European Journal of AI, Computing & Informatics

Vol. 1 No. 4 2025

Article Open Access

Generative AI Test Generation and Intelligent Defect Attribution Method for Large Scale Distributed Systems

Mingde Guo [1,*]

[1] Amazon, Irvine, CA 92620, United States
* Correspondence: Mingde Guo, Amazon, Irvine, CA 92620, United States

Received: 14 November 2025
Revised: 30 November 2025
Accepted: 12 December 2025
Published: 16 December 2025

Abstract: To address the increasingly complex demands of large-scale distributed systems in test generation and defect localization, this study proposes a strategy grounded in a generative AI-driven test generation and intelligent attribution framework. The test environment is first constructed through the formal definition of system architecture, business semantics, and interaction constraints, ensuring that the generated tests accurately reflect real operational logic. Then, leveraging the reasoning and combinatorial capabilities of large-scale generative models, the test environment is expanded into a diversified set of execution paths, thereby enhancing the system's ability to cope with highly dynamic and uncertain runtime conditions and improving test coverage in complex operational states. Furthermore, multi-source observability data-including logs, traces, metrics, and dependency metadata-is integrated and modeled to produce a structured representation for abnormal correlation analysis. Based on this representation, causal reasoning is applied to derive dependency relationships and event propagation paths among system modules, enabling effective and efficient root cause diagnosis across distributed nodes. This approach significantly reduces the diagnostic search space and improves the interpretability of system anomalies. Experimental validation using a model test prototype demonstrates that the proposed method outperforms traditional testing and fault attribution approaches in terms of test diversity, fault excitability, and accuracy of causal determination. These results indicate that the framework offers robust support for the construction of AI-based assurance mechanisms in large-scale distributed systems, contributing to improved system reliability, fault tolerance, and automated quality assurance.

Keywords: generative AI; test generation; observability causal inference; defect attribution

1. Introduction

With the continuous expansion of distributed system architectures, the unpredictability of system behavior and the increasingly intricate interactions among heterogeneous computing and storage resources have become increasingly prominent. Modern large-scale distributed environments often operate under diverse workloads, dynamic scaling strategies, and cross-platform coordination, resulting in an exponential growth of system state spaces [1]. Under such circumstances, both testing and diagnosis face substantial challenges. Traditional testing approaches, which depend heavily on manually crafted scenarios or domain expert experience, are unable to provide sufficient scenario coverage and cannot effectively expose the diverse fault modes triggered by complex resource interactions. Similarly, traditional diagnostic methods frequently rely on static rule sets or pattern matching, which struggle to adapt to evolving system structures and dynamic operational contexts [2].

Compounding these issues is the fragmented nature of observational data in distributed systems. Operational logs, performance metrics, and distributed traces-although abundant-typically exist in isolated formats without unified semantic definitions, making cross-source correlation extremely difficult. The absence of shared semantics not only limits comprehensive anomaly recognition but also hinders the reconstruction of causal chains behind abnormal events. Consequently, diagnosing faults becomes time-consuming and error-prone, which further restricts the reliability and maintainability of large-scale distributed systems [3].

In recent years, the emergence of generative AI has offered a new paradigm for addressing these long-standing bottlenecks. Compared with traditional analytical models, generative AI demonstrates significant advantages in semantic modeling, abstraction of complex interactions, representation of contextual dependencies, and reasoning over structured and unstructured information. These capabilities make it possible to interpret distributed systems from a more holistic perspective, reconstruct multi-level relationships, and automatically infer hidden patterns that are difficult to capture through manual methods. Leveraging these strengths, a new generation of automated test construction and intelligent fault-cause diagnosis systems can be developed, enabling more diversified test generation, improved abnormal event reasoning, and more accurate dependency reconstruction between system modules.

Building upon this motivation, this article explores how generative AI technologies can be used to integrate system architecture information, business semantics, and operational constraints to generate richer, more adaptive test environments and improve the efficiency of fault analysis. The goal is to form a coherent theoretical and methodological system that enhances both testing completeness and diagnostic precision in large-scale distributed systems. By constructing semantic linkages across heterogeneous data sources and applying reasoning-based approaches to uncover the propagation paths of abnormal events, the proposed framework aims to overcome the limitations of traditional methods and provide a robust foundation for intelligent assurance of complex distributed systems. Ultimately, the work presented in this paper contributes to the development of more reliable, interpretable, and autonomous monitoring and testing technologies suitable for the next generation of large-scale system architectures.

2. Test Generation Method Based on Generative AI

2.1. Formal Modeling of System-Level Test Generation

The complexity of action states, types, component dependencies, and external conditions makes large-scale distributed system behavior hard to capture. A formal action model is needed to describe the system's structure, state transitions, and constraints, so test semantics can be expressed and derived in a unified framework. System behavior can be represented as constrained state transitions:

$$s_{t+1} = \mathcal{T}(s_t, e_t, \Xi_t) subjectto \Phi(s_t, e_t, C) = \text{True} \quad (1)$$

Among them, the system state set is denoted as S, the event set as e, the environmental context as Ξ, the constraint set as C, T is the state transition function, expressing the next state of the system under given event and environmental conditions, Φ is the constraint satisfiability determination, such that the transition is only valid when both logical and business conditions are met, s_t is the state of the system at time t, s_{t+1} is the next state of the system after the action of the event, e_t is the event that occurred at time t, from the event set E, Φ(s_t, e_t, C) constraint satisfiability determination function, subject to indicates that the state transition is only allowed to be executed when the constraint conditions are met, which is used to ensure the semantic consistency of the system behavior.

To describe the structural dependencies of the cross-component call chain, A three-dimensional interaction tensor A is introduced to characterize the interaction intensity between components that evolves over time:

$$A_{i,j,t} = \Psi(v_i, v_j, s_t) \quad (2)$$

Among them, v_i and v_j are system components,Ψexpresses the probability of interaction occurrence or call intensity from component i to component j under state s_t, and A includes synchronous calls. Based on the state transition model and interaction tensor, a global-level test semantic reachability function can be defined to determine whether a certain test target g can be triggered through the event sequence E:

$$\mathcal{R}(g \mid S_0, \vec{E}) = \mathbb{P}(\exists ts.t.s_t \in \Omega(g) \wedge \Gamma(s_0, \ldots, s_t, \mathcal{A}) = \text{True}) \quad (3)$$

Among them, Ω(g) is the set of state conditions that trigger the target g, Γ is used to verify whether a series of state changes along the event sequence are consistent with the interaction structure, and I(·) is the indicator function used to determine whether the target is reachable. This reachability expression provides a clear direction and interpretability for test semantic generation, enabling the generative model to construct effective test inputs based on the system structure, constraints, and state evolution laws.

2.2. Test Semantic Generation Assisted by Large Models

Based on the above formal description, the generative model can be utilized to optimize the processing and understanding of formal extensions in generation, that is, to accept the semantic network composed of information such as system architecture, state, the order of invocation, business logic, and input constraints as background data, thereby generating test scenarios. Due to the complex business processes and hierarchical relationships of distributed systems, generative models can generate the main path scenarios and those hard-to-reach deep states by analyzing each business path and deriving equivalent scenarios, boundary scenarios or compound abnormal scenarios.

In semantic modeling, generative patterns can expand structured input into a sequential list of actual activities, including calling services, inserting parameters, interference, and timing, and have the ability to adaptively repair. For instance, if parameters are missing or conditions are ambiguous, it is possible to adaptively identify a value range or complete the missing steps. By using models based on multi-stage production, branch path transfer, and real-time response systems to expand the state space, a more complete and logically coherent test set can be obtained.

2.3. Parameter Space Generation and Constraint Solving Mechanism

For distributed complex systems, the focus of test generation lies in the design and modeling of parameter space models. Different combinations of business type requirements, workload intensity, environmental conditions, and potential problem areas can all lead to completely different system behaviors. Therefore, it is necessary to organize a regular representation of the parameter space. Generally speaking, the parameter space includes four aspects: domain, boundary, association, and implementation constraints. When setting up parameter domains, business consistency, context relationships, and system constraints should be comprehensively considered to ensure that the generated scenarios can be implemented persistently. For details, please refer to Table 1 below.

Table 1. Examples of pure data in parameter space.

Parameter ID	Type encoding	Minimum value	Maximum value	Default value	Dispersion degree	Constraint group ID	Dependent parameter ID
P01	1	0	3	1	1	C01	-
P02	2	100	5000	1200	50	C02	-
P03	1	0	5	2	1	C01	P01

P04	3	0	100	30	5	C03	-
P05	2	0	300	80	10	C02	P04
P06	1	1	10	3	1	C04	-
P07	3	10	200	50	5	C03	P05
P08	2	0	1	0	1	C05	-
P09	3	50	500	120	10	C04	P06
P10	2	20	2000	300	20	C05	P08

By analyzing the correlations, conflicts, exclusivity and constraints among parameters, and applying methods such as graphical solution, range guessing, reachability verification or random generation, the matching schemes of test parameters are found, and the generated semantics are covered, so that the actual operation process can cover the state space that occurs.

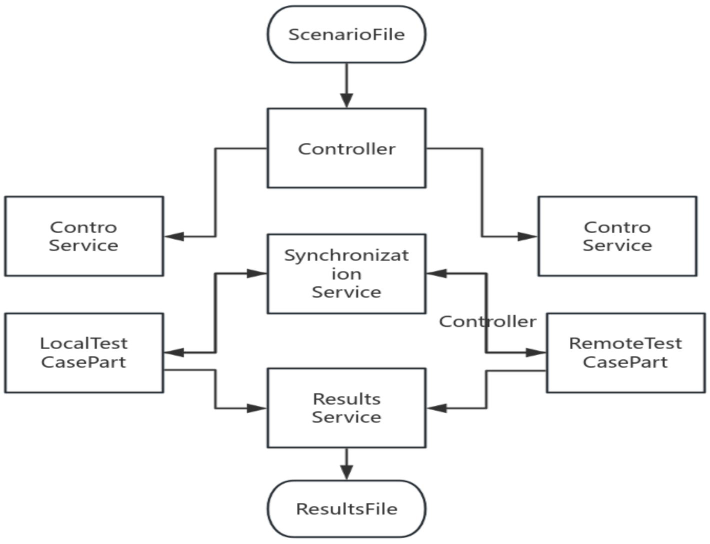

Figure 1. Schematic diagram of the generative test generation framework.

Figure 1 shows the logical structure of the overall framework for generative test generation and intelligent defect attribution. The dependencies among various functional modules and the flow of data are systematically presented in the figure.

3. An Intelligent Defect Attribution Method Based on Multi-Source Information Fusion

3.1. Unified Modeling of Multi-Source Observability Data

The time series granularity, event division and context scope of observability data are often inconsistent, which directly affects the continuous identification of abnormal relationships in the system and the construction of the propagation chain. To avoid information fragmentation, it is necessary to map the three core data types of logs, metrics and distributed tracking into a unified structural space, so that they can be aligned according to event boundaries, session cues or link fragments. A multi-source data set can be represented as:

$$O = \{L, M, Tr\} \tag{4}$$

Among them, L represents the log, M represents the sequence of metric vectors, and T_r represents the cross-node tracking link. Logs provide behavior descriptions, metrics record resource fluctuations, and tracking reflects call topologies and delay paths. The unified representation of data can be accomplished by vectorization and time alignment functions:

$$f(o_i) = \mathrm{Align}(\mathrm{Embed}(o_i)) \quad (5)$$

Here, o_i represents a single raw data item from the observability set O, $f(o_i)$ is the unified representation vector of the data item o_i, Embed(·) is the feature embedding function, Align(·) is the time and structure alignment function, and O is a multi-source observability data set, including the log set L, the metric set M, and the tracking set T_r. The i data item index is used to enumerate data instances from different sources and in different time slices.

3.2. Abnormal Semantic Understanding Assisted by Large Models

After achieving a consistent description, semantic fusion and classification processing need to be carried out to deal with different abnormal situations. In general, in practice, one can observe ambiguous expressions in system logs, unique proper nouns in components, and disorganized information. These content changes may not accurately represent the main issue, and traditional detection methods based on word and sentence alignment and threshold setting cannot effectively handle a large number of systems.

To handle unclear language descriptions, large models can aggregate event chains and contextual information for more thorough analysis. They not only analyze single events but also reason across nodes and related paths. By adding component identities, network structures, invocation paths, and expected behaviors to the prompts, large models can build cross-source anomaly maps and group subtle, related behaviors into a single anomaly category. Even when trace chains are missing, descriptions are brief, or data changes are small, the model can infer likely anomaly causes from context, maintaining stable understanding. Anomaly markers and propagation cues are then passed to the cause analysis module, providing strong semantic evidence for root-cause localization.

3.3. Precise Root Cause Localization Driven by Causal Inference

There are usually complex propagation relationships among multi-source abnormal events, and it is necessary to restore the causal chain from the system dependency structure. The causal structure of a distributed system can be abstracted as:

$$Gc = (V, Ec) \quad (6)$$

Among them, V represents components or logical nodes, and Ec represents causal edges formed by dependency relationships. If event a_i may trigger event a_j, the magnitude of its causal effect needs to be calculated:

$$\mathrm{CE}(a_i \to a_j) = P(a_j \mid \mathrm{do}(a_i)) - P(a_j \mid \mathrm{do}(\neg a_i)) \quad (7)$$

Here, P (·) represents the probability of an event occurring, $\mathrm{do}(a_i)$ is the intervention operation, $\mathrm{do}(\neg a_i)$ is the probability of event a_j occurring when event a_i is forcibly set not to happen in the cause-and-effect diagram, and P (a_j $\mathrm{do}(a_i)$) is the probability of event a_j occurring when event a_i is forcibly triggered. P(a_j $\mathrm{do}(\neg a_i)$) represents the probability of event a_j under the condition that event a_i is prevented from occurring, and $\mathrm{CE}(a_i \to a_j)$ is the causal effect. When inferring the root cause, it is necessary to integrate the node scores by combining the intensity of anomalies, causal effects and semantic consistency:

$$R(v) = \alpha \cdot I(v) + \beta \cdot CE(v) + \gamma \cdot C(v) \quad (8)$$

Among them, I(v) represents the anomaly intensity of the node, CE(v) represents its causal contribution, C(v) represents context consistency, and α,β,γ are the weights. In actual systems, abnormal propagation is often disturbed by implicit factors such as resource competition, caching strategies, and scheduling behaviors, resulting in weak correlations or indirect relationships among events. To identify these characteristics, the attribution mechanism needs to combine the dependency structure and semantic exception representation to construct a more robust propagation view at the cross-node and cross-time slice levels.

4. Prototype System Implementation and Experimental Verification

4.1. Generative Testing Semantics and Orchestration Mechanism

The prototype system receives input from the formal semantic model. By designing polymorphic detection environments, combinations of business grammar, interaction structures, and boundary conditions generate multi-level call routes, contextual information, edge-condition inputs, and classifications of chaotic behaviors. All generated semantic fragments undergo consistency checks before execution to ensure correct event links, variable types, and business rules.

The test execution architecture is built on a containerized environment, allowing adaptation to different test devices, construction of dialogue scenarios, and on-demand control of concurrent threads. It can map generated test semantics directly to real operating scenarios, producing executable steps. During execution, it supports path selection, traffic control, fault-point rotation, and response-delay injection, providing flexible stress and anomaly types.

4.2. Multi-source Observability Data and System Modeling

During test execution, the prototype system continuously collects logs, metrics, and traces and maps them into a unified event space. Because these data differ in timing accuracy and expression, cross-node and cross-level association is required. To integrate multiple event sequences into a consistent model, the system uses an event-fusion mechanism that aligns semantic, temporal, and contextual features, allowing all events to be analyzed within a unified representation space. This process can be defined as:

$$E_t = F(f(L_t), f(M_t), f(Tr_t)) \quad (9)$$

Among them, E_t is the fusion event representation of time slice t, f (·) is the aforementioned vectorization and alignment function, F (·) represents the fusion operation, L_t is the log data set of time slice t, M_t is the metric data of time slice t, and Tr_t is the distributed tracking fragment of time slice t. Through this formula, the system can integrate the originally scattered abnormal clues into consistent event nodes for subsequent abnormal semantic recognition and causal chain construction. This modeling mechanism enables the prototype system to maintain the stability of event representation under conditions of data noise, incomplete links or asynchronous sampling, thereby significantly enhancing the reliability of abnormal semantic understanding and root cause inference.

4.3. Causal Inference and root Cause Localization Algorithm

After obtaining structured abnormal events, the prototype system uses causal inference to restore the abnormal propagation chain and identify the most likely root cause nodes. Due to the complexity of the link and the presence of noise, in order to improve the positioning accuracy, the "propagation delay consistency" index is introduced, so that the candidate events need to simultaneously satisfy causal relationships and reasonable time patterns, in order to calculate a more reliable comprehensive root cause score. It can be defined as follows:

$$S(v) = R(v) + \lambda \cdot D(v) \quad (10)$$

Among them, S(v) is the comprehensive root cause score of node v, R(v) is the aforementioned causal score (including anomaly intensity, causal effect, context consistency, etc.), D(v) is the propagation delay consistency measure, which is used to measure whether the anomalies generated by the node and the downstream events conform to a reasonable propagation timing sequence, and$\lambda$is the parameter for adjusting the delay consistency weight. This formula makes the root cause candidates not only reasonable in terms of semantics and causality, but also conform to the law of abnormal propagation in terms of time patterns, thereby filtering out pseudo-associated nodes that only have statistical correlation. Through the joint inference of structured, semantic and temporal features, the prototype system can still accurately locate the key nodes that cause system degradation even in the case of complex links, high noise or incomplete data, improving the stability and credibility of diagnosis.

4.4. Distributed Execution and Fault Injection Experiment

To verify the feasibility of this method in different operational scenarios, this paper designs five typical scenarios: stable load, burst traffic, access skew, abnormal diffusion, and node failure. The covered paths include the central business path and some business paths with low correlation. Under pressure levels, potential performance bottlenecks can be identified. In scenarios of abnormal diffusion and node failure, the accuracy of large models and causal inference diagnosis is mainly examined.

As can be seen from Table 2, this method can explore the system state space more comprehensively and performs outstandingly in the reconstruction of abnormal propagation chains. Traditional methods often fail to restore the complete causal chain in cases of complex dependency relationships or missing data. However, this method, by integrating multi-source semantic understanding and causal inference, can maintain good performance under conditions of missing links or high noise.

Table 2. Comparison of Experimental Results (Test Generation and Defect Attribution).

Indicator	Traditional method	The method proposed in this paper	Increase the proportion
Test scenario coverage	62.3%	87.5%	+40.5%
Defect triggering capability	41.7%	69.2%	+65.9%
Root cause location accuracy	58.4%	82.1%	+40.6%
The degree of restoration of abnormal transmission chains	46.8%	78.3%	+67.4%
Stability of complex link diagnosis	Medium	high	-

5. Conclusion

The combination of generative semantic modeling, multi-source observability fusion, and causal inference provides a more efficient technical approach for the testing and diagnosis of distributed systems. Describe its structure and significance through the consistency method, so that the experiment can better cover the important operation processes and maintain stable support in high-pressure environments. The test results show that, compared with the traditional solutions, this method has achieved good results in both the consistency of abnormal discrimination and the correctness of the root cause, especially in the case of complex connections and data loss. With the continuous expansion of the system scale and the constant update of business models, self-adjustment capabilities, adaptability in dynamic environments, and cross-domain information acquisition will become the directions for further improvement. Future research can verify the scalability of the system proposed in this paper in a larger-scale environment.

References

1. G. Kambala, "Intelligent fault detection and self-healing architectures in distributed software systems for mission-critical applications," *International Journal of Scientific Research and Management (IJSRM)*, vol. 12, no. 10, pp. 1647-1657, 2024. doi: 10.18535/ijsrm/v12i10.ec11
2. R. Jai Ganesh, and S. Muralidharan, "Islanding detection system for grid connected photovoltaic system under different fault condition using intelligent detection method (IDM)," *Electric Power Components and Systems*, vol. 51, no. 13, pp. 1355-1366, 2023. doi: 10.1080/15325008.2023.2196680
3. G. Liu, W. Guo, X. Liu, and J. Xiong, "Security Analysis and Improvements on a Remote Integrity Checking Scheme for RegeneratingCodingBased Distributed Storage," *Security and Communication Networks*, vol. 2021, no. 1, p. 6652606, 2021.