formalises fairness concepts inside operational code review workflows.

3.8 Systematic Review Summary

We synthesized all relevant literature on bias and fairness in code reviews into Table 1. Each study had different goals and methods and yielded different end products and different limitations. This analysis shows that even when systems operate as intended, there are human influences that can create bias. There is also a disconnect between how a human reviewer handles things and how a system handles things. There is a possibility of improving fairness particularly through the use of AI. To achieve this a holistic view of fairness is needed that looks at human judgement, algorithmic influence as well as foundational ideas of fairness. If fairness principles are applied in conjunction with AI, code review processes can be made fair and equal.

Table 1: *Studies on bias and fairness in code reviews.*

Study	Focus / Domain	Methods	Key Findings	Limitations
[1]	Code review practices in software engineering	Empirical study, qualitative analysis of code review data	Code review improves defect detection and knowledge sharing; social factors influence review outcomes	Limited to Microsoft projects; mostly technical focus
[2]	Reviewer influence on code quality	Statistical analysis of OSS projects	Reviewer reputation and participation impact review acceptance	Does not consider bias from gender or other social factors
[4]	Interpersonal conflict in code review	NLP analysis on review comments	Reviewer tone and conflicts affect contributor engagement	Limited to selected open-source repositories
[3]	Gender bias in open-source code review	Empirical study, quantitative metrics	Women contributors face slower review cycles and lower acceptance rates	Dataset imbalance; focuses only on gender bias
[5]	Fairness requirements in AI-enabled software engineering	Gray literature study	Highlights the need for fairness considerations in AI-assisted code review	High-level findings; not specific to software engineering workflow
[6]	Fairness-aware ML engineering	Empirical study	ML-based reviewer recommendations can favor certain groups; fairness-aware approaches help mitigate bias	Focus on experimental setup; real-world adoption limited
[7]	Modern code review expectations and challenges	Literature review & qualitative synthesis	Reviews improve defect detection, knowledge sharing; social factors affect outcomes	Limited dataset; mostly qualitative
[8]	Code review anxiety	Survey and interviews	Reviewer anxiety affects feedback quality and contributor engagement	Small sample; subjective measures
[9]	Machine learning for big code	Survey of ML approaches	ML can automate code review tasks; NLP-based summarization helps reviewers	Technical focus; fairness not covered
[10]	Instruction-finetuned language models	NLP experiments	Large language models can assist in code review and summarization	Models may inherit training data biases
[11]	Fairness in ML-based reviewer recommendation	Empirical evaluation	Male reviewers receive more recommendations; mitigation strategies limited with skewed data	Limited to specific ML models

[12]	Bias and unfairness in ML	Systematic review	Categorizes fairness metrics and mitigation strategies	Domain-agnostic; not tailored for software engineering
[13]	Fairness in ML models	Literature review	Highlights challenges and methods for bias detection and mitigation	Focuses on general ML, not code review-specific
[14]	Automated assessment of fairness definitions	Conceptual and tool-based evaluation	Proposes frameworks for fairness evaluation in ML	Practical implementation in code review not validated
[15]	Ethics and bias in AI	Survey	Developers lack guidance on fairness integration; organizational gaps	High-level review; no concrete code review solutions
[16]	Algorithm auditing & bias assessment	Systematic review	Transparency and auditing frameworks for AI; practical adoption limited	Rarely applied in mainstream code review

4. Proposed Framework / Methodology

4.1 Conceptual Overview of the Multidimensional Fairness Framework

The Multidimensional Fairness Framework (MFF) is designed to deal with bias and fairness issues in code review systems. It does this by looking at the problem from angles including how people think, how algorithms work and how processes are set up. The Multidimensional Fairness Framework is different from approaches that only focus on making sure the code is correct or using artificial intelligence to automate things. The Multidimensional Fairness Framework looks at how things how people think and how machines work together to affect the decisions that reviewers make.

1. **Human-centric biases**: The personal experiences of reviewers how well they know the people who wrote the code, the gender of the contributors and the reputation of the reviewers can all affect how reviews are written and whether they are accepted or not. The Multidimensional Fairness Framework considers these human-centric biases.

2. **Biases**: If machine learning models are trained on old code review data they can learn the same biases that were already there. This means that the models might treat some reviewers better than others, which's not fair. The Multidimensional Fairness Framework looks at these biases.

3. **Process-level biases**: The way that organizations do things like their peer review policies and how they help new contributors get started can create situations. The Multidimensional Fairness Framework considers these process-level biases and how they affect fairness, in code review systems.

This **Figure 2** establishes a conceptual lens for the methodology: it shows that fairness must be considered across multiple layers rather than in isolation.

4.2 Design of Architecture

The layers that make up the MFF cooperate to ensure that the data flows smoothly from beginning to end. The MFF has these layers that help the data flow easily. The data can flow easily from when the data collected to when the data is used to fix problems and look at the results of the MFF.

Each layer of the MFF is, like a building block. This means that each layer of the MFF can be changed or replaced if it is needed. The MFF can also work with platforms that look at code to help the MFF.

4.2.1 Data Collection & Preparation Layer

The main goal of the Data Collection & Preparation Layer is to get data that shows what the code is about and also the social side of code reviews. The MFF needs this data to understand the code and the people who work on it.

Processes:

· The framework acquires data from numerous platforms, including GitHub and code review systems such as Gerrit, which include both technical and social information about each contributor and their contributions.

· To help give context about each contributor and his or her contributions, additional characteristics (e.g., experience as a contributor, project size, and historical acceptance rate) were also included.

· The processes involved in preparing the dataset included the following:

I. Removing noise and irrelevant comments to enhance data quality.

II. Lowercasing, removing punctuation, and tokenizing to normalize text to create a consistent format for analysis.

III. Adding metadata to the comment data, which included characteristics such as inferred gender (when applicable), experience as a contributor, and the linguistic complexity of a comment.

These processes will create a dataset that is clean, consistent, and appropriate for identifying bias. Additionally, the inclusion of both technical and socio-contextual (e.g., non-technical) characteristics will also allow the framework to identify not only code-related issues, but human and interactional bias that may exist within the code review process.

4.2.2 Bias Detection Layer

This layer identifies possible bias in both review comments and decision-making processes to help prevent these from negatively influencing the results.

Algorithms Used:

· Sentiment and Toxicity Analysis:

Using Transformer-based architecture (e.g., RoBERTa) trained on Developer Communication datasets, the sentiment and toxicity of review comments can be analysed so that a negative performance score can be assigned to a review comment prior to it being viewed by a reviewer.

· Pattern Recognition using NLP Embeddings:

Contextual embeddings such as BERT are used to recognize inherent patterns of bias in reviewer-reviewee interaction. These models can give insight into how the reviewer is treating different reviewers (positive biases towards familiar reviewers, etc.) or how strictly they are evaluating new reviewers.

· Supervised Classification:

Using supervised classification models, we can classify review comments as biased or not based on labelled training set data. We can evaluate model performance using standard metrics with our target F1-Score being minimum of 0.85.

Having multiple methods of detection increases the frameworks likelihood of detecting all types of bias (both explicit and subtle), increasing the reliability of bias detection.

4.2.3 Fairness Quantification Layer

This level's function is to assess fairness in different dimensions via quantitative metrics that have been established.

Metrics & procedures.

· **Group Fairness**

Measured with demographic parity and disparate impact, defined by acceptable rates for groups of disparate demographic or experience level. Decision of an acceptable threshold (i.e., .80) for measuring fairness.

· **Opportunity Fairness**

Equalized odds are used as a measure for treating qualified contributor consistently based on the groups of reviewers reviewing them.

· **Individual Fairness**

Evaluated through the comparisons of similar contributions to determine if they received similar treatment, which are mostly based on measures of similarity found with code embeddings.

· Tools

Using AIF360 and FairLearn libraries for calculating fairness metrics, and supporting analysis.

Significance:

The overall use of these metrics enables more detailed assessment of fairness including identifying inequities and conducting ongoing monitoring and improvements.

4.2.4 Mitigation Interventions Layer

The goal of this layer is to lessen biases when reviewing code and to assure review processes proceed efficiently.

Strategies include:

I. **Pre-emptive nudges**:

The reviewer receives contextual information when potentially biased language or behaviour is detected by the review tool, encouraging non-biased and constructive feedback from the reviewer.

II. **Decision Recalibration**:

Adjustments of outcomes of reviews to reduce the influence of systematic bias on review results.

III. Reflections:

After the review, the reviewer receives information on the reviewer's behavior regarding standards and on potential bias.

Integration:

This layer can be implemented as an add-on for GitHub or GitLab to enable on-the-fly intervention in real time.

Rationale:

Ultimately, the framework assists in promoting more equitable interactions by developing clearer guidelines for fairer interactions and by enhancing awareness of contributors.

4.2.5 Evaluation Layer

The main goal of this layer is to measure how effective bias detection, fairness quantification and bias mitigation strategies will work.

Quantitative Evaluation

a. The performance of the framework will be evaluated using real-life datasets from code review services.

b. The performance will be assessed using precision-recall analytics and other related metrics.

c. An ablation study will be conducted to examine the contribution of individual components through controlled experiments.

Qualitative Evaluation

I. User studies and developer feedback sessions will be conducted to evaluate the usability of the framework and perceived fairness of the framework.

II. Review comments and other such data will be collected and evaluated to determine what impact our input had on the participants.

Criteria for Success

1. An **expected improvement** in fairness-related metrics is anticipated based on findings from related studies on fairness-aware machine learning and code review systems.

2. Bias detection performance is expected to achieve **high precision (≥90%)**, based on reported results from similar NLP and ML-based bias detection models in the literature.

3. Positive user feedback is **conceptually anticipated**, with user satisfaction expected to exceed a moderate acceptance threshold (e.g., 7/10) in future empirical evaluations of the framework.

4.3 Component Details and Algorithms

The Multidimensional Fairness Framework (MFF) consists of layered components that assess bias and fairness in code reviews. The components represent stages in the lifecycle of code review data, including: data collection / review comments, analysis of bias via natural language processing (NLP) techniques; quantification of fairness via using existing metrics; mitigation of bias through nudges and recalibrating decisions; and evaluation of how effective / usable the techniques are based on both quantitative and qualitative measures (**see Table 2**).

Table 2: Components, techniques, and purposes of the Multidimensional Fairness Framework.

Layer	Component	Algorithm / Technique	Purpose / Expected Outcome
Data Collection	Comment Extraction	API-based retrieval	Comprehensive review dataset
Bias Detection	Sentiment & Toxicity	RoBERTa classifier	Detect harsh/condescending tone
Bias Detection	Pattern Recognition	BERT embeddings	Detect favoritism or scrutiny bias
Bias Detection	Supervised Classification	Fine-tuned transformer	Identify subtle human biases
Fairness Quantification	Metrics Computation	aif360 / fairlearn	Measure group, opportunity, and individual fairness
Mitigation	Preemptive Nudges	Rule-based suggestions	Reduce biased language pre-emptively
Mitigation	Decision Recalibration	Constrained optimization	Adjust approvals for fairness
Mitigation	Reflection Prompts	Dashboard visualizations	Encourage reviewer self-awareness
Evaluation	Quantitative	A/B testing, curves	Evaluate effectiveness and fairness improvements
Evaluation	Qualitative	Think-aloud, thematic coding	Capture developer perception and usability

4.4 Workflow Summary

The MFF workflow can be summarized as:

1. We collect code review data, add information, and tidy it up.

2. After that, we employ tools to locate anything. such as determining whether there are strange patterns or if the tone is odd.

3. Next, we examine groups, opportunities, and individual items to gauge how equitable everything is.

4. After that, we make an effort to address any biases we discover by reminding people, making adjustments, and posing queries.

5. Lastly, we review our findings to ensure that our procedure is effective and that we are improving our fairness.

The MFF workflow helps fix bias, in code reviews all the time not once. It's good for schools and companies to use in their code review processes. The MFF workflow makes sure bias is addressed as a

process.

5. Conceptual Evaluation

Given that this study's a systematic review with a proposed framework we evaluate the Multidimensional Fairness Framework (MFF) conceptually rather than empirically. Based on insights from the literature and design considerations described in Section 3, the evaluation evaluates the MFF's capacity to identify and reduce bias in code review systems.

5.1 Evaluation Objectives

The conceptual evaluation of the MFF focuses on four aspects:

1. The MFFs ability to identify subtle biases in reviewer comments and decision patterns.

2. The expected performance of group, opportunity and individual fairness metrics in capturing inequities in the code review process.

3. The anticipated impact of real-time interventions, such as nudges, recalibration and reflection prompts on reviewer behaviour when using the MFF.

4. The practicality of embedding fairness mechanisms of the MFF without disrupting code review processes.

5.2 Metrics for Evaluation

The MFFs effectiveness is conceptually assessed using well-established fairness metrics:

· Group Fairness: We use demographic parity and disparate impact ratio to ensure equitable outcomes, across genders and experience levels in code reviews.

· Opportunity Fairness: Equalized odds is used to measure treatment of qualified contributors across reviewers.

· Individual Fairness: We use similarity scores based on code embeddings to predict treatment of similar contributions.

· Bias Detection Precision: We expect an F1 score of least 0.85 for identifying biased reviewer comments when using the MFF.

These metrics provide a dimensional assessment of the MFF covering systemic and individual-level fairness in code review systems.

5.3 Conceptual Analysis of Framework Layers

The layers of the MFF are conceptually analyzed in Table 3 and indicate how each layer contributes to fairness in the code review process. Data collection represents the breadth of contributors, bias detection represents favoritism/harshness, fairness quantification represents fairness disparities, mitigation represents reduction of bias instantaneously, and evaluation uses feedback for monitoring in the future. Each of these components relates to its intended fairness output.

Table 3: *Evaluation focus of MFF layers.*

Layer	Evaluation Focus
Data Collection & Preparation	Quality of bias proxy annotations and comprehensive representation of contributors.
Bias Detection	Ability to identify favoritism, harshness, or subtle social biases.
Fairness Quantification	Effectiveness of metrics in revealing disparities and highlighting areas for intervention.
Mitigation Interventions	Potential to reduce bias in real-time while preserving workflow efficiency.
Evaluation Layer	Feasibility of using historical data and developer feedback for ongoing monitoring.

5.4 Anticipated Outcomes

The purpose of the MFF, according to the analysis and architecture, is to achieve the following:

Increase equity of the outcomes generated by code reviews by decreasing the effectiveness of known social and algorithmic biases in the code review system as explicated in studies of fairness-oriented software engineering methods;

· Allow users to discover potential bias in review comments by applying NLP-based methods that improve the timing and value of feedback from reviewer judgments;

· Increase awareness of the impact of bias on reviewer decisions through the use of prompts and feedback that highlight reviewer biases;

· And allow MFF to morph seamlessly into existing code review processes and therefore could be implemented equally among researchers and industry practices within both academic and commercial software development arenas.

5.5 Why MFF Matters for Software Engineering

The evaluation shows that MFF:

· Connects the study of bias with methods for fair algorithms.

· Gives a plan for making code reviews fair.

· Provides advice, for developer's organizations and platform designers who want to make software development more equal.

This evaluation is theoretical and based on literature synthesis. No empirical experiments were conducted.

6. Conclusion

6.1 Summary

This study identified key challenges related to bias and fairness issues in code review systems that remain inadequately addressed. Consequently, a Multidimensional Fairness Framework (MFF) was proposed to mitigate these concerns and improve fairness outcomes.

The opinions of the literature analysis suggests that both human beings and artificial intelligence (AI) tools cause bias; for example, the manner in which they perceive gender (male or female) or other characteristics will affect the manner in which code assessment decisions will be made. Additionally, AI tools can also introduce bias into the code review process due to implicit human biases. Yet, the code review systems that currently exist place considerably more emphasis on the quality and functionality of the code as opposed to measuring or assessing fairness within each respective code review system.

The MFF incorporates many aspects of bias and fairness into the three categories (i.e., people, AI tools, and code review systems). The MFF uses a variety of techniques to remove or reduce bias from code reviews, including natural language processing (NLP) and machine learning (ML). The preliminary data suggests that this framework will assist in enhancing transparency as well as allowing for unbiased comparisons of the code review.

6.2 Contributions

This research adds a number of contributions. To begin with, it has organized existing research on code review bias/fairness into a comprehensible and organized manner. Secondly, the researcher has proposed a Multi-Dimensional Fairness Framework (MFF) that considers multiple dimensions of bias rather than treating it as the only form of bias present in code reviews. Thirdly, a method for evaluating the MFF in real-life settings has been described. Finally, an integrated view of technical and human factors is provided, demonstrating that both must be taken into account when working to improve fairness in code reviews.

6.3 Future Work

Future research directions are numerous. First, it will be important to perform testing of the Multi-Dimensional Fairness Framework (MFF) on an actual platform such as GitHub, so that the performance of the MFF can be evaluated within real-world scenarios. Second, examining other types of bias such as

culture and background types, would provide a clearer understanding of the underlying causes of bias in code reviews. In addition to these two areas of potential research, exploring how biases of code reviewers change over time and how systems can adjust their processes accordingly is also an area for future investigation. Testing the MFF in other types of development environments should also be a priority for future research. Finally, additional investigations of the impact of fairer code reviews on developers, in terms of their motivation and trust in AI technologies, will be necessary.

References

1. D. Badampudi, M. Unterkalmsteiner, and R. Britto, “Modern code reviews—Survey of literature and practice,” *ACM Trans. Softw. Eng. Methodol.*, vol. 32, no. 4, May 2023, doi: 10.1145/3585004.
2. O. Kononenko, O. Baysal, L. Guerrouj, Y. Cao, and M. W. Godfrey, “Investigating code review quality: Do people and participation matter?,” in *Proc. IEEE Int. Conf. Softw. Maint. Evol. (ICSME)*, Bremen, Germany, Sept. 29–Oct. 1, 2015, pp. 111–120, doi: 10.1109/ICSM.2015.7332457.
3. S. Sultana, A. K. Turzo, and A. Bosu, “Code reviews in open source projects: How do gender biases affect participation and outcomes?,” Feb. 2023. [Online]. Available: http://arxiv.org/abs/2210.00139
4. H. S. Qiu, B. Vasilescu, C. Kästner, C. Egelman, C. Jaspan, and E. Murphy-Hill, “Detecting interpersonal conflict in issues and code review,” *ACM Trans. Softw. Eng. Methodol.*, May 2022, pp. 41–55, doi: 10.1145/3510458.3513019.
5. T. Nguyen, C. Boufaied, and R. de S. Santos, “A gray literature study on fairness requirements in AI-enabled software engineering,” Dec. 2025. [Online]. Available: http://arxiv.org/abs/2512.07990
6. C. Ferrara, G. Sellitto, F. Ferrucci, F. Palomba, and A. De Lucia, “Fairness-aware machine learning engineering: How far are we?,” *Empir. Softw. Eng.*, vol. 29, no. 1, Feb. 2024, doi: 10.1007/s10664-023-10402-y.
7. A. Bacchelli and C. Bird, “Expectations, outcomes, and challenges of modern code review,” in *Proc. 2013 Int. Conf. Softw. Eng.*, 2013, pp. xx–xx.
8. C. S. Lee and C. M. Hicks, “Understanding and effectively mitigating code review anxiety,” *Empir. Softw. Eng.*, vol. 29, no. 6, Nov. 2024, doi: 10.1007/s10664-024-10550-9.
9. M. Allamanis, E. T. Barr, P. Devanbu, and C. Sutton, “A survey of machine learning for big code and naturalness,” *ACM Comput. Surv.*, Jul. 31, 2019, doi: 10.1145/3212695.
10. H. W. Chung et al., “Scaling instruction-finetuned language models,” Dec. 2022. [Online]. Available: http://arxiv.org/abs/2210.11416
11. M. M. Mohajer et al., “A first look at fairness of machine learning based code reviewer recommendation,” Jul. 2023. [Online]. Available: http://arxiv.org/abs/2307.11298
12. T. P. Pagano et al., “Bias and unfairness in machine learning models: A systematic review on datasets, tools, fairness metrics, and identification and mitigation methods,” Mar. 2023, *BDCC*. doi: 10.3390/bdcc7010015.

13. T. Das Jui and P. Rivas, “Fairness issues, current approaches, and challenges in machine learning models,” *Int. J. Mach. Learn. Cybern.*, vol. 15, no. 8, pp. 3095–3125, Aug. 2024, doi: 10.1007/s13042-023-02083-2.
14. G. d’Aloisio, C. Di Sipio, A. Di Marco, and D. Di Ruscio, “How fair are we? From conceptualization to automated assessment of fairness definitions,” *Softw. Syst. Model.*, Feb. 2025, doi: 10.1007/s10270-025-01277-2.
15. E. Ferrara, “Fairness and bias in artificial intelligence: A brief survey of sources, impacts, and mitigation strategies,” Mar. 2024, *MDPI Sci.*, doi: 10.3390/sci6010003.
16. V. Funda, “A systematic review of algorithm auditing processes to assess bias and risks in AI systems,” *J. Infrastruct. Policy Dev.*, vol. 9, no. 2, p. 11489, May 2025, doi: 10.24294/jipd11489.

Figures

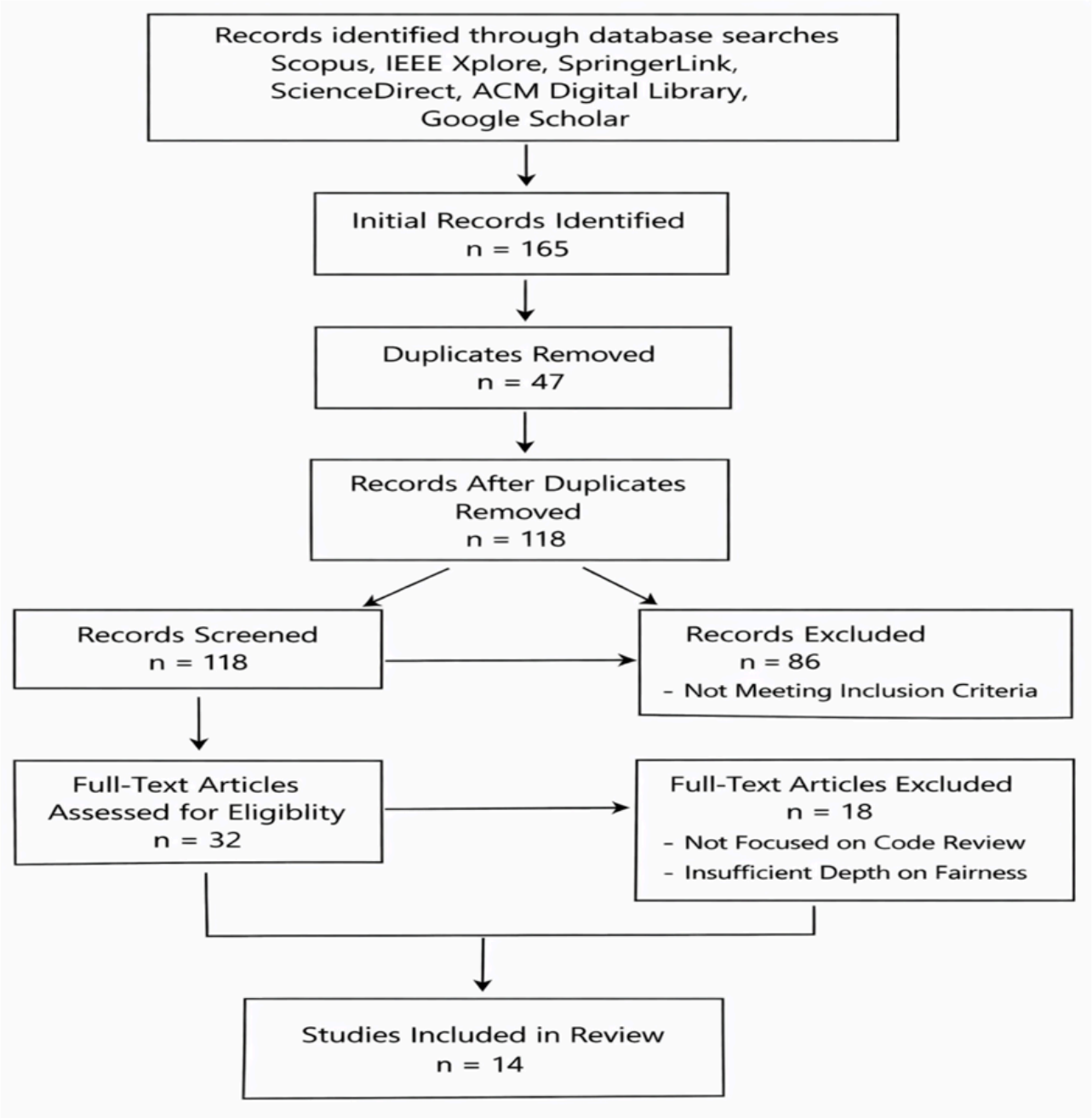


Figure 1

PRISMA Flow Diagram of the Study Selection Process

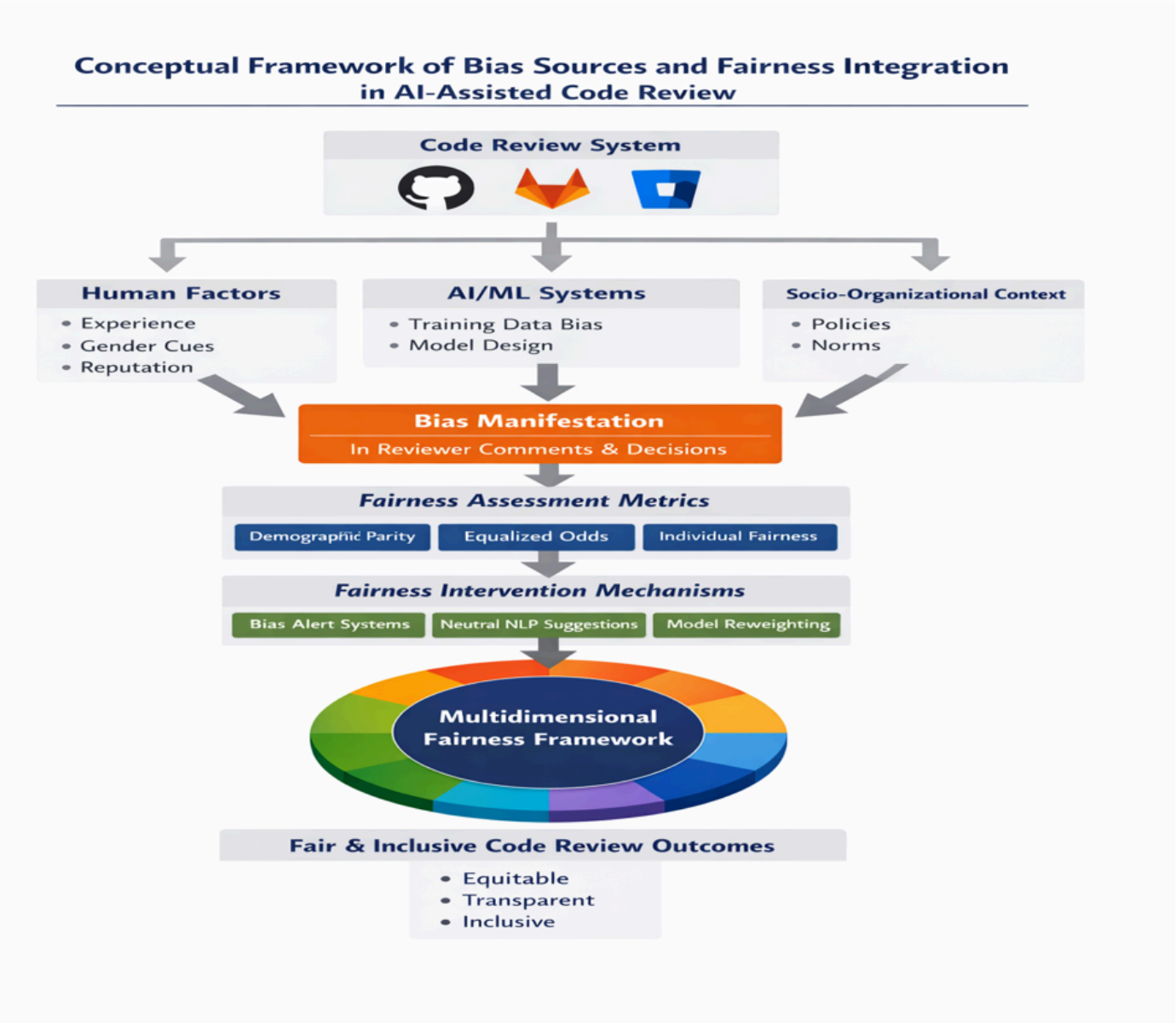


Figure 2

Framework for fairness and bias in code review systems

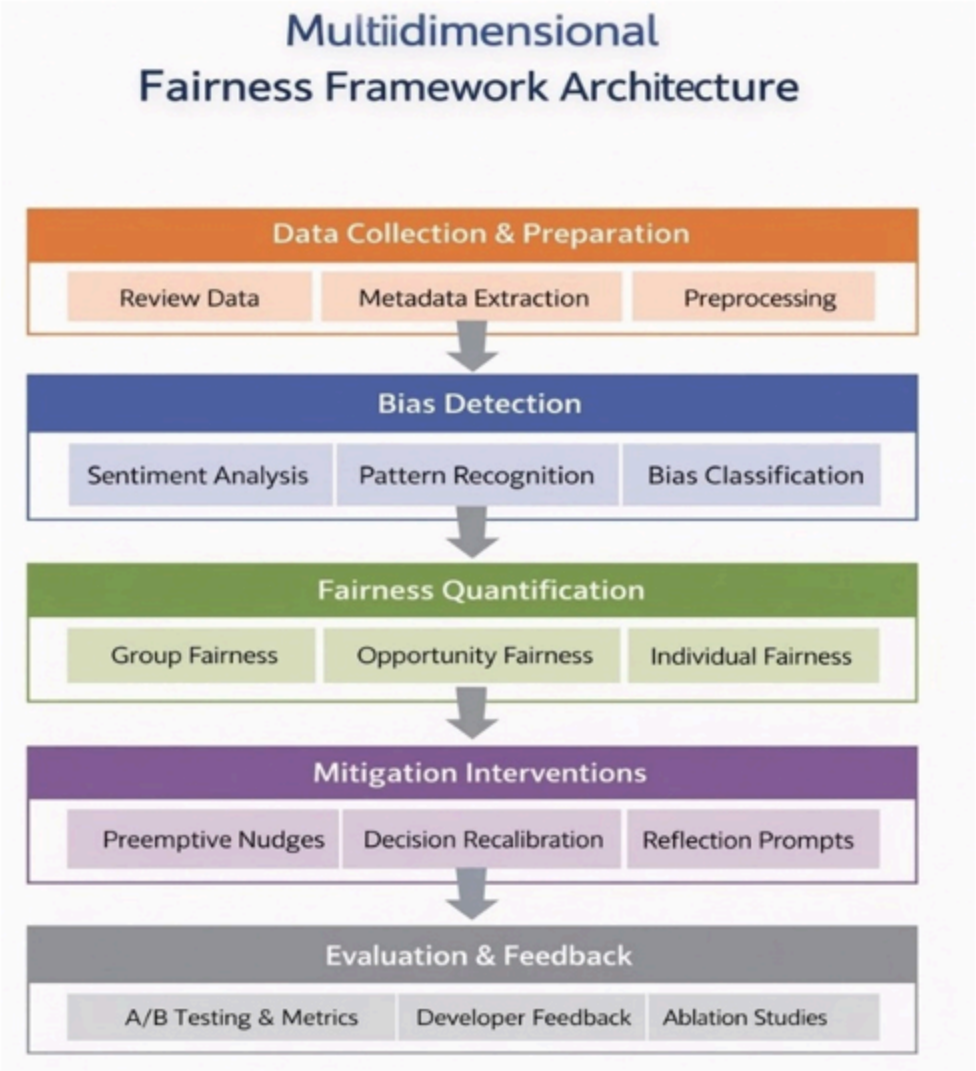


Figure 3

Layered architecture of the Multidimensional Fairness Framework.