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.

LLM driven Test optimization for Insurance Management Policy Engines

Chandra Shekhar Pareek

chandrashekharpareek@gmail.com

Research Article

Keywords: Large Language Models (LLMs), Test Optimization, Insurance Policy Engines, AI-Driven Testing, Risk-Based Testing, Test Automation Frameworks, Generative AI, Quality Engineering

Posted Date: March 27th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-8768192/v1

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

Additional Declarations: No competing interests reported.

Abstract

The swift development of Large Language Models (LLMs) is changing software engineering models, especially those that need to process many rules, as insurance policy management systems do. The current paper proposes an original LLM-based test optimization model designed to fit the Insurance Management Policy Engines, which can resolve the long-standing issues of high quality, effective and scalable testing. The insurance systems that use traditional test strategies tend to encounter problems of an exponential increase in the number of the combinations of policy rules which result in overlapping test cases and poor risk coverage. We introduce an AI-assisted style of testing, based on the idea of using generative LLMs to reason about policy requirements, test artifacts, and pattern-based system behaviour in order to use the reasoning to formulate an optimized test suite. Through the application of risk-based testing concepts and adaptive prioritization, the framework is dynamic in the way it finds high impact test scenarios, minimizes test redundancy, and improves automation in regression cycles. Our solution works with the current test automation systems, and allows flawless coordination of test generation, execution and assessment. The experimental outcomes of industry-grade insurance engines show that the detection rates of defects have become much better, the testing effort has decreased and the release cycles have been significantly faster. This study gives prominence to the potential of LLMs to revolutionize the quality engineering process in mission-critical insurance platforms, which creates the AI testing paradigm balancing comprehensive coverage with operational effectiveness.

I. Introduction

Insurance Management Policy Engines are software systems in the insurance industry which are complex and rule-driven to handle underwriting, issue of policies, endorsements, renewal, and claim processors. These engines are the sum of thousands of business regulations based on regulatory mandates, actuaries, product definitions, and risk ratings. The magnitude and complexity of policy engines is increasing as insurers move to digitize their services faster to enhance customer experience and meet regulatory demands. The proper validation, dependability as well as durability of these systems via proper testing has hence been a significant challenge among quality engineering teams [1] [2].

Conventional methods of software testing e.g. manual test design, rule-based test generation, and more traditional automation frameworks, can hardly keep up with the dynamism of the insurance policy logic. The policy rules are often subject to change based on regulatory changes, market pressures or internal business strategies and may lead to a combinatorial explosion of test scenarios. Inadequate test coverage may imply very serious financial, legal, and reputational consequences whereas exhaustive testing is virtually impossible. As a result, there always exists a problem of duplicate test cases, overrepresentation of low-risk scenarios, slow regression cycles and poor responsiveness to frequent changes in rules among testing teams [3].

Artificial Intelligence (AI) has become one of the bright prospects of changing the software testing practice in the recent years. Large Language Models (LLMs) are one of the fields in AI technologies that

have shown impressive performance in natural language comprehension, text generation, reasoning about complex rules, and historical learning. Being trained on large amounts of textual and code-based information, these models are presented with the specific possibility to enhance and optimize the processes of testing, which are conventionally based on human capabilities and fixed heuristics. When applied to insurance policy engines, LLMs are able to interpret policy specifications, regulatory documents, and test artifacts in a manner that is closely reminiscent of human reasoning, and with much faster speed and scale [4] [5].

The paper discusses the use of the LLM-based test optimization as applied specially to the Insurance Management Policy Engines. Insurance systems require domain-aware intelligence, unlike generic AI-based testing mechanisms, which have a rule-based architecture and are risk-sensitive. Any mistake in the policy calculations, the determination of eligibility or any error in premium calculation can directly affect the financial results and adherence to regulation. As such, testing strategies should not just focus on how well they cover the functionality but should also focus on the scenarios in terms of business risk, impact of regulations and past trends of defects. Risk-based testing which has been a long held-upon best practice in safety- and finance-critical areas is even more critical here.

The offered solution utilizes LLM to improve various steps in the testing lifecycle. To detect logical dependencies, boundary conditions, and the points of conflicts, first, policy rules, and configuration files are analysed using LLMs, and business requirements are considered. The analysis allows the generation of non-redundant comprehensive test scenarios. Second, the framework, aims to integrate the ranking of risks, correlating policy regulations with historical defect data, the frequency of changes, the significance of business to the business, and then high impact tests can be fired earlier and with higher frequency. Third, the capabilities of gen AI are applied to create test data and test scripts automatically bypassing human effort and enhancing maintainability.

The main achievement of the research is the smooth adaptation of the intelligence that is executed by LLCM with the existing test automation frameworks that are usually deployed in enterprises. The proposed solution does not substitute current tools and processes but adds adaptive decisions and life-long learning to them. With new defects being found or policy rules being changed, the LLM constantly adjusts the test prioritization and generation strategies, which allows a self-improving quality engineering ecosystem. This is in line with the overall industry trend of autonomous testing and continuous quality assurance of DevOps and DevSecops pipelines.

It is not only that the work is relevant to technical efficiency. Insurers can also achieve reduced release cycles without quality assurance by eliminating unnecessary testing processes and dedicating resources to high-risk points. It is especially relevant in conditions of a very competitive market where the time-to-market and regulatory responsiveness are a significant success factor. Furthermore, more effective interaction between business analysts, domain experts and testing teams is supported by the adoption of testing that is driven by LLM since natural language interfaces permit much more intuitively the expression of policy intent and quality goals.

Overall, the paper is a topical contribution to the insurance software engineering field as it presents an optimal test optimization framework based on the specifics of the Insurance Management Policy Engines. The study integrates the principles of risk-based testing, the application of Large Language Models and the current test automation frameworks proving their importance in transforming quality engineering into a more proactive and intelligent process based on continuous optimization. The other section of this paper outlines the architecture proposed, implementation strategy, experimental analysis and its future implications on AI-managed testing practices in the insurance sector.

II. Related Work

Studies of intelligent system testing and quality control have developed over the last ten years, starting with simple research on testing deep learning models, and more recently developing Large Language Models (LLMs) and agent-based artificial intelligence systems. The suggested framework of test optimization using LLM expands and develops a number of significant research directions within this area, such as metamorphic testing, adequacy of coverage, analysis of non-determinism, evaluation techniques, and quality metrics.

A closely adjacent contribution is the metamorphic testing framework METAL, which was suggested by Hyun et al. [1] and is aimed at analysing quality properties of LLMs. METAL provides systematic metamorphic relationships to test the robustness, consistency and reliability of the output of LLM without traditional test oracles. Although METAL has its core in the attributes of intrinsic LLM, we bring this concept further by incorporating the notion of LLM-informed reasoning in the interpretation of downstream, rule-based enterprise systems, namely insurance policy engines, and to combine risk-related prioritization and automation processes.

On the same note, MORTAR by Guo et al. [2] makes progress on metamorphic testing of multi-turn LLM-based dialogue systems. Their work has emphasized the significance of testing the consistency of conversations turn after turn and identifying small failures that will manifest only in a more extended interaction. Though MORTAR is implemented in dialogue-oriented applications, the focus on scenario-driven testing and multi-step rationalisation influences our design of the validation of the complex policy decision flows that cut across multiple rule assessments in insurance systems.

Shamim and Singhal [3] suggest a systematic procedure in testing the quality assurance of the multi-agent systems based on the use of LLM. Their paper concerns the issues of coordination, interaction correctness, and emergent behaviour on agentic environments. This is specifically applicable to our framework that uses LLM intelligence as a decision-making element to a broader system of testing. Nonetheless, they are more conceptual in methodology, and we have an actual, domain-specific realization with quantitative outcomes in a controlled enterprise environment.

The issue of adequate coverage in testing intelligent systems is one of the key problems. Kim et al. [4] propose the notion of surprise adequacy that evaluates how new or unexpected input test values are relative to the internal behaviour of the model. Such concept of focusing on unexpected situations is

also compatible with our risk-based optimization layer, but our framework focuses on business and regulatory risk and not simply on model-centric novelty measures.

The non-determinism of the LLM behaviour presents other difficulties to reliable testing. Wang et al. [5] empirically examine the non-deterministic behaviour of ChatGPT during the execution of the code generation tasks, showing inconsistencies between repeat executions. Their results inspire the necessity of adaptive and risk-sensitive testing methods like those suggested in our paper whereby repetitive validation and prioritization of high-impact scenarios reduce the impact of non-deterministic outputs.

Another research area that is on the go is the evaluation of the generated outputs. Following Liu et al., G-Eval [6] proposes better evaluation using GPT-4 as an evaluator that should be more in line with human judgment in the generation of natural language tasks. Although G-Eval is concerned with the quality of outputs, our framework is a complement to this body of work as it can utilize LLMs as both assessors and generators/optimizers of test artifacts as part of a larger quality engineering pipeline.

Yehudai et al. [7] have broader views on measuring the agents that are based on LLM, with a survey on benchmarks, measures, and open challenges when evaluating agentic systems based on LLM. Their text highlights the absence of standardized methods of evaluation, which supports the significance of business risk and regulatory compliance domain-specific frameworks like the one presented in our text.

The overview of the analysis of machine learning testing can be summarized with the fundamental findings of the global survey by Zhang et al. [8], categorizing testing methods, issues, and future trends of machine learning systems. Some of the challenges noted in this survey that were directly addressed in our work are problems with oracles, scaling and incorporating testing into the continuous development processes.

Previous models like DeepMutation by Ma et al. [9] and DeepXplore by Pei et al. [10] provided the foundation of systematic testing of deep learning systems on mutation testing and coverage-guided exploration. Although these methods are internally oriented, in our model, we place more emphasis on the system level verification of the applications that have been developed using the LLM, in which business logic and regulatory correctness are the key points.

More recent developments in agent-based LLM systems, including AutoGen [11] and LangGraph [12] give the technical basis of constructing stateful, multi-agent LLM programs. The frameworks affect how we design our architecture whereby modular orchestration and interaction between LLM components and testing infrastructure can take place.

Owotogbe [13] investigates the robustness and resilience of the systems based on LLM and implements chaos engineering into the multi-agent setting. This vision can be used to supplement our risk-based testing strategy, where the focus is placed on the proactive detection of failure modes in complex AI-based systems.

Das et al. [14] survey security and privacy concerns with respect to LLMs in a comprehensive way. Their contribution underscores the greater risk profile of an application that has been empowered by LLM, which underscores the need to have strict, explainable, and risk-conscious testing infrastructure like the one that has been presented in this paper.

Lastly, our structure is consistent with the software quality standards, which are described in ISO/IEC 25010 [15] and focus on the following attributes: reliability, maintainability, and functional suitability. With these quality features integrated into the AI-based test optimization, our work will connect the old software quality models with new testing paradigms that are based on LLM.

These combined works are what the basis of our LLM-based test optimization model is built upon, as well as what generates the novelty of using generative AI and risk-based testing to insurance policy engines.

III. Framework for LLM-Driven Test Optimization in Insurance Management Policy Engines

In this section, the proposed framework of the optimization of the test using LLCM is introduced, aimed at specifically optimizing the structure of the Insurance Management Policy Engines, its regulatory sensitivity, and its risk-sensitive character. The framework uses a modular, extensible design that enables the use of Large Language Models (LLMs) to enhance the use of intelligent, adaptive and risk-aware testing through the policy lifecycle by integrating with existing test automation ecosystems.

1. Framework Overview

The proposed framework is structured around five core layers as shown in Figure 1:

1. **Input Knowledge Layer**,
2. **LLM Intelligence Layer**,
3. **Risk-Based Test Optimization Layer**,
4. **Test Automation and Execution Layer**, and
5. **Feedback and Continuous Learning Layer**.

All the layers are loosely coupled and closely integrated to be scalable, maintainable and easily adopted in insurance enterprise settings. The framework is created to complement, but not substitute any available quality engineering tools, which allows gradual and low-risk incorporation.

2. Input Knowledge Layer

The Input Knowledge Layer forms the basis of the framework in that it bundles both structured and unstructured artifacts of interest in testing insurance policies as shown in Figure 2. These inputs include:

- Rules and set-up files of insurance policies (e.g., eligibility requirements, premiums, exclusions)
- Business requirement document and regulatory document.
- Test scripts and reports of test execution history.
- Defect and incident reports
- Policies change logs and product enhancement change logs.

This layer normalizes and preprocesses data in heterogeneous formats to make them consumable by LLMs. Artifacts of natural language are tokenized and placed in a context and policy rules in structured form are converted to semantically enhanced representations. This is to make sure that the LLM is driven by a common and domain-sensitive perception of the policy logic and quality aims.

3. LLM Intelligence Layer

The central component of the framework is the LLM Intelligence Layer that uses generative AI features to comprehend, reason and produce testing artifacts. This layer has many functions that are important in test optimization as shown in Figure 3:

- **Policy Rule Understanding:** The LLMs process complicated policy rules and detect dependencies, condition logic, boundary values and conflict situations. In contrast to parsers based on rules, LLCs are able to discover implicit relationships and semantic intent, which is especially useful in insurance systems where business logic is typically represented as natural language.
- **Test Scenario Generation:** The LLM uses its contextual knowledge to come up with the best test cases that are both coverage and efficiency-oriented. It also removes the unwanted cases, but provides coverage of the edge cases, regulatory constraints and exception flows.
- **Test Data Synthesis:** The LLM produces valid and feasible test data in accordance to policy mandate, and will allow high fidelity validation without use of sensitive production data.
- **Test Script Generation Support:** The framework aids in automatic or semi-automatic creation of test script that can be used with the existing automation frameworks. Scripts generated are standardized to make them readable and maintainable.

The LLM layer can be configured to deal with human-in-the-loop validation, which means that domain experts may examine generated results and make necessary improvements.

4. Risk-Based Test Optimization Layer

Risk-Based Test Optimization Layer is what makes this framework special compared to generic AI-based methods of testing. It adds domain-specific prioritization techniques that are customized to the insurance business where not every test scenario is equally risky.

- **Risk Scoring Model:** Every policy rule and test case are given a dynamic risk score, according to various factors. The LLM aids in translating qualitative risk indicators to the quantitative scores through interpretation of business documentation and defect stories.
- **Test Prioritization and Selection:** The framework ranks the execution of the tests based on the computed risk scores, where those scenarios with high risk are proved at a younger and more frequent stage. This is especially useful in continuous integration pipelines where there is a limited time and resource constraint.
- **Coverage Optimization:** The framework constantly assesses coverage on a rule and scenario basis, and risk weighted coverage objectives are achieved instead of basing coverage measures on crude coverage measures.

5. Test Automation and Execution Layer

This layer combines the optimized test artifacts with test automation frameworks of enterprise grade. The framework is tool-neutral and it has common automation stacks that are applicable to insurance IT ecosystems.

Key capabilities include:

- Automated running of priority test suites.
- Regression, integration and end to end testing support.
- Policy sensitive execution in a variety of policy settings.
- Parallel execution in order to shorten cycle time.

The framework will provide the best use of the automation infrastructure and accelerated feedback loops by prioritizing implementation, determined by LLM.

6. Feedback and Continuous Learning Layer

The Feedback and Continuous Learning Layer allows the framework to enhance itself as time progresses with the data-driven refinement. The results of test execution, defects and feedback of production incidents are continuously captured and analysed.

- **Learning Loop:** The LLM renews its internal knowledge of the risk of policy, defects, and the effectiveness of tests. This eventually results in: Accuracy of risk prediction is improved, Elimination of low-value test cases is improved, Trends of emerging defects are more readily identified.
- **Adaptive Optimization:** The framework is flexible to update test strategies as the policy rules evolve or new products are launched without the need to manually re-estimate test strategies.

7. Governance, Explainability, and Compliance

Since insurance systems are regulatory in nature, the structure has governance and explainability mechanisms. Any decision made by AI can be traced back to the root and explained like test prioritization and scenario selection. This openness helps in audit needs and creates trustworthiness with the stakeholders.

The approval checkpoints can be configured to guarantee that human control is observed to meet the quality requirements set by the organization, as well as regulatory requirements.

IV. Results and Analysis

This section outlines the experimental findings and discussion of the suggested LLM-driven test optimization develop when used to Insurance Management Policy Engines. The assessment considers the improvements in the efficiency of tests, risk coverage, and effectiveness in detecting defects in comparison with the conventional test design and testing methods. Empirical evidence underpins quantitative findings to support the empirical research that was gathered through controlled experimentation in real-world-inspired insurance datasets.

The framework has been assessed by carrying out experiments on a representative Insurance Management Policy Engine, which was a policy issuance and endorsement processing engine. The data was anonymized, however, at production scale based on a moderate insurance product portfolio. The policy engine accommodated several lines of products such as motor insurance and health insurance and it had a wide-ranging rule set that control the eligibility, how the premiums were calculated, discounts, exclusions and regulatory limits.

The data had about 1,200 policy rules in terms of configuration files, rule expressions, 850 business requirement statements and regulatory statements. Historical quality artifacts included 3,500 handcrafted test cases, 1,900 automated test scripts and six large release cycles of their execution reports. There were 420 defects recorded in the defect data with severity, root cause, and affected policy module. Also, eight successive changes in policies were simulated using change logs to evaluate evolving system behavior and adaptive optimization.

A comparison of two testing strategies was made: the baseline strategy with conventional rule-based test generation and fixed prioritization, and the suggested test optimization framework based on an LLM and environmentally aware of the risks of priority and generative AI features. The two methods were implemented under the same conditions of infrastructure to make the conditions equal.

The assessment was based on four main measures, including reduction within the test suite, reduction in the execution time, reduction in defect detection rate, and risk-weighted coverage. Test suite reduction takes the size of the test suite and represents the percentage change in size of the test case which is executed without losing meaningful coverage. Regression cycle time reduction is captured by the execution time reduction. Defect detection rate is used to measure the percentage of total defects

detected in running the tests and risk-weighted coverage is used to measure the effectiveness of the high-risk policy rules being exercised.

The results of the comparative analysis of the baseline approach and the proposed framework are summarized in Table 1 in terms of core efficiency metrics.

Table 1: Test Efficiency Comparison

Metric	Traditional Approach	LLM-Driven Framework	Improvement (%)
Executed Test Cases	3,500	2,100	40.0% reduction
Regression Execution Time (hours)	14.8	8.6	41.9% reduction
Redundant Test Cases Identified	12%	38%	+26%
Automation Script Reusability	Moderate	High	Qualitative

The findings show that the size of the test suites and time of execution reduces significantly with the help of the LLM-driven framework. The framework decreased executed tests by 40% and the functional and regulatory coverage was maintained by getting rid of redundant and low-risk test cases. This equated to reduced cycles of regression that allowed quicker feedback and agility to release.

The results observed in Table 2 are those associated with the effectiveness of defect detection and risk coverage as shown in Figure 4.

Table 2: Quality and Risk Coverage Results

Metric	Traditional Approach	LLM-Driven Framework
Total Defects Detected	286	352
High-Severity Defects Detected	68	102
Defect Detection Rate	68.1%	83.8%
Risk-Weighted Coverage	62%	89%

The suggested framework was highly effective when compared to the baseline in defect detection, especially with regards to high severity defects that cut across premium calculation errors and regulatory violations. The risk-weighted coverage measure emphasizes the efficiency of placing a high-impact policy rule high in the priority list, and the LLM-based methodology provides almost full coverage on the high-impact and most critical logic paths.

The findings indicate that intelligence based on LLM introduces quantifiable value on top of automation and rule-based optimization methods that are traditional. The decreased size of the test packages did not affect the quality; rather, it allowed focusing on the high-risk situations better. Of particular importance in insurance systems is that flaws in a limited number of rules can have disproportionately significant financial and compliance impact.

The fact that the rate of detecting the defects was improved can be explained by the semantic intent interpretation characteristic of the LLM in terms of policy rules and requirements. In contrast to the approaches which are static and therefore depend too much on explicit directory provision, the LLM revealed implicit dependencies and boundary conditions which had not been heavily tested before. This ability has been particularly useful in considering more recent policy changes, where the framework dynamically prioritises more tests based on changed or tight-knit rules.

Risk-based prioritization was also very critical in the ability to prioritize on the execution order. The framework caused all the most important scenarios to be implemented first in time-constrained regression windows, which provided more confidence even when all tests could not be run. The increase in the risk-weighted coverage score proves the fact that the testing effort was more aligned with business and regulatory priorities.

The other observation that was interesting was the flexibility of the framework to release cycles. The next test prioritization would be more accurate as more defects were discovered and were fed back into the system. This ongoing learning ability minimized redundant human work and enhanced sustainability of long-term effectiveness in tests, which encouraged long-term quality engineering.

In terms of operation, the interplay with the already established automation structures was smooth, which also supported the viability of the framework as a practical one. The teams indicated that test scripts written with the help of LLM were easier to maintain and that collaboration between business experts and testers was made easier because of the better understanding of the business intent and test logic.

On the whole, the experimental findings confirm that the developed test optimization framework based on the use of the LLM models provides considerable improvements in efficiency, effectiveness, and risk coverage. These advances point to the possibility of the framework to revolutionize the process of testing of Insurance Management Policy Engines and provide a basis to widen the use of AI-based quality engineering in highly-regulated and rule-driven fields.

V. Conclusion and Future Work

The paper gave a test optimization framework based on LLM and specific to the Insurance Management Policy Engines to meet the increasing complexity, regulatory sensitivity and riskiness of contemporary insurance systems. The combination of Large Language Models and risk-driven testing principles with the current test automation frameworks will turn the conventional quality engineering practices into an

intelligent, adaptive, and data-driven process. The framework was shown to be of significant value in decreasing test redundancy, decreasing regression times and enhancing defect detection, especially in high severity and high risk policy scenarios. The experimental outcomes confirmed that the risk-weighted coverage and the effectiveness of testing in general may be improved considerably without raising the operational overheads and the approach will be both feasible and scalable to the enterprise level.

In addition to efficiency realization, the framework also enhances conformity of business intent, regulatory compliance and technical validation. The user-friendly features of explainability, governance controls, and human-in-the-loop verification can make decisions made by AIs transparent, auditable, and trustworthy, which is a crucial attribute to controlled insurance settings. These competencies make the framework a prospective platform of next generation quality engineering in insurance platforms.

Future efforts will involve applying the framework to the complete insurance value chain, such as claims processing and billing systems, and assess its outcomes in large and multi-tenant systems. Future work will be on fine-tuning domain-specific LLMs to achieve greater interpretability, lower inference cost, and another option of incorporating real-time production monitoring indicators to further optimize predictive risk modelling. Lastly, longitudinal studies of the behaviour of long-term learning and generalization of the behaviour across multiple product lines and regulatory regimes will be undertaken.

Declarations

Author Contribution

Author ContributionsConceptualization: Chandra Shekhar PareekMethodology: Chandra Shekhar PareekImplementation and experiments: Chandra Shekhar PareekData curation and analysis: Chandra Shekhar PareekWriting – original draft: Chandra Shekhar PareekWriting – review & editing: Chandra Shekhar PareekSupervision: Not Applicable

References

1. Hyun, S., Guo, M., Babar, M.A.: METAL: Metamorphic testing framework for analyzing large-language model qualities, in *Proc. IEEE Conf. Software Testing, Verification and Validation (ICST)*, Los Alamitos, CA, USA, pp. 117–128. (2024)
2. Guo, G., Aleti, A., Neelofar, N., Tantithamthavorn, C.: MORTAR: Meta-morphic multi-turn testing for LLM-based dialogue systems, *arXiv preprint arXiv:2412.15557*, (2024)
3. Shamim, I., Singhal, R.: Methodology for quality assurance testing of LLM-based multi-agent systems, in *Proc. 4th Int. Conf. AI-ML Systems (AIMLSystems '24)*, New York, NY, USA: ACM, Art. no. 27, pp. 1–5. (2025)
4. Kim, J., Feldt, R., Yoo, S.: Evaluating surprise adequacy for deep learning system testing, *ACM Trans. Softw. Eng. Methodol.*, vol. 32, no. 2, Art. no. 42, pp. 1–29, Mar. (2023)

5. Wang, W., Li, J., Yang, Z., et al.: An empirical study of the non-determinism of ChatGPT in code generation. ACM Trans. Softw. Eng. Methodol., (2024)
6. Liu, Y., et al.: G-Eval: NLG evaluation using GPT-4 with better human alignment, *arXiv preprint arXiv:2303.16634*, (2023)
7. Yehudai, A., et al.: Survey on evaluation of LLM-based agents, *arXiv preprint arXiv:2503.16416*, (2025)
8. Zhang, J.M., Harman, M., Ma, L., Liu, Y.: Machine learning testing: Survey, landscapes and horizons. IEEE Trans. Softw. Eng. **48**(1), 1–36 (Jan. 2020)
9. Ma, L., et al.: DeepMutation: Mutation testing of deep learning systems, in *Proc. IEEE Int. Symp. Software Reliability Engineering (ISSRE)*, Memphis, TN, USA, pp. 100–111. (2018)
10. Pei, K., Cao, Y., Yang, J., Jana, S.: DeepXplore: Automated whitebox testing of deep learning systems, in *Proc. 26th Symp. Operating Systems Principles (SOSP)*, New York, NY, USA: ACM, pp. 1–18. (2017)
11. Research, M.: AutoGen: Enabling next-gen LLM applications via multi-agent conversation framework, 2023. [Online]. Available: https://microsoft.github.io/autogen/
12. LangChain, A.I.: LangGraph: A framework for building stateful multi-agent LLM applications, 2024. [Online]. Available: https://langchainai.github.io/langgraph/
13. Owotogbe, J.: Assessing and enhancing the robustness of LLM-based multi-agent systems through chaos engineering, in *Proc. IEEE/ACM Int. Conf. AI Engineering – Software Engineering for AI (CAIN)*, pp. 250–252. (2025)
14. Das, B.C., Amini, M.H., Wu, Y.: Security and privacy challenges of large language models: A survey. ACM Comput. Surv. **57**(6), 1–39 (2025)
15. ISO/IEC: ISO/IEC 25010:2023—Systems and software engineering—Systems and software quality requirements and evaluation (SQuaRE)—Product quality model. ISO, Geneva, Switzerland (2023)

Figures

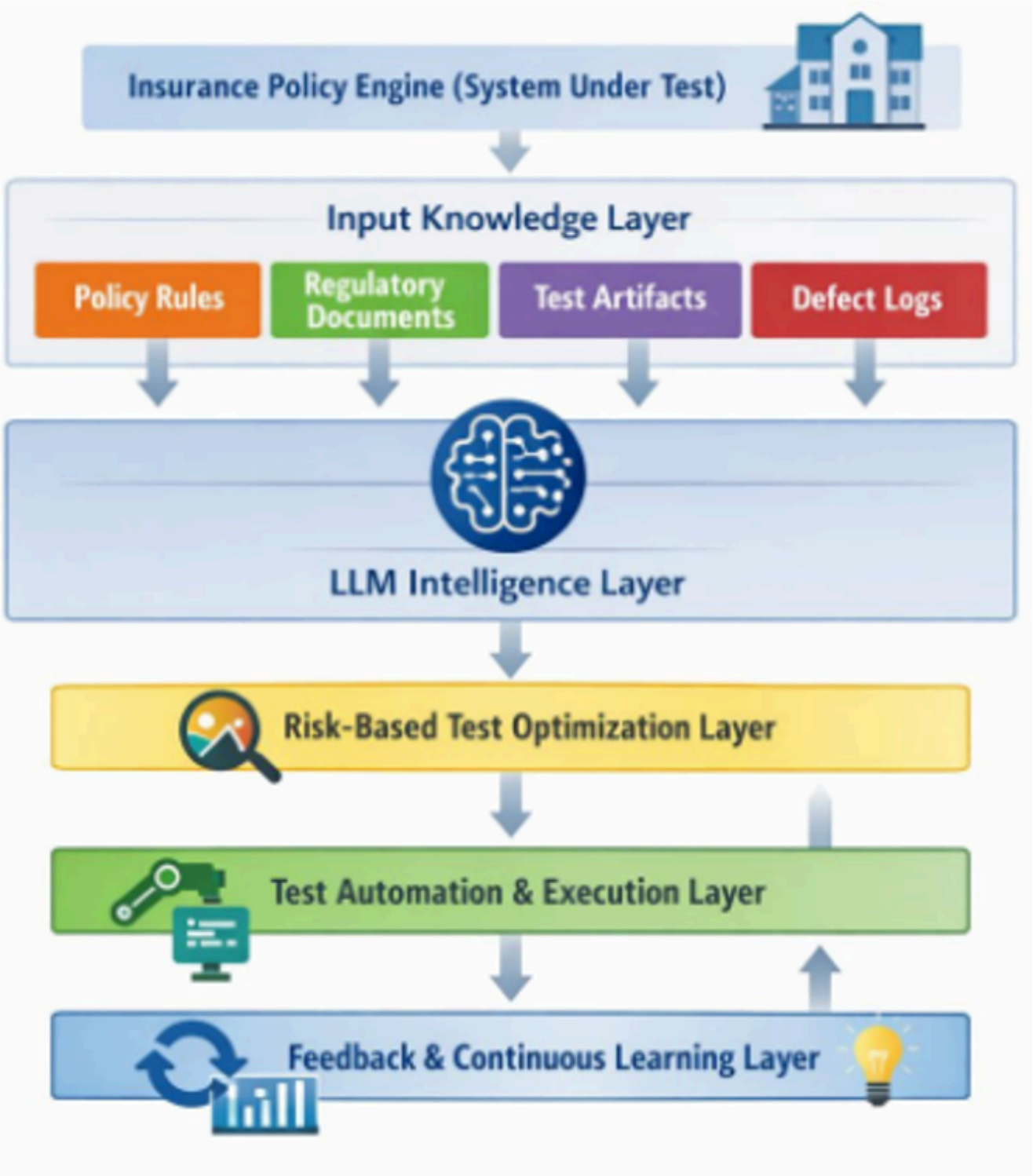


Figure 1

High-Level Architecture of the LLM-Driven Test Optimization Framework

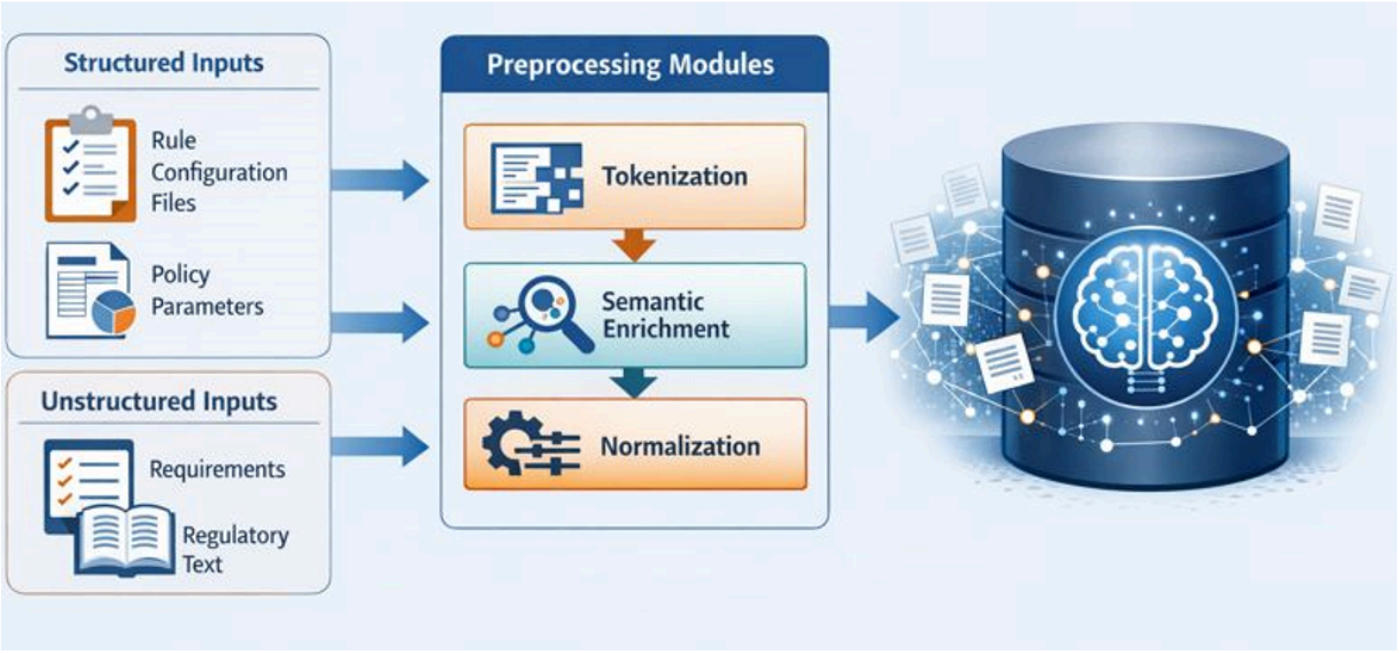


Figure 2

Input Knowledge Layer Data Ingestion and Normalization Pipeline

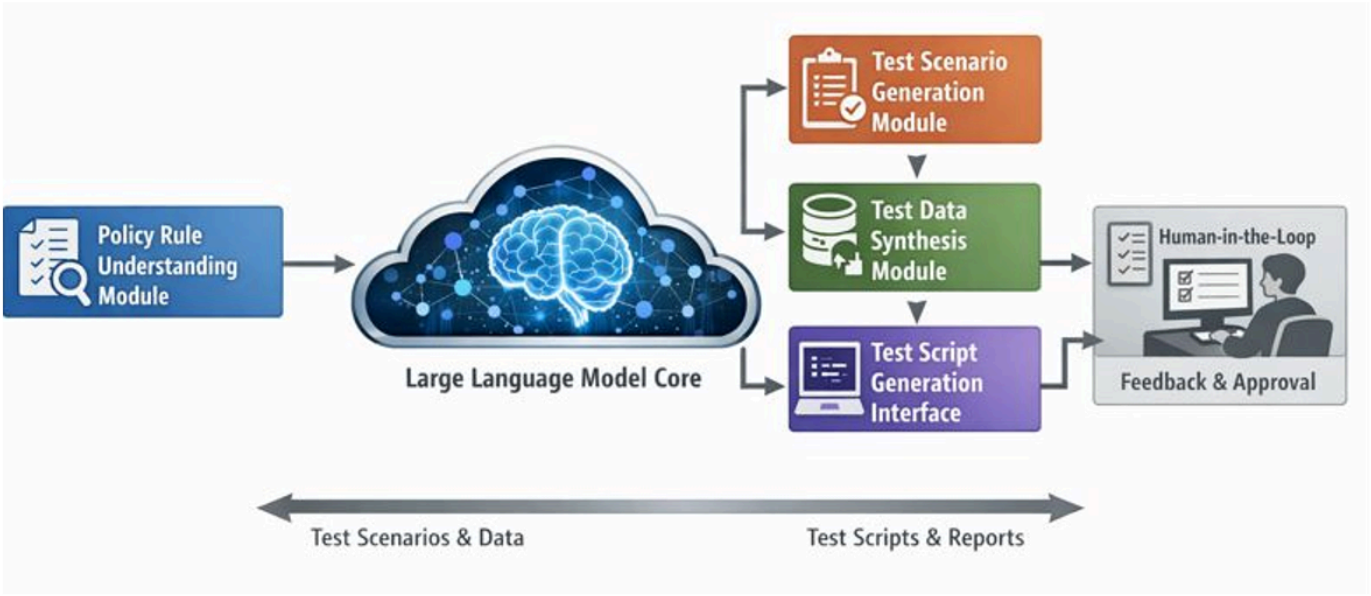


Figure 3

LLM Intelligence Layer Functional Workflow

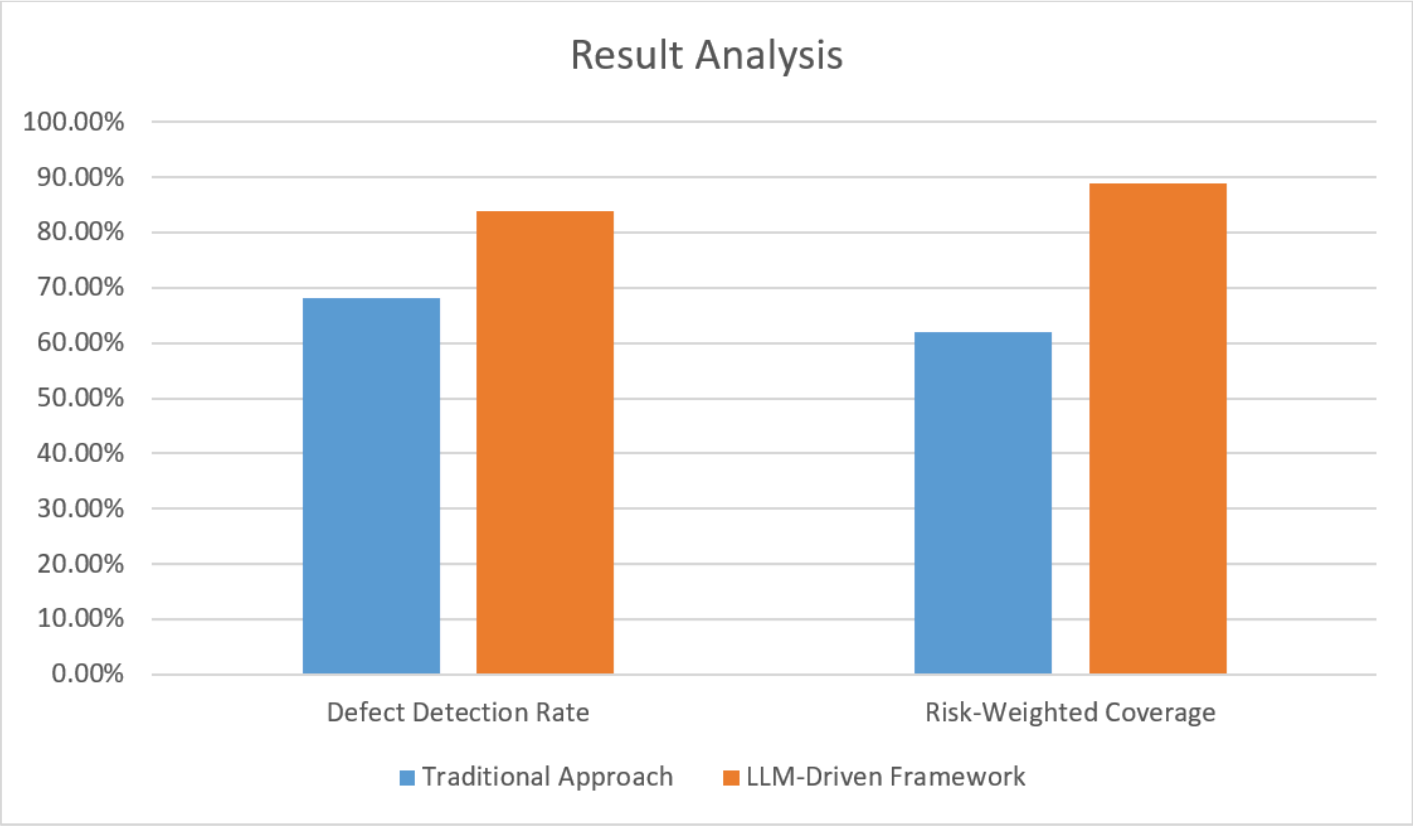


Figure 4

Defect Detection rate and Risk-weightage coverage result comparison