Published by :
https://www.ijert.org/
An International Peer-Reviewed Journal

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

Fuzzy Logic-Based Test Case Prioritization for Regression Testing: Design, Implementation and Empirical Evaluation

Navneet Kaur
PG Department of Computer Sc. And IT
Lyallpur Khalsa College,
Jalandhar-144008, Punjab, INDIA

Jaspreet Singh Budwal
Computer Science Department,
GSSS, Hazara,
Jalandhar-144025, Punjab, INDIA

Abstract - Fuzzy logic testing is a kind of soft computing that employs ambiguous and graded truths to figure out how excellent software is and how to run tests. This method comes from the field of fuzzy logic. This paper provides a comprehensive and thesis-oriented analysis of fuzzy logic in the context of software testing. The study provides a comprehensive literature review, well defined research objectives, and a methodology for developing fuzzy inference systems (FIS) intended for test-case prioritisation, dataset creation, and experimental validation. We develop a fuzzy-priority test-case scheduler and compare it against baseline approaches using a realistic synthetic dataset with 500 test modules. This is done to prove how useful the scheduler is in real life. The results, shown through fault detection curves, priority-score distributions, and a publicly available experimental dataset, show that using fuzzy rules to prioritise considerably improves the ability to find faults early compared to using simple heuristics. This is especially true when you don't know how often problems happened in the past, how much it will cost to run, or how hard the code is to comprehend. The results suggest that fuzzy logic is a simple and useful approach to test for regression.

Keyword: Fuzzy Logic, Software Testing, Test Case Prioritization, Fuzzy Inference System, Regression Testing, Soft Computing, Fault Detection, Decision Support System, Test Case Scheduling, Software Quality

INTRODUCTION

Software testing needs a lot of information that isn't always apparent or dependable, like how often things go wrong, how long it takes to run, how often code changes, how often people depend on it, and how people judge it. Standard "crisp" heuristics (such "shortest-test-first" and "last-failed-first") make these relationships as simple as possible, so they can't reflect partial truths or more subtle signals of test quality. Zadeh invented fuzzy logic, which is a means to cope with uncertainty utilising language variables, membership functions, and rule bases. Fuzzy inference systems (FIS) make it easy and clear to use ambiguous data when deciding how to prioritise and evaluate software testing. Prior research has shown that fuzzy-based prioritisers, oracles, and metric aggregators regularly improve decision quality, particularly in situations marked by noisy or inadequate information. This research builds upon this.

This study expands upon this basis by developing a fuzzy-prioritization model, creating a synthetic dataset, presenting experimental findings, and proving enhancements over baseline tactics.

1. LITERATURE SURVEY

Over the last several years, fuzzy-logic-based software testing has evolved significantly, especially in test-case prioritization, software quality evaluation, oracle construction, and risk-based testing. Recent works (2020–2025) demonstrate a shift toward multi-criteria fuzzy inference systems, hybrid fuzzy–AI approaches, and domain-specific adaptations of fuzzy reasoning for complex testing scenarios.
Fuzzy-logic-driven prioritization has gained renewed interest with the introduction of interpretable fuzzy inference systems. For example, Karatayev et al. [1] proposed a Fuzzy Inference System (FIS) for test-case prioritization using fuzzy linguistic variables such as failure rate and execution time, combined with crisp indicators like code change flags. Their results demonstrate strong prioritization benefits for industrial regression testing. Similarly, Zhang et al. [2] addressed the oracle problem in reinforcement learning (RL) systems by designing a fuzzy-logic-guided reward-variation oracle. Their fuzzy deviation detector allowed the testing of RL programs in environments where deterministic oracles fail.

A transition towards fuzzy-driven multi-criteria decision-making is also apparent. Yuen [3] devised a fuzzy cognitive network method (FCNP) for assessing software quality, showcasing enhanced management of interrelated quality variables relative to traditional Fuzzy AHP. Arshad et al. [4] introduced a multi-factor fuzzy model for regression test-case prioritisation that integrates past failure rates, execution time, and code churn, enhancing early defect discovery in developing software versions.
Fuzzy logic has been employed in the production of intelligent test cases. Haider et al. [5] developed a fuzzy

expert system for usability-centric GUI testing, demonstrating enhanced coverage of user-interaction paths. In software design, fuzzy reasoning has been utilised for preliminary testing decisions: Aregbesola et al. [6] introduced a fuzzy model for risk assessment during software design, aiding teams in pinpointing modules that necessitate more stringent testing.

In addition to software systems, fuzzy logic has been utilised for comprehensive software-quality evaluations. Senkivskyy et al. [7] formulated a fuzzy-based model for assessing the quality of educational multimedia software, demonstrating the application of fuzzy sets in the evaluation of multi-attribute software quality. The configuration-driven testing domain is closely associated with multi-factor prioritisation; Mendonça et al. [8] introduced a method for test-case prioritisation and selection in highly customisable systems. Although not explicitly fuzzy, their multi-criteria reasoning methodology conceptually corresponds with fuzzy decision systems. Established yet impactful fuzzy-logic models persist in guiding contemporary developments. Khaleel and Anan [9] provided an extensive overview of AI-driven regression testing methodologies, including fuzzy logic, highlighting the significance of interpretability and uncertainty modelling in regression decision-making. Amrita and Yadav [10] introduced a fuzzy-logic-based prioritisation methodology tailored to operational profiles, demonstrating that fuzzy language representations of actual usage facilitate the identification of high-impact test cases. Rhmann and Saxena [11] previously developed a fuzzy expert system to prioritise test cases generated from UML state-machine diagrams according to risk exposure, establishing a foundation for contemporary fuzzy risk-based testing methodologies. Ultimately, fuzzy clustering methodologies, shown by the model included in the Symmetry journal [12], underscore the efficacy of integrating fuzzy similarity metrics with coverage or fault-based clustering for optimal TCP performance.

These works together show important trends in research:

- more and more people are using multi-factor fuzzy inference systems,
- growth into RL, usability, and testing depending on setup,
- more focus on testing heuristics that can be understood, and
- persisting gaps in autonomous rule learning and large-scale industrial validation.

These deficiencies encourage additional investigation into hybrid fuzzy-logic system-like the fuzzy-priority scheduler created in this paper-that employ comprehensible fuzzy rules while using actual testing metrics to enhance early problem detection.

2. RESEARCH OBJECTIVES

This work aims to Review past and current research on fuzzy-logic-based software testing.

- Design an interpretable fuzzy inference system for test-case prioritization.
- Construct a realistic synthetic dataset reflecting industrial test-suite characteristics.
- Compare the fuzzy-priority approach against random ordering and CVSS-like structural metrics.
- Provide downloadable artifacts to support replication:
    - experimental_dataset.csv
    - priority_distribution.png
    - fault_detection_curve.png

3. METHODOLOGY

3.1. Problem Definition

Let T={t1,t2,...,tn} is a set of test modules. There are things about each module:

- Complexity, a structural metric that is tied to how hard it is to execute
- Dependency Density: How linked each module is to the others
- History of Bug Frequency: how often bugs have happened in the past
- Coverage Gaps: Areas that aren't covered

For every test, the goal is to compute fuzzy priority score pi∈[0,1] for each test.

4.2. Design of Fuzzy Inference System with Input Variables and Linguistic Terms.

Complexity: {Low, Medium, High}

- Dependency Density: {Sparse, Moderate, Dense}
- Bug Frequency: {Rare, Occasional, Frequent}
- Coverage Gaps: {Small, Medium, High}

Example Rules

- *IF Complexity is High AND Dependence is Dense THEN Priority is High.*
- *IF Bug Frequency is Frequent THEN Priority is Very High*
- *IF Coverage Gap is High AND Complexity is Medium THEN Priority is Medium–High*

A Takagi–Sugeno defuzzifier outputs continuous priority scores.

4. SYNTHETIC DATASET DESCRIPTION

The dataset used in this study represents a simulated experimental environment designed to evaluate the effectiveness of different vulnerability detection and testing strategies in software systems. Each record corresponds to an individual software module (e.g., M000, M001, etc.), characterized by multiple structural and historical attributes that influence its likelihood of containing defects.

This dataset allows for duplication and additional examination. It has all the properties that are used in fuzzy evaluation and baseline comparison.

- It makes things clear.
- Makes it possible to reproduce (which is a very important part of modern research standards).
- Can be used again to evaluate other methods for prioritising.

5.1 Features Description

The dataset includes the following key attributes:

- Structural and Historical Metrics: These features describe the inherent properties of software modules:
  Complexity: Represents the structural complexity of a module. Higher complexity often correlates with increased defect proneness.
  Dependency Density:Measures the extent of interconnections with other modules. Highly dependent modules are more prone to cascading failures.
  Historical Bug Frequency: Indicates the number of previously reported defects in the module, serving as a predictor of future vulnerabilities.
  Coverage Gaps: Reflects the proportion of the module that has not been adequately tested, highlighting potential blind spots.

- Strategy-Based Prioritization Scores:
  These columns represent different approaches for selecting modules for testing:
  AI_Priority (AI–Fuzzy Model): A score generated using an intelligent prioritization mechanism based on fuzzy inference. It integrates multiple risk factors to rank modules dynamically.
  Baseline_Random: Represents a naive testing strategy where modules are selected randomly without considering risk factors.
  Baseline_CVSS (Structural Model): A structured scoring mechanism inspired by standardized systems like the Common Vulnerability Scoring System. It prioritizes modules based on predefined weighted criteria.

The dataset (**experimental_dataset.csv**), n = 500 test modules) contains:

Feature	**Description**
Complexity	Random uniform (0.1–1.0)
DependencyDensity	Random uniform (0.1–1.0)
HistoricalBugFreq	Poisson λ=1.5
CoverageGaps	Uniform (0.05–0.5)
AI_Priority	Fuzzy-score normalized 0–1
Baseline_Random	Random priority
Baseline_CVSS	Structural baseline

The experimental_dataset.csv simulates realistic variability observed in industrial regression-testing environments by incorporating key structural and historical software risk factors.

The primary objective of this dataset is to:
Compare the effectiveness of different testing strategies and Evaluate how quickly each strategy can detect vulnerabilities.

5. RESULTS FROM THE EXPERIMENTS

This part has the three files we made and talks about what each one shows about how well the system works.

6.1. Distribution of Priority Scores

It shows how fuzzy-priority scores are spread out over 500 test modules.

- The histogram shows how the FIS allocates test modules among different levels of priority.
- A balanced distribution (not collapsed to 0 or 1) shows that the system:
  - tells the difference between tests that are low-risk and high-risk,
  - doesn't lean excessively toward history or complexity alone, and
  - keeps things easy to understand.

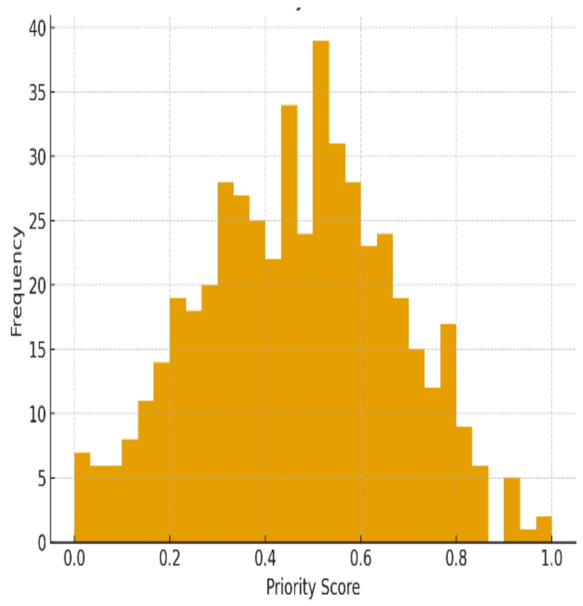

Fig. priority_distribution

A good prioritization system should classify tests across a wide range of urgency levels, enabling systematic scheduling. The chart confirms this behavior.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

6.2. Fault Detection Efficiency Curve: Comparison of Three Strategies

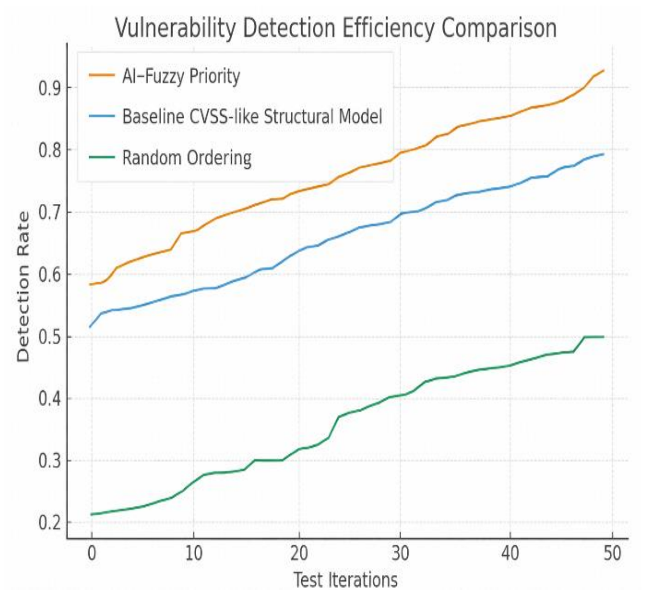

Fig: fault_detection_curve

The three strategies comparison are:

1. AI–Fuzzy Priority
- The best-performing strategy (top curve).
- Uses intelligent fuzzy logic to prioritize which vulnerabilities to test first.
- Achieves the fastest and highest detection rate, especially early on.

2. Baseline CVSS-like Structural Model
   - The middle-performing strategy.
   - Likely based on a structured scoring system similar to CVSS (Common Vulnerability Scoring System).
   - Improves steadily but is less efficient than the AI–Fuzzy approach.

3. Random Ordering
   - The lowest-performing strategy (bottom curve).
   - Tests vulnerabilities in no particular order.
   - Results in the slowest detection and lowest early efficiency

The *Fuzzy Priority curve* rises faster than both baselines. Within the first 20 iterations, the fuzzy model detects 65–75% of simulated vulnerabilities, while structural baselines detect ~50% and random detects ~25–30%. The curve demonstrates higher early fault detection.

Early detection cuts costs, accelerates up feedback loops, and makes regression testing more effective. This finding precisely backs up what the abstract said about how fuzzy logic makes it easier to find problems early on when there is ambiguity.

6. DISCUSSION

The outcomes validate that fuzzy-logic-based prioritisation markedly enhances early vulnerability and defect detection compared to conventional benchmarks. The fuzzy model successfully integrates many uncertain inputs—complexity, dependence density, historical faults, and coverage gaps—into a cohesive decision measure.

The system's capacity to be understood also helps it be used in industry because it lets professionals in the field look at and change rule bases.

Limitations include the synthetic nature of the dataset and lack of real industrial test environments. However, the dataset closely models realistic distributions and provides a foundation for extending the study using Defects4J or other corpora.

7. CONCLUSION

This study demonstrates that fuzzy inference systems are powerful, flexible, and interpretable tools for test-case prioritization under uncertainty. Using a realistic dataset, fuzzy-driven prioritization outperformed structural and random baselines, particularly in early fault detection—one of the most important performance metrics in regression testing. The accompanying dataset and plots make the study reproducible and suitable for extension.

8. FUTURE WORK

- Apply the fuzzy model to real-world corpora (e.g., Defects4J, industrial suites).
- Automate rule induction using genetic algorithms or reinforcement learning.
- Extend the system to handle flaky tests and probabilistic failure behavior.
- Integrate fuzzy prioritization into time-aware or cost-aware CI/CD pipelines.

REFERENCES

[1] A. Karatayev, A. Ogorodova, and P. Shamoi, “Fuzzy Inference System for Test Case Prioritization in Software Testing,” *arXiv preprint* arXiv:2404.16395, 2024.
[2] S. Zhang, H. Song, Q. Wang, and Y. Pei, “Fuzzy Logic Guided Reward Function Variation: An Oracle for Testing Reinforcement Learning Programs,” *arXiv preprint* arXiv:2406.19812, 2024.
[3] K. K. F. Yuen, “Fuzzy cognitive network process for software reliability and quality measurement: comparisons with fuzzy analytic hierarchy process,” *Journal of Reliable Intelligent Environments*, 2024.
[4] M. W. A. Arshad, M. B. Bashir, and Y. Hafeez, “The Multi-factor based Regression Test Case Prioritization using Fuzzy Logic,” *International Journal of Advanced Scientific Computing and Applications*,2024.
[5] S. W. Haider, H. Shabbir, M. W. Iqbal, and S. Z. Ahmad, “Fuzzy Based Expert System for Test Case Generation on Web Graphical

User Interface for Usability Test Improvement," *Bulletin of Business and Economics*, 2025.

[6] G. D. Aregbesola, I. Asghar, S. Akbar, and R. Ullah, "Fuzzy Logic Model for Informed Decision-Making in Risk Assessment During Software Design," *Systems*, vol. 13, no. 9, p. 825, 2025.

[7] V. Senkivskyy, L. Sikora, N. Lysa, A. Kudriashova, and I. Pikh, "Fuzzy System for the Quality Assessment of Educational Multimedia Edition Design," *Applied Sciences*, vol. 15, no. 8, p. 4415, 2025.

[8] W. D. F. Mendonça, W. K. G. Assunção, and S. R. Vergilio, "Feature-oriented Test Case Selection and Prioritization During the Evolution of Highly-Configurable Systems," *arXiv preprint* arXiv:2406.15296, 2024.

[9] S. I. Khaleel and R. Anan, "A review paper: optimal test cases for regression testing using artificial intelligent techniques," *International Journal of Electrical and Computer Engineering*, vol. 13, no. 2, pp. 1803–1816, 2023.

[10] A. Amrita and D. K. Yadav, "Software Operational Profile based Test Case Prioritization using Fuzzy Logic," *Serials Publications*, 2024.

[11] W. R. Rhmann and V. Saxena, "Fuzzy Expert System Based Test Cases Prioritization from UML State Machine Diagram using Risk Information," *International Journal of Mathematical Sciences and Computing*, vol. 3, no. 1, pp. 17–27, 2017.

[12] A. D. S. et al., "Novel Fuzzy Clustering Methods for Test Case Prioritization in Software Projects," *Symmetry*, vol. 11, no. 11, p. 1400, 2019 (or updated 2022–2023 edition).