

(REVIEW ARTICLE)



DESIGNING RESILIENT ENTERPRISE SYSTEMS THROUGH AI-DRIVEN DEPENDENCY MAPPING AND IMPACT ANALYSIS

Pradip Baral *

West Bengal University of Technology, Kolkata, India.

* Corresponding Author

ORCID Details

Pradip Baral: <https://orcid.org/0009-0000-2721-5018>

World Journal of Advanced Engineering Technology and Sciences, 2026, 21(01), 186–195

Publication history: Received on 14 June 2026; revised on 12 October 2026; accepted on 14 October 2026

Article DOI: <https://doi.org/10.30574/wjaets.2026.21.1.0411>

Copyright © 2026 Author(s) retain the copyright of this article. This article is published under the terms of the Creative Commons Attribution License 4.0

ABSTRACT

Enterprise systems are now often spread across a range of hidden technical and organizational dependencies; technical debt can accumulate, configurations can change, and failures can propagate. Redundancy or post-incident recovery alone is not enough for resilience design; such environments require continuously updated dependency knowledge and credible impact analysis before, during, and after change. This review considers peer-reviewed journal literature published between 2015 and 2024 that is relevant to the dependency mapping and impact analysis of resilient enterprise systems using AI. The studies included in the review are grouped around microservice architecture, process mining, digital twins, graph learning, anomaly detection, explainable artificial intelligence and resilience measurement, as well as AIOps. Based on the literature, it is shown that AI-based mapping can increase observability of runtime relations, aid in change-risk prioritization, and aid in failure localization with architectural models and operational telemetry. There are still gaps in causal validation, cross-layer dependency representation, data quality, explainability and evaluation against real enterprise incidents. There is scholarly value in the field, as resilient design increasingly relies on dynamic, evidence-based knowledge of how enterprise elements behave under stress, change and degradation.

Keywords: AIOps; Dependency Mapping; Enterprise Resilience; Impact Analysis; Microservices; System Observability

1 INTRODUCTION

Resilient enterprise systems require the ability to predict, withstand, contain and rebound from disruption across software, data, infrastructure and organizational process layers. While previous studies on resilience have focused on quantitative measures of reliability, the concept has also been extended to consider absorptive, adaptive, and restorative capacity, which is considered a multidimensional phenomenon; the notion of engineering such capacity into dependency-rich digital systems is still not well understood, particularly within the context of enterprise software scholarship [1]. The implication of this gap is that modern enterprise systems seldom fail individually. Typical areas of failure include service calls, shared databases, message brokers, configuration repositories, identity services, third-party APIs, deployment pipelines, and business constraints. Dependency mapping and impact analysis are thus core design challenges, and not a separate documentation task.

With microservices and cloud-native architectures, this becomes more apparent. Journal studies show that decomposition makes it easier to deploy microservices independently and to scale up, while at the same time it introduces more coupling in operation, monitoring complexity, and runtime uncertainty. When dependency knowledge is incomplete, architectural freedom often results in resilience risk, as evidenced by the following: when implementing microservices, services get finer and finer, which creates problems involving interface instability, data ownership, and distributed debugging [3]. These results suggest that simply having a higher number of services, redundancy, or deployment automation is not sufficient to ensure enterprise resilience. Resilience relies on the ability of design actors to detect the impact on assets, workflows, controls and recovery when a node or dependency edge changes and/or a design assumption is violated.

In this regard, dependency mapping with AI addresses the challenge by deriving relations from traces, logs, event streams, code structure, configuration files, process logs, and architecture repositories. It has become even more important with the growing use of continuous integration, delivery and deployment methods that are increasing the pace of change and decreasing the time for manual impact assessment [4]. In this context, impact analysis should calculate blast radius, critical path exposure, recovery priority, compliance exposure, and latent coupling prior to deployment or incident escalation. Static architecture diagrams are insufficient when runtime paths change by tenant, by transaction, by traffic, by feature flag, by failure mode, or any combination of these.

The field of current research is interdisciplinary. Studies in software architecture inform on modularity, service boundaries, quality attributes, and how to move away from monoliths. One of the contributions of process mining is the event-log techniques used to uncover workflows and deviations from conformance. Digital twin research offers implementable models of industrial and cyber-physical assets. Computational methods are provided to learn dependency patterns, such as graph neural networks and anomaly-detection studies, from high-dimensional telemetry. Explainable AI research adds methods for explaining AI-generated impact assessments, which are essential for trustworthy impact decisions. AIOps research ties these threads together with incident management and failure localization. All these fields share the same issue: how to transform the operational evidence, which can be very heterogeneous, into the knowledge that is necessary for action to be taken about dependency structure and downstream effects.

This convergence is not enough to compensate for the fact that the study of the same topic in a peer-reviewed journal corpus is quite limited. The best literature is spread out in adjoining but related subfields, not in one well-developed subfield. Although process discovery studies show that event logs can be used to recreate the operational behavior, enterprise dependency mapping involves the interdependency of process, software, infrastructure and data lineage relations [5]. Microservice studies tend to focus on architectural impact models, and they tend to have fewer impact models that span multiple layers. While AI and graph-learning research deliver impressive computing capabilities, they do not always assess outcomes in relation to enterprise resilience metrics like recovery time, blast radius reduction, or degraded-mode continuity. This review then assesses the contribution of existing literature in the reviewed literature to the field of designing resilient enterprise systems using the dependency mapping and impact analysis techniques through the use of AI. The goal is to critically analyze evidence, identify patterns in methodology, understand the limitations, and create a brief conceptual and/or visual representation for future research.

2 LITERATURE REVIEW

There are four thematic areas in the literature that can be grouped around dependency mapping and impact analysis with the help of AI: dynamic representation of enterprise assets, computational inference of dependencies, interpretation of failure and change impact, and design control for resilience. Digital twin approaches offer a useful foundation for resilient enterprise design because they can maintain an operational representation of assets and their changing states. Tao et al. [6] define digital twins as “cyber-physical representations” which link the physical entity, virtual model, data and services and thus enable dependency reasoning to be more operational than documentation-based approaches. The above argument is further refined by distinguishing digital model, digital shadow, and digital twin forms by Jones et al. [7] which are important for enterprise resilience as they are only supported by higher levels of data integration that enable impact reasoning in time. While these studies demonstrate conceptual value, they also point to a limitation: in digital twin research, service calls, deployment pipelines, and data contracts and authentication chains are often treated too broadly, as they also play a significant role in enterprise software.

This representation agenda is extended in the literature of computational graph learning. Wu et al. [8] review the graph neural network methods that are able to explicitly model relational structure, neighborhood aggregation, and the prediction of node-level or graph-level properties. Zhou et al. also demonstrate the broader application of graph neural networks to non-Euclidean data, like dependency graphs [9]. Dependencies in enterprise systems are important because they are relational, directed, weighted, temporal and heterogeneous. The graph learning literature is however

often method-centric, not enterprise-resilience-centric. It focuses on predictive accuracy, representation power and scalability; impact analysis requires causal interpretation, governance and actionability.

A third cluster is related to anomaly detection and interpretability. Deep anomaly detection has been demonstrated to be able to capture high-dimensional deviation patterns, but is limited by the scarcity of labels, contaminated labels, evolving normality, and evaluation bias as reviewed by Pang et al. [10]. The same constraints apply to AIOps settings, with "noisy" telemetry, "rare" incidents, and post-release changes in system behavior. Studies by Guidotti et al. and by Barredo Arrieta et al. show that the fidelity, scope and human interpretability of black-box model explanations vary [11, 12]. Their relevance is direct: If the model for an enterprise predicts a high blast radius without saying which dependency path is causing the issue, then it is of little design value. The explanations of resilient design (where plural explanations are intended) needs to be provided at the services, workflows, data objects and recovery control levels.

A fourth cluster looks at resilience and critical infrastructure. The study by Rehak, Senovsky, Hromada and Lovecek shows a structured approach to assessing the critical infrastructure resilience, with a focus on functional continuity and the interdependency of components [13]. Zio maintains there are cascading dependencies, uncertainty and incompleteness of models that need to be considered in vulnerability and risk analysis of critical infrastructure [14]. While not exclusively focused on enterprise information systems, these studies also have conceptual value, as enterprise resilience is also a cascading-interdependency problem. The challenge of transferability is that critical infrastructure models may be less granular than software dependency models, and may not support short development cycles.

Studies that focus on microservices offer the best available architectural evidence. Li et al. show that quality attributes in microservice architecture are interdependent, with architectural decisions influencing system availability, maintainability, scalability, performance, security, and testability [15]. these attributes are independent of each other [15]. Eski and Buzluca show how to automatically break a monolithic application down into microservices and how they can be informed by dependency analysis and service boundary identification [16]. Jin et al. take this step further by identifying service candidates based on execution traces, where runtime behavior is the focus of the decomposition [17]. Waseem et al. provide further practitioner evidence of the close relationship between monitoring, testing and design challenges in microservice systems [18]. Notaro, Cardoso, and Gerndt's review paper [19] is an attempt to link AI techniques with failure management, identifying incident detection, diagnosis, localization, and remediation as interdependent steps. In these studies, the common drawback is that the dependency mapping is analyzed in isolation for architecture recovery, migration or incident diagnosis and resilient enterprise design requires a dependency model, impact model, action model, and feedback model.

Table 1: Summary of key findings

Ref	Focus	Key Findings
[6]	Digital twin representation	Connected physical, virtual, data, and service layers can support operational dependency reasoning.
[7]	Digital twin classification	Distinctions among model, shadow, and twin clarify whether dependency state is static, updated, or closed-loop.
[8]	Graph neural networks	Relational learning can encode dependency graphs but faces scalability, over smoothing, and interpretability concerns.
[9]	Graph learning applications	Graph models generalize to heterogeneous relations relevant to service, data, and infrastructure dependency structures.
[10]	Deep anomaly detection	High-dimensional anomaly modelling is promising but sensitive to labels, drift, contamination, and benchmark validity.
[11]	Black-box explanation	Explanations vary by fidelity and scope; local explanation may not equal system-level impact understanding.
[12]	Explainable AI taxonomy	Responsible AI requires transparent, interpretable, and accountable models for high-stakes operational decisions.
[13]	Infrastructure resilience assessment	Interdependence and functional continuity are central to resilience evaluation.
[14]	Vulnerability and risk analysis	Cascading effects and uncertainty remain major barriers to credible resilience modelling.
[15]	Microservice quality attributes	Quality attributes interact across architectural decisions, increasing need for dependency-aware design.

[16]	Monolith-to-microservice extraction	Automated extraction can reveal latent structural dependencies during migration.
[17]	Execution-trace decomposition	Runtime traces improve service candidate identification by exposing behavioural coupling.
[18]	Microservice practice	Monitoring, testing, and design are coupled operational concerns in real microservice systems.
[19]	AIOps failure management	AI can support detection, diagnosis, localization, and remediation, but integration remains uneven.

As can be seen from Table 1, there is no consensus in the literature on a model for dependency mapping and impact analysis using AI. Instead, it offers complementary building blocks. Digital twin studies provide representational continuity, graph studies provide relational learning, anomaly and explainability studies provide diagnostic intelligence, resilience studies provide cascading-risk concepts, and microservice studies provide architectural grounding. The central research question is where the two fields intersect – few articles in the reviewed literature examine the effect of the dependency maps created by AI on decisions in enterprise resilience design in realistic change and failure scenarios.

3 METHODOLOGY

The literature reviewed indicates that there are four interconnected methodological layers that support resilient enterprise design: dependency evidence acquisition, dependency inference, impact propagation, and resilience action selection. Evidence acquisition gathers signals from code repositories, runtime traces, logs, deployment descriptors, event logs, incident records, infrastructure telemetry, and configuration data. Process mining brings event-log discovery and conformance reasoning, whereas execution-trace approaches bring runtime coupling that is not present in source code and architecture descriptions [5, 17]. The advantage of these methods is that they are based on real behavior. They are limited by data incompleteness: lack of trace propagation, inconsistent logging, sampling bias, and undocumented external dependencies can affect their inferred maps.

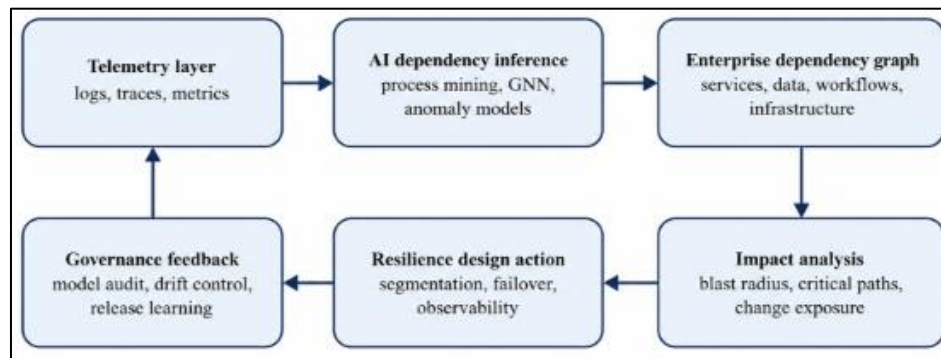
Evidence is transformed into a graph, a model, or a digital representation in dependency inference. While static extraction and architecture recovery techniques are useful when code and configuration can be extracted, runtime techniques are more effective for discovering conditional paths, temporal coupling and emergent dependencies. The concept of mapping is extended by digital twin approaches that include representation with operational state but software twins are still less developed than industrial twins [6, 7]. Graph neural networks provide expressive models for heterogeneous dependency graphs and can be used to solve node risk scoring, link prediction and anomaly propagation tasks [8, 9]. Their drawback is that learned embeddings can obscure the reasoning behind the significance of a particular dependency, reducing their usefulness in change advisory, compliance and incident command situations.

Impact propagation is the process of estimating what effects are likely to occur downstream of changes, failures or degraded components. Two major pathways are suggested in the literature. The first is rule-based or model-based propagation on known dependency graphs, which works for modelling a transparent blast radius but is sensitive to the incompleteness of maps. The second approach is AI-based inference using anomaly detection, graph representation learning or AIOps pipelines, which handles more complex runtime signals but is more difficult to validate causally [10, 19]. Explainable AI techniques can partially address this challenge by relating the results of a model to features, examples, rules or surrogate models [11, 12]. However, many explanation methods were not developed for enterprise dependency graphs where explanation needs to refer to causal sequences, impacted business capabilities, and recovery options.

The selection of resilience actions establishes a connection between impact analysis and design intervention. These may be service decoupling, circuit breaking, redundancy placement, data replication, deployment sequencing, monitoring enrichment, control-plane hardening, or degraded-mode workflow design. The literature on resilience reviewed suggests that it is important to take into account the interdependence, uncertainty and functional continuity of actions, not just component reliability [13, 14]. Quality studies of microservices also show that there are trade-offs among quality attributes between availability, maintainability, performance, and security in the architectural and operational aspects [15, 18]. Therefore, impact analysis has utility only when it is used to make specific design choices.

Table 2: Method comparison

Ref	Method	Strengths	Limitations
[5]	Event-log process discovery	Reveals operational workflow dependencies from observed events.	Weak at infrastructure and code-level relations without enrichment.
[6], [7]	Digital twin modelling	Maintains stateful representation for operational reasoning.	Enterprise software twins remain hard to maintain across fast release cycles.
[8], [9]	Graph neural network modelling	Learns relational patterns across complex dependency graphs.	Explanations and causal claims remain difficult.
[10]	Deep anomaly detection	Detects high-dimensional deviations in telemetry.	Label scarcity, drift, and contaminated normal data affect validity.
[11], [12]	Explainable AI	Improves interpretability of model outputs.	Local explanations may not show full cascade impact.
[16], [17]	Automated extraction and trace clustering	Supports architecture recovery and service boundary analysis.	Migration evidence may not generalize to live incident response.
[19]	AIOps failure-management pipelines	Connects detection, diagnosis, localization, and remediation.	Tool chains often remain fragmented across organizational silos.

**Figure 1: Dependency-aware resilience design framework**

In Figure 1, a topic-specific framework is provided that connects telemetry, AI-based dependency inference, enterprise graph construction, impact analysis, resilience design action, and governance feedback. This is important because resilient enterprise design requires a closed-loop approach rather than a static architecture repository. The framework conceptually integrates methodological patterns reported in [2, 19].

4 DISCUSSION

The analyzed research shows that while dependency mapping is a valuable part of enterprise resilience with AI as a tool, it needs to go beyond architecture inventory and extend to a dynamic, multi-layer impact reasoning capability. The trend which is most noticeable is the change from describing a static representation to a behavior-based one. Process mining and execution-trace analysis can help this transition as they have revealed that dependencies in observed event sequences and call sequences at runtime can uncover relations that are not captured in the design documents [5, 17]. The undocumented dependencies often dictate incident blast radius which is important to the enterprise resilience. Although it may look like a stand-alone payment service in a deployment diagram, it actually relies on other workflows such as identity, fraud scoring, streaming events, configuration, audit logging and reporting. It can be identified via AI-supported mapping, but the reviewed literature suggests that this operational coupling is incomplete without any one evidence stream.

This is confirmed by the microservice studies. Microservices offer advantages in terms of evolvability but come with costs in terms of challenges related to observability, testing, coordination, and distributed debugging, as demonstrated by Di Francesco and colleagues and Soldani and colleagues, respectively [2] and [3]. The implication for impact analysis is that architectural decomposition can shift the risk from the code level to the runtime dependency level. Of course, quality attributes are not independent, but rather interact with each other, Li and his colleagues add [15]. Redundancy

may require consistency, consistency may make redundancy more complicated, performance optimizations may make consistency harder to maintain, and failure isolation may be hard to achieve if performance is optimized. Multi-objective reasoning is a key element of AI-driven impact analysis. Factors such as data ownership, failover order, authentication, and compliance-critical workflows are not captured by dependency graphs that include only service calls.

A complementary result comes from the digital twin studies, where dependency maps gain more significance when connected to state change. Tao et al. [6] and Jones et al. [7] describe digital twin architectures based on interaction along dimensions of entity, model, data and service; and distinguish representations based on the extent of automated data flow respectively. This is an important difference. A static enterprise architecture model can help with planning, and a dynamic model is required for impact analysis during an incident. A digital shadow or twin can theoretically be a representation of evolving dependencies, degraded assets, and recovery alternatives. However, there is limited evidence regarding enterprise software twins that can integrate business capability maps, process logs, runtime service graphs, configuration drift, security posture, etc. in the literature so far. This is not merely a conceptual issue; it is also a practical maintenance challenge.

Another important trend is that of graph learning studies. Enterprise systems are better represented as heterogeneous nodes and edges in a dependency graph rather than as isolated records [8], [9]. A graph model can be used for critical node scoring, missing link inference, propagation pattern learning and abnormal subgraph classification. This helps with impact analysis by shifting from the presence of a dependency to its relevance. However, often graph-learning studies are judged by standard tasks that are not representative of enterprise resilience needs. To achieve enterprise impact analysis, three elements are needed: temporal validity, causal defensibility, and explainable prioritization in incomplete data. If a graph embedding is poor, then it may not be used in practice because it cannot differentiate between correlation and actual failure propagation.

This tension is reflected in anomaly detection literature as well. Deep anomaly detection can be applied to detect anomalies in logs, metrics and traces, particularly when the traditional thresholds are not sufficient [10]. Anomaly detection, however, is not synonymous with impact analysis. An abnormal latency pattern could be a symptom, a cause, or a harmless adaptation. Impact analysis must relate anomaly signals to dependency paths and affected enterprise capabilities. The literature on AIOps failure management is making strides towards this integration, where detection, diagnosis, localization, and remediation are linked [19]. One of its drawbacks is fragmentation; models focus on the aspects of the lifecycle of an incident without maintaining a traceable continuity of dependency evidence leading to design change.

Explainable AI is thus an essential component of this. Guidotti et al. argue that explanation methods can be local, global, model-specific, model-agnostic, rule-based, example-based or surrogate-based [11]. Barredo Arrieta et al. explains how explainability is related to responsible AI, transparency, and accountability [12]. Explanation has to be more than feature attribution in dependency mapping and impact analysis. It needs to explain why something is essential, how it becomes exposed, how it affects a business process, how it helps to mitigate risk and why there is evidence to support that. Though there are general principles for XAI, there are no mature explanation patterns on enterprise dependency graphs in the current literature.

Table 3: Results comparison

Ref	System / Context	Metric / Basis of Comparison	Outcome
[5]	Event-log process discovery	Review and benchmark of automated discovery algorithms	No single algorithm dominated across fitness, precision, generalization, and complexity.
[6]	Industrial digital twin architecture	Representation of physical, virtual, data, and service layers	Dependency reasoning improves when models connect data and services to operational assets.
[7]	Digital twin literature	Classification by data-flow relationship	Closed-loop representations provide stronger operational value than static models.
[8]	Graph neural network methods	Relational learning capability across graph tasks	Graph models support dependency inference but complicate explanation and scalability.
[10]	Deep anomaly detection	Review across data modalities and learning settings	Drift, labels, contamination, and benchmark bias limit operational reliability.
[13]	Critical infrastructure elements	Resilience assessment across functional dimensions	Interdependence must be included for credible resilience assessment.

[15]	Microservice quality attributes	Systematic review of quality concerns and tactics	Quality attributes interact, making single-metric dependency decisions inadequate.
[16]	Monolithic application migration	Automated extraction toward microservice structure	Structural dependency analysis can guide service boundary discovery.
[17]	Execution traces in monolithic systems	Runtime behaviour as service candidate evidence	Runtime coupling improves decomposition evidence beyond static structure alone.
[18]	Microservice practice	Practitioner evidence on design, monitoring, and testing	Operational resilience depends on combined design, observability, and validation practices.
[19]	AIOps failure management	Review of detection, diagnosis, localization, remediation	AI supports incident handling, but end-to-end integration remains immature.

The journal evidence is summarized in Figure 2, which shows the distribution of the selected journal evidence by topic-relevant clusters. As depicted in the figure, the reviewed literature spans several research areas, with the largest number of studies focusing on microservice architecture, while fewer studies address process mining and AIOps failure management.

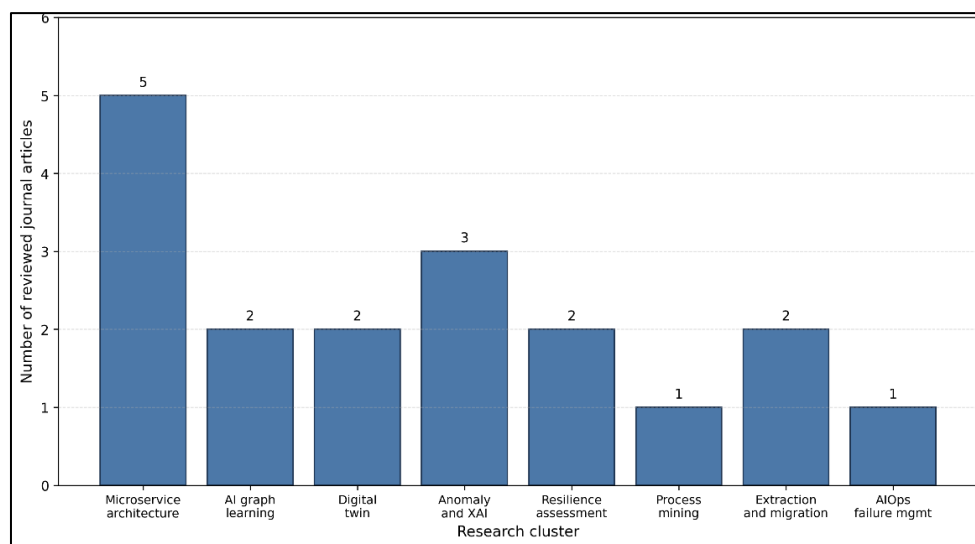


Figure 2: Evidence distribution across reviewed research clusters

the patterns presented in Table 3 and Figure 2 reveal that there are strong elements in literature but limited integration across research areas. Architecture and decomposition are the focus of microservice and migration studies; inference is the focus of AI studies; interdependence is the focus of resilience studies; and incident handling is the focus of AIOps studies. The challenge is to build this out into a structured pipeline. Dependency maps are an essential component of a resilient enterprise system that are operationally timed, explainable, governed and accurate. This is an essential requirement to address resilience impacts, rather than simply monitoring or architecture documentation. In Figure 3, the major methodological families are compared to resilience design objectives. It demonstrates that no single method is as effective as the others when it comes to all the objectives, which is why enterprise impact analysis needs to include static, runtime, AI, and governance methods. The methodological strengths and limitations reported in [5]–[19] are reported using interpretive coding and given scores ranging from 0 to 3, with 3 indicating the highest relevance.

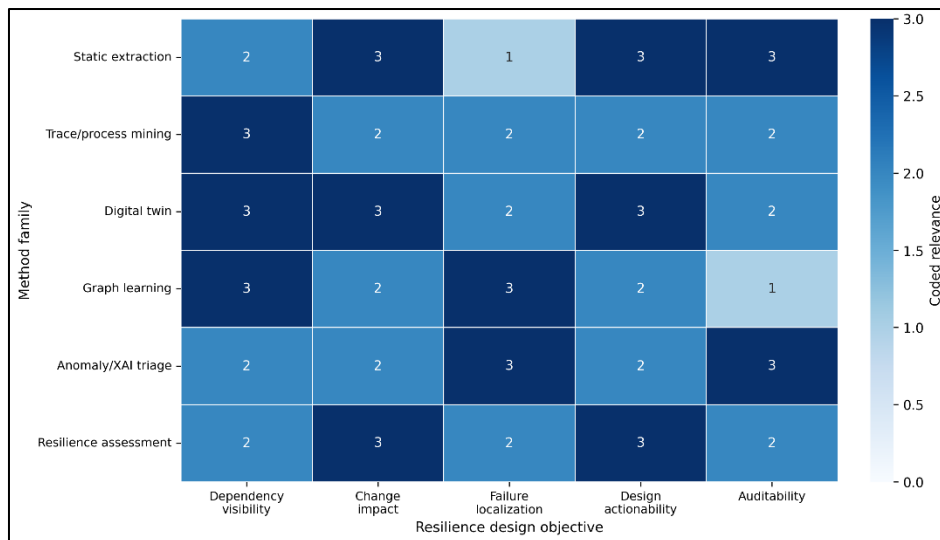


Figure 3: Method-objective relevance matrix for dependency-aware resilience

Significant contradictions also exist in literature. While microservice research is sometimes touted as offering design benefits like decentralization and independent deployment, resilience analysis reveals that this is not always the case, that is, deployment independence does not imply independence at runtime. There is a focus on faithful representation in digital twin research and continuous delivery brings drift that might outpace model maintenance. Learning from graphs offers the possibility of predicting complex relational structures while explainability research highlights that black-box models might be unsuitable for important operational decisions. Resilience action will need causal and architectural interpretation, while anomaly detection will be able to detect unusual behavior. The conflicts highlight that enterprise resilience will not be realized by simply plugging AI tools into the current monitoring platforms without rethinking the knowledge model that connects evidence, dependency, impact and response. The methodological families are plotted in Figure 4, where their intensity of observability and architectural granularity are the axes, and the size of the bubbles reflects the actionability for resilience design. The figure is important because the choice of method depends on what kind of design problem is to be addressed, whether it is a local component analysis, runtime diagnosis or impact governance at the enterprise level. Qualitative coding of the Tables 1-3 and methodological emphasis reported in [5]–[19].

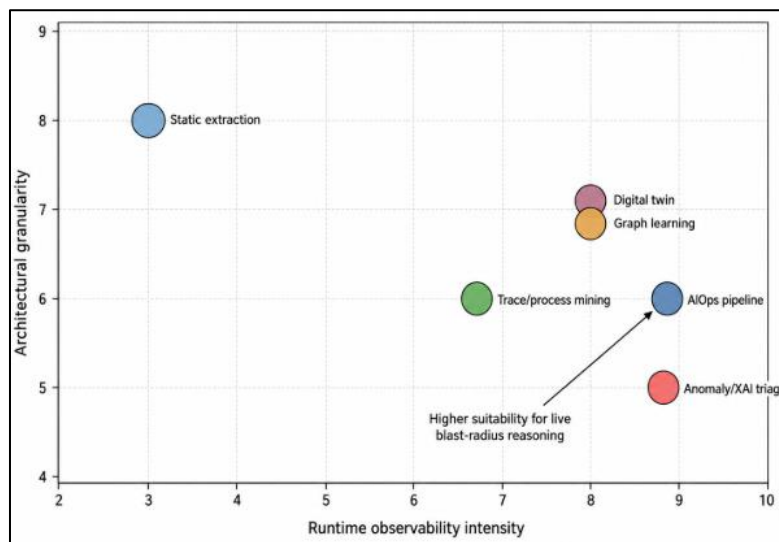


Figure 4: Positioning of dependency-mapping methods for enterprise resilience design

An important implication is that dependency mapping needs to be built on AI rather than being an add-on to resilience design. Design capability has four requirements. First, it has to be a mix of static and dynamic evidence, since neither is sufficient individually. Second, it must represent the type, direction, criticality and timeliness of the dependency. Third, it needs to link technical impact, compliance requirements, and recovery priorities to business capabilities. Fourth, it needs to facilitate explanation and governance to allow decisions to be challenged, audited and improved following

incidents. Parts of this capability are supported by existing studies but there is currently a lack of validated end-to-end evaluations across real enterprise change and incident environments.

Future Directions

There is a need for further research on causally credible impact analysis. There is considerable evidence of correlations between the different telemetry streams, but for resilient design, more evidence on how things propagate is required. Research should integrate dependency graphs with causal discovery, intervention testing, and incident replay to enable comparisons between blast radius predictions and observed failure paths. This should include false positives and false negatives, as false alarms can cause problems in operational trust as much as a lack of impact.

Multi-layer dependency representation is a second priority. The focus of current studies is predominantly on software architecture, process logs, digital twins or anomaly signals individually. Integrated graphs that link services, data stores, workflows, infrastructure, identity controls, deployment artifacts and business capabilities are needed for enterprise resilience. Future models should model heterogeneous edge types, directionality, criticality, confidence, temporal validity. Graph neural networks can be used to fill in missing links, but to make the models more interpretable, the methods must turn the model output into dependency paths and design decisions instead of abstract feature scores [8, 11, 12].

A third is the use of evaluation benchmarks based on enterprise incidents. Surveys, conceptual taxonomies, benchmark datasets or migration cases are often used in existing journal research. These are useful but insufficient to validate resilience outcomes. Future research should evaluate whether deploying AI tools for dependency mapping decreases recovery time, narrow the blast radius, enhance change-risk triage, improve incident localization accuracy, or improve degraded-mode continuity. Release events, configuration changes, partial outages, security-control failures, and data-pipeline degradation are among the instances to consider during evaluation.

A fourth direction concerns, governance of dependency intelligence. Dependency maps can become outdated, biased, incomplete, or overconfident. Future work should include the specification of model drift controls, confidence scoring, data lineage, human review points and accountability mechanisms for automated impact analysis. AIOps pipelines must be automated, but also institutionalized throughout the architecture, operations, risk, security, and compliance teams [18], [19].

Design integration is the last priority. AI-generated maps need to inform actual actions for resilience, like service segmentation, redundancy placement, circuit-breaker configuration, canary deployment scope, observability enrichment, and recovery sequencing. Studies that relate the impact of AI inference to architectural decisions and then test for resilience improvements under realistic enterprise constraints will make the strongest contribution in the future.

5 CONCLUSION

Reliable, dynamic and explainable dependency structure and impact behavior are essential when designing resilient enterprise systems. The literature reviewed indicates that there are already good foundations in the fields of microservice architecture, process mining, digital twins, graph learning, anomaly detection, explainable AI, resilience assessment and AIOps. While each domain makes a significant contribution, the field is still fragmented. Structural coupling can be revealed by static architecture recovery, runtime traces can help reveal behavioral relations, digital twins can help maintain the operational state, graph learning can help infer complex patterns, and explainable models can help facilitate accountable interpretation. However, these elements alone are insufficient.

The central scholarly insight is that the resilience design process needs to be a closed loop, from evidence to dependency inference to impact analysis to design action to governance. Enterprise resilience is not a single-dimensional property. It must be designed with a continuous understanding of how services, data, workflows, infrastructure and controls react to change and stress. Repeatedly identified areas for improvement include causal validation, cross-layer representation, real incident evaluation, model transparency, and organizational integration. Methods that can predict dependency risks, explain propagation paths, and support concrete architectural decisions will be critical for future progress. The field shows considerable potential, but results from empirical studies showing tangible resilience benefits in complex enterprise environments will be important for the field's maturity.

REFERENCES

- [1] Hosseini, S., Barker, K., & Ramirez-Marquez, J. E. (2016). A review of definitions and measures of system resilience. *Reliability Engineering & System Safety*, 145, 47–61.
- [2] Di Francesco, P., Lago, P., & Malavolta, I. (2019). Architecting with microservices: A systematic mapping study. *Journal of Systems and Software*, 150, 77–97.
- [3] Soldani, J., Tamburri, D. A., & Van Den Heuvel, W. J. (2018). The pains and gains of microservices: A systematic grey literature review. *Journal of Systems and Software*, 146, 215–232.
- [4] Shahin, M., Babar, M. A., & Zhu, L. (2017). Continuous integration, delivery and deployment: A systematic review on approaches, tools, challenges and practices. *IEEE Access*, 5, 3909–3943.
- [5] Augusto, A., Conforti, R., Dumas, M., La Rosa, M., & Polyvyanyy, A. (2019). Automated discovery of process models from event logs: Review and benchmark. *IEEE Transactions on Knowledge and Data Engineering*, 31(4), 686–705.
- [6] Tao, F., Zhang, H., Liu, A., & Nee, A. Y. C. (2019). Digital twin in industry: State-of-the-art. *IEEE Transactions on Industrial Informatics*, 15(4), 2405–2415.
- [7] Jones, D., Snider, C., Nassehi, A., Yon, J., & Hicks, B. (2020). Characterising the digital twin: A systematic literature review. *CIRP Journal of Manufacturing Science and Technology*, 29, 36–52.
- [8] Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., & Yu, P. S. (2021). A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32(1), 4–24.
- [9] Zhou, J., Cui, G., Hu, S., Zhang, Z., Yang, C., Liu, Z., Wang, L., Li, C., & Sun, M. (2020). Graph neural networks: A review of methods and applications. *AI Open*, 1, 57–81.
- [10] Pang, G., Shen, C., Cao, L., & van den Hengel, A. (2021). Deep learning for anomaly detection: A review. *ACM Computing Surveys*, 54(2), Article 38.
- [11] Guidotti, R., Monreale, A., Ruggieri, S., Turini, F., Giannotti, F., & Pedreschi, D. (2018). A survey of methods for explaining black box models. *ACM Computing Surveys*, 51(5), Article 93.
- [12] Barredo Arrieta, A., Díaz-Rodríguez, N., Del Ser, J., Bennetot, A., Tabik, S., Barbado, A., García, S., Gil-López, S., Molina, D., Benjamins, R., Chatila, R., & Herrera, F. (2020). Explainable artificial intelligence: Concepts, taxonomies, opportunities and challenges toward responsible AI. *Information Fusion*, 58, 82–115.
- [13] Rehak, D., Senovsky, P., Hromada, M., & Lovecek, T. (2019). Complex approach to assessing resilience of critical infrastructure elements. *International Journal of Critical Infrastructure Protection*, 25, 125–138.
- [14] Zio, E. (2016). Challenges in the vulnerability and risk analysis of critical infrastructures. *Reliability Engineering & System Safety*, 152, 137–150.
- [15] Li, S., Zhang, H., Jia, Z., Li, Z., Zhang, C., Li, J., Gao, Q., Ge, J., & Shan, Z. (2021). Understanding and addressing quality attributes of microservices architecture: A systematic literature review. *Information and Software Technology*, 131, Article 106449.
- [16] Eski, S., & Buzluca, F. (2018). An automatic extraction approach: Transition to microservices architecture from monolithic application. *Applied Sciences*, 8(6), Article 992.
- [17] Jin, W., Liu, T., Cai, Y., Kazman, R., Mo, R., & Zheng, Q. (2021). Service candidate identification from monolithic systems based on execution traces. *IEEE Transactions on Software Engineering*, 47(5), 987–1007.
- [18] Waseem, M., Liang, P., Shahin, M., Di Salle, A., & Márquez, G. (2021). Design, monitoring, and testing of microservices systems: The practitioners' perspective. *Journal of Systems and Software*, 182, Article 111061.
- [19] Notaro, P., Cardoso, J., & Gerndt, M. (2021). A survey of AIOps methods for failure management. *ACM Transactions on Intelligent Systems and Technology*, 12(6), Article 75