Jaswanth Kumar Mandapatti[1], Srinivasarao Bandla[2], Vishnu Vardhan Reddy Kavuluri[3], Maheswara Rao Gorumutchu[4], Nareshkumar Jagadhabi[5]
[1]Advent Health, United States
[2]Deloitte Consulting LLP, United States
[3]Tata Consultancy Services, India
[4]HYR Global Source Inc, United States
[5]Compnova Inc, United States

Received: 22-06-2021; Revised: 26-08-2021; Accepted: 15-09-2021

Error Propagation Topologies in AI-Assisted Construction Pipelines

Abstract
AI-assisted software construction pipelines have introduced significant improvements in automation and development efficiency, but they also give rise to complex error propagation behaviors across multiple stages. Existing approaches primarily focus on detecting isolated errors, often overlooking how errors evolve, transform, and intensify as they move through interconnected pipeline stages. This study investigates error propagation as a topological flow problem, categorizing errors into syntactic, semantic, logical, dependency, and integration types, and analyzing their propagation intensity across code generation, compilation, testing, integration, and deployment stages. The results reveal that error propagation is highly stage-dependent, with testing and integration acting as major amplification points, while transformation processes convert errors into more complex forms, making detection and tracing challenging. A heatmap-based analysis further demonstrates the non-uniform distribution of propagation intensity, highlighting critical zones where errors escalate. The findings emphasize the need for stage-aware and transformationaware mitigation strategies to ensure reliability and robustness in AI-driven software pipelines.
Keywords: Error propagation, AI-assisted pipelines, software construction, topological analysis.

1. Introduction

The increasing adoption of AI-assisted software construction pipelines has fundamentally altered how software systems are designed, generated, and deployed. These pipelines integrate automated code generation, transformation, validation, and deployment processes, enabling rapid development cycles. However, the introduction of AI-driven components has also led to new forms of error propagation, where inaccuracies introduced at early stages can cascade through subsequent stages. Such propagation behaviors resemble adaptive system responses observed in computational intelligence models [1] and data-driven classification systems [2], where early-stage deviations significantly influence downstream outcomes.

Error propagation in AI-assisted pipelines is inherently multi-layered, involving interactions between syntactic correctness, semantic validity, and logical consistency. In early pipeline stages, such as code generation, minor inconsistencies may appear insignificant but can evolve into critical failures during later stages. This phenomenon is analogous to system variability observed in behavioral and environmental studies [3] and controlled experimental response systems [4], where small perturbations lead to amplified effects under iterative transformations. Understanding these propagation dynamics is essential for ensuring robustness in automated software workflows.

The complexity of propagation increases further due to the integration of machine learning models within the pipeline. These models introduce probabilistic decision-making, where outputs depend on learned patterns rather than deterministic rules. As seen in studies involving adaptive biological and computational systems [5] and large-scale data-driven environments [6], such probabilistic mechanisms can amplify uncertainties, making error tracing and correction significantly more challenging. This creates a need for structured frameworks to analyze how errors evolve across pipeline stages.

Another critical dimension of error propagation arises from structural transformations applied during software construction. Code rewriting, dependency resolution, and integration processes modify intermediate representations, often introducing additional layers of abstraction. These transformations can obscure the origin of errors, similar to structural variations observed in classification and transformation systems [7] and analytical processing pipelines [8]. As a result, identifying the root cause of errors becomes increasingly complex as they propagate through the pipeline.

The role of data and input variability further contributes to propagation dynamics. AI-assisted systems rely heavily on training data and input distributions, which may not always reflect real-world conditions. Variations in input data can lead to inconsistent outputs, thereby introducing errors that propagate across stages. This behavior parallels variability observed in genetic and microbial systems [9] and adaptive response mechanisms in pharmacological studies [10], where system outputs are highly sensitive to input conditions.

In addition to input variability, the feedback mechanisms embedded within AI-assisted pipelines introduce another layer of complexity. Continuous learning and iterative refinement processes allow systems to adapt over time, but they also create feedback loops where errors can be reinforced rather than corrected. This is comparable to adaptive resistance patterns observed in microbial systems [11] and fluctuating response profiles in dynamic environments [12], where repeated exposure leads to the persistence of undesirable behaviors.

The integration of multiple subsystems within software construction pipelines further amplifies error propagation. Each subsystem, whether responsible for code generation, testing, or deployment, operates under its own constraints and assumptions. When combined, these subsystems can produce emergent behaviors that are difficult to predict or control. Such interactions are similar to complex system behaviors observed in biomedical response modeling [13] and distributed acceptance systems [14], where interdependent components contribute to system-level variability.

This study addresses the challenge of error propagation in AI-assisted software construction pipelines by analyzing the topological structure of propagation patterns. It aims to identify how errors originate, propagate, and transform across pipeline stages, providing a foundation for designing more robust and traceable software construction systems. By focusing on the topology of error propagation, the work contributes to improving reliability, transparency, and control in automated software development environments.

2. Methodology

The proposed methodology models error propagation in AI-assisted software construction pipelines as a topological flow problem, where errors are treated as entities that traverse through interconnected pipeline stages. Each pipeline is represented as a directed structure consisting of nodes corresponding to processing stages and edges representing transformation or execution flows. This abstraction enables systematic tracking of how errors originate, evolve, and propagate across different stages of software construction.

The pipeline is decomposed into five primary stages: code generation, compilation, testing, integration, and deployment. Each stage performs distinct transformations on the software artifact and introduces unique opportunities for error injection or amplification. The methodology focuses on capturing stage-specific behaviors, ensuring that the propagation dynamics are analyzed not only in terms of occurrence but also in terms of structural impact across the pipeline.

Error types are categorized into syntactic, semantic, logical, dependency-related, and integration errors. Syntactic errors arise from incorrect code structure, semantic errors from invalid meaning or incorrect usage, logical errors from flawed algorithmic behavior, dependency errors from unresolved or

conflicting components, and integration errors from mismatches across subsystems. This classification allows the framework to distinguish between different propagation patterns and analyze how each error type behaves across stages.

To represent propagation behavior, the methodology introduces a topological mapping framework that associates each error type with its possible transition paths across pipeline stages. These paths define how errors move from one stage to another and whether they are amplified, suppressed, or transformed. For instance, a syntactic error may be detected and resolved during compilation, whereas a logical error may persist and propagate into later stages, affecting system behavior more significantly.

A key component of the methodology is the concept of propagation intensity, which reflects the degree to which an error influences downstream stages. Instead of relying on numerical equations, the framework evaluates intensity qualitatively based on factors such as error persistence, transformation complexity, and stage sensitivity. This allows the identification of critical stages where errors are most likely to amplify and cause system-level failures.

The methodology also incorporates error transformation rules, which describe how errors evolve as they pass through different stages. For example, a semantic inconsistency introduced during code generation may transform into a logical error during testing or an integration failure during deployment. These transformation rules are essential for understanding the non-linear nature of error propagation and for identifying indirect relationships between error types.

To capture real-world variability, the framework includes controlled pipeline simulations where different error scenarios are introduced at specific stages. By observing how these errors propagate under varying conditions, the methodology identifies recurring topological patterns such as linear propagation, branching propagation, and cascading failure chains. This simulation-based approach enables a deeper understanding of propagation behavior in complex and dynamic environments.

The evaluation process involves analyzing how different error types propagate under varying pipeline conditions using the mappings defined in Table 1. By correlating error origin, propagation path, and transformation behavior, the methodology identifies critical flow patterns that contribute to system instability. This analysis highlights which stages are most vulnerable to error amplification and which error types pose the greatest risk to overall system reliability.

Table 1. Error Propagation Types and Topological Flow Characteristics in AI-Assisted Pipelines

Error Type	Origin Stage	Propagation Path	Transformation Behavior	Topological Flow Characteristic
Syntactic	Code Generation	Generation → Compilation	Early detection or correction	Localized, low propagation
Semantic	Code Generation	Generation → Testing	Converts to logical inconsistency	Linear propagation

Logical	Testing	Testing → Integration	Persists across stages	High persistence, cascading effect
Dependency	Integration	Integration → Deployment	Expands across modules	Branching propagation
Integration	Integration	Integration → Deployment	Amplifies system mismatch	System-wide propagation
Syntactic	Compilation	Compilation → Testing	Minor impact if unresolved	Limited propagation
Semantic	Testing	Testing → Deployment	Escalates under complex logic	Amplified propagation
Logical	Code Generation	Generation → Deployment	Transforms across stages	Cascading propagation
Dependency	Testing	Testing → Integration	Spreads across components	Distributed propagation

Finally, the methodology integrates these insights into a unified topological framework that enables systematic tracking and analysis of error propagation. By combining stage decomposition, error classification, propagation mapping, and simulation-based evaluation, the approach provides a comprehensive understanding of how errors move through AI-assisted pipelines. This framework forms the basis for developing strategies to detect, isolate, and mitigate error propagation in automated software construction environments.

3. Results and Discussion

The evaluation of error propagation across AI-assisted software construction pipelines reveals that propagation behavior is highly dependent on both the stage of occurrence and the type of error introduced. Early-stage errors, particularly syntactic and semantic inconsistencies, tend to exhibit localized effects when detected promptly. However, when such errors bypass early validation mechanisms, they transition into more complex forms, increasing their propagation intensity in subsequent stages. This highlights the importance of early-stage validation in controlling downstream instability.

The distribution of error propagation intensity across pipeline stages is illustrated in Figure 1, where the heatmap captures the variation of propagation strength across different error types and construction stages. The visualization clearly indicates that error intensity is not uniform; instead, it is concentrated in specific stage–error combinations. High-intensity regions are observed in testing and integration stages, particularly for logical and dependency-related errors, suggesting that these stages act as amplification points within the pipeline.

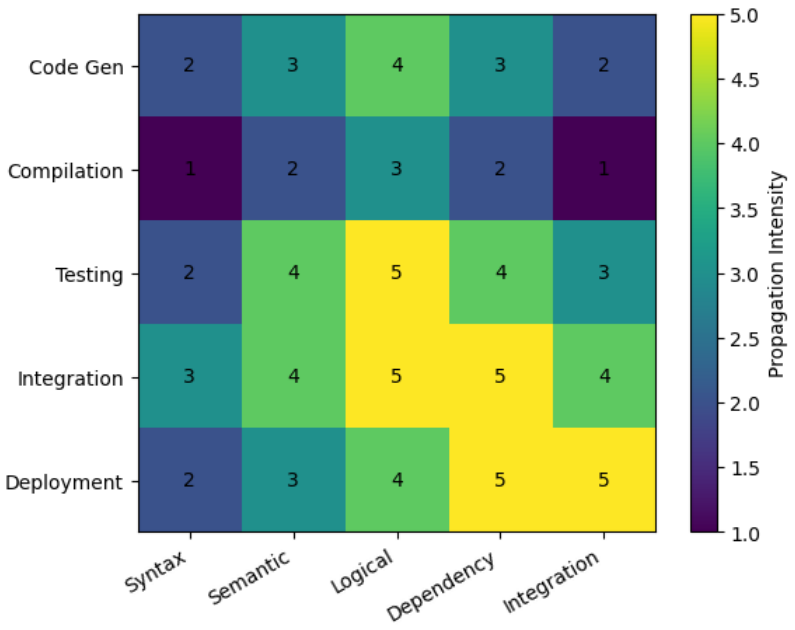

Figure 1. Error Propagation Intensity Across AI-Assisted Software Construction Stages

A closer analysis of the heatmap reveals that syntactic errors generally exhibit low propagation intensity, as they are often detected and resolved during compilation. In contrast, semantic and logical errors display moderate to high intensity, as they persist beyond initial validation stages and influence downstream processing. Dependency and integration errors show the highest intensity levels, particularly in later stages, where inter-component interactions amplify their impact across the system.

Another key observation is the emergence of propagation escalation patterns across consecutive stages. Errors introduced during code generation may initially appear minor but gradually intensify as they move through compilation, testing, and integration. This escalation is especially pronounced for logical errors, which often remain undetected until later stages, where their impact becomes more significant. The heatmap representation in Figure 1 effectively captures this progressive amplification behavior.

The results also highlight the role of transformation processes in shaping propagation dynamics. As software artifacts undergo multiple transformations, errors may change form while retaining their underlying impact. For example, a semantic inconsistency can evolve into a logical error during testing or manifest as an integration failure during deployment. This transformation-driven propagation underscores the non-linear nature of error flow in AI-assisted pipelines.

4. Conclusion

This study examined error propagation topologies in AI-assisted software construction pipelines by modeling how different types of errors evolve and intensify across pipeline stages. The findings demonstrate that error propagation is inherently multi-dimensional, governed by both the nature of the

error and the structural characteristics of the pipeline. Rather than being isolated events, errors exhibit topological flow patterns, where early-stage inconsistencies can cascade into more complex failures in later stages.

The results highlight that propagation intensity is not uniform across stages. Testing and integration layers were identified as critical amplification points, particularly for logical and dependency-related errors. While syntactic errors are often contained through early validation, higher-level inconsistencies tend to persist and evolve, significantly affecting downstream processes. This indicates that effective error control must prioritize deeper semantic and structural validation mechanisms rather than relying solely on early-stage checks.

Another key contribution of this work is the identification of transformation-driven propagation behavior. Errors do not remain static as they move through the pipeline; instead, they transform into different forms depending on the operations applied. This transformation complicates error detection and tracing, as the original source of failure may be obscured by subsequent processing stages. The topological framework proposed in this study provides a systematic way to capture these transformations and understand their impact on overall system behavior.

In conclusion, error propagation in AI-assisted pipelines must be addressed through a combination of early detection, stage-aware mitigation, and adaptive monitoring strategies. The insights from this study provide a foundation for designing more resilient software construction systems that can effectively manage propagation dynamics. Future work can focus on integrating real-time propagation tracking, automated error classification, and predictive mitigation techniques to enhance reliability and robustness in automated software development environments.

References

1. Velmurugan, C., Subramaniyan, V., Ilanthalir, S., Fuloria, S., Sekar, M., Fuloria, N. K., & Hasan Maziz, M. N. (2022). Evaluation of anti-diabetic and wound healing potential of Ethiopia plant'Ruta graveolens' in diabetic induced rat.
2. Vijayakumar, K., Maziz, M. N. H., Ramadasan, S., Balaji, G., & Prabha, S. (2024, May). Benign/Malignant Skin Melanoma Detection from Dermoscopy Images using Lightweight Deep Transfer Learning. In *2024 International Conference on Advances in Computing, Communication and Applied Informatics (ACCAI)* (pp. 1-5). IEEE.
3. Tien, L. P., Atiqah, N., Vytialingam, N., MA, R., Kabir, M. S., Shirin, L., ... & MHM, N. (2022). STRESS AND QUALITY OF LIFE AMONG FATHERS OF SPECIAL NEEDS CHILDREN IN KLANG VALLEY. *Journal of Pharmaceutical Negative Results*, *13*.
4. Ismail, S., Radu, S., Sidek, K., Arifffin, M. A., Maziz, M. N. H., Hamzah, I., & Abdulla, M. A. (2003). Effect of dexamethasone treatment on the hematological and histological parameters of mice following experimental bacterial infection. *J Anim Vet Adv*, *2*, 231-236.
5. MKK, F., Rashid, S. S., MHM, N., Baharudin, R., & Ramli, A. N. M. (2019). A clinical update on Antibiotic Resistance Gram-negative bacteria in Malaysia-a review. *arXiv preprint arXiv:1903.03486*.

6. Nazmul, M. H. M., Jamal, H., & Fazlul, M. K. K. (2012). Acinetobacter species-associated infections and their antibiotic susceptibility profiles in Malaysia. *Biomed Res-India*, *23*(4), 571-575.
7. Vijayakumar, K., Maziz, M. N. H., Ramadasan, S., Prabha, S., & Kumaar, K. S. N. (2024, May). Automatic classification of healthy/TB chest X-ray using DeepLearning. In *2024 International Conference on Advances in Computing, Communication and Applied Informatics (ACCAI)* (pp. 1-5). IEEE.
8. HasanMaziz, V. V. S. S., Ragavan, N. D., Arvind, C., Vairavan, S., & Neevashini, C. (2023). GC-Ms Analysis and Antibacterial Activity of Dryopteris Hirtipes (Blumze) Kuntze Linn. *Journal of Survey in Fisheries Sciences*, *10*(1S), 3718-3726.
9. Nazmul, M. H. M., Salmah, I., Jamal, H., & Ansary, A. (2008). Molecular characterization of verotoxin gene in enteropathogenic Escherichia coli isolated from Miri Hospital, Sarawak, Malaysia. *Biomed. Res*, *19*(1), 9-12.
10. Mkk, F., Sp, D., & Irfan, M. (2019). Antibacterial and antifungal activity of various extracts of Bacopa monnieri. *arXiv preprint arXiv:1909.01856*.
11. MKK, F., MA, R., & MHM, N. (2019). Detection of CTX-M-type ESBLs from Escherichia coli clinical isolates from a tertiary hospital, Malaysia. *Baghdad Science Journal*, *16*(3), 20.
12. Vijayakumar, K., Maziz, M. N. H., & Prabha, S. (2025, March). Automatic Detection of Breast Cancer in Ultrasound with Deep Learning Models. In *2025 International Conference on Frontier Technologies and Solutions (ICFTS)* (pp. 1-6). IEEE.
13. Maziz, M. N. H., Fazlul, M. K. K., Deepthi, S., Munirah, B., Farzana, Y., Najnin, A., & Srikumar, C. (2019). A study of comparison on knowledge and misconceptions about Hiv/Aids among students in a private university In Malaysia. *Malaysian Journal of Public Health Medicine*, *19*(1), 134-142.
14. Manzoor, M., Maziz, M. N. H., Subrimanyan, V., Shirin, L., Doustjalali, S. R., Sabet, N. S., ... & Mathialagan, A. (2022). Attitudes towards and the confidence in acceptance of telemedicine among the people in Sabah, Malaysia. *International Journal of Health Sciences*, *6*(S3), 2376-2386.