Business Risk Analytics for AI-Generated Security Patches: Measuring Oracle Divergence in Enterprise Software Workflows

Isabel Navarro[1], Miguel Ángel Rojas[2], Clara Benítez[3,*]

[1]Department of Information Systems, University of Murcia, Murcia 30100, Spain

[2]Department of Business Organization and Marketing, University of Vigo, Vigo 36310, Spain

[3]Department of Computer Science, University of Castilla-La Mancha, Albacete 02071, Spain

*Email: clara.benitez@uclm.es (Corresponding Author)

Abstract

AI-generated security patches are increasingly embedded in software engineering workflows, yet many enterprise release decisions still rely on narrow validation signals such as successful compilation, functional-test passage, static-warning disappearance, or blocking of a single demonstrated exploit. These signals reduce local uncertainty but do not necessarily measure whether a patch removes the vulnerability root cause, resists exploit variants, and preserves business-critical services after release. This article develops a business risk analytics framework for evaluating AI-generated security patches in enterprise software workflows. The framework translates cross-oracle divergence into business risk indicators by linking patch-level validation outcomes to release exposure, remediation delay, operational interruption, compliance risk, and security debt accumulation. A controlled enterprise-style analytical experiment is constructed with 480 AI-generated candidate patches across four vulnerability families: SQL injection, path traversal, cross-site scripting, and missing authorization. Each patch is evaluated across six validation layers: build validity, functional preservation, original proof-of-concept blocking, exploit-variant resistance, root-cause conformance, and regression-safety review. The results show that weak-oracle acceptance substantially overstates business-safe release readiness. In the constructed dataset, 76.7% of patches block the original exploit evidence, whereas only 47.7% satisfy the release-acceptance protocol after variant, root-cause, and regression checks. Oracle divergence is highest for path traversal and missing authorization, where business risk arises from exploit-specific blocking and incomplete trust-boundary repair. The article contributes a managerial analytics model that separates apparent technical repair from release-grade assurance, demonstrates how patch-level validation data can be converted into portfolio risk metrics, and provides governance guidance for enterprises adopting LLM-based vulnerability repair in CI/CD and DevSecOps environments.

Keywords: AI-generated security patches; business risk analytics; oracle divergence; enterprise software workflows; LLM-based vulnerability repair; DevSecOps; exploit variants; root-cause conformance; software security governance

Article History:

Received: July 23, 2023

Revised: September 28, 2023

Accepted: November 21, 2023

 https://doi.org/10.63646/jbda.2023.010405

Available Online: December 30, 2023

Business Risk Analytics for AI-Generated Security Patches: Measuring Oracle Divergence in Enterprise Software Workflows

1. Introduction

Enterprise software organizations are beginning to use large language models and other generative AI systems to propose code patches for security vulnerabilities. The practical attraction is straightforward: engineering teams face constant pressure to remediate vulnerabilities faster than manual triage, code review, and release cycles traditionally allow. AI-generated repair promises shorter response times, broader candidate generation, and lower developer workload. In business terms, the value proposition is not merely technical productivity. It is the possibility of reducing breach exposure, shortening vulnerability dwell time, and improving the resilience of digital operations that support revenue, customer trust, and regulatory compliance (Arora et al.,2010). Enterprise adoption of automated patching is therefore inseparable from the wider movement toward AI-enabled financial and operational decision systems (Yang et al.,2025). The underlying investment problem resembles classic information-security economics, where firms must balance the marginal cost of assurance against the expected reduction in loss exposure (Gordon and Loeb,2002).

Yet the same speed that makes AI-based patching attractive also introduces a new management problem. An automatically generated patch may look successful because it compiles, passes available tests, removes a static-analysis warning, or blocks the original proof-of-concept exploit. These validation signals are useful, but they are not equivalent to business-safe repair. A patch that blocks one exploit string while leaving semantically equivalent variants open can create a false sense of readiness. A patch that silences an analyzer warning without changing the unsafe data flow can move risk from the security dashboard into production. A patch that prevents exploitation by disabling legitimate behavior can create service disruption, customer dissatisfaction, and manual rollback costs. These failure modes transform a technical validation problem into a business risk analytics problem (Böhme,2010). This concern is consistent with business analytics research showing that operational indicators become useful only when they are connected to decision consequences (Chen et al.,2012). Recent software engineering evidence similarly shows that large language models can produce plausible repairs while still requiring strong validation around behavioral correctness (Xia et al.,2023).

The uploaded source manuscript motivating this article formulates the issue as security patch overfitting. It shows that apparent success under weak validation evidence can diverge from stronger evidence such as exploit-variant resistance, CWE-specific root-cause conformance, and regression safety. The manuscript evaluates generated patches through layered validation and reports that original proof-of-concept blocking can substantially overestimate full-protocol security-fix acceptance. This article does not reproduce that manuscript. Instead, it reinterprets the same research direction for the Journal of Business and Data Analytics by asking how oracle divergence should be measured, monetized, and governed inside enterprise software workflows. The same logic appears in Web 3.0 governance debates, where technical verification must be translated into institutionally meaningful trust signals (Zhang and Lu,2025). Security investment

 https://doi.org/10.63646/jbda.2023.010405

scholarship also warns that private incentives and organizational visibility may not align with system-level resilience (Anderson and Moore,2006).

The central argument is that organizations should not treat AI-generated patch acceptance as a binary engineering decision. It should be treated as a portfolio risk decision. Each validation oracle provides a partial signal. Build validity tells managers that the patched code can execute. Functional tests show whether observed benign behavior is preserved. Original exploit blocking shows whether a known demonstrated attack path is closed. Variant resistance indicates whether the fix generalizes across equivalent attack forms. Root-cause conformance indicates whether the underlying security mechanism has changed in the expected direction. Regression-safety review indicates whether the fix introduces new weaknesses or breaks adjacent controls. Business risk analytics is needed because the difference between these signals determines release exposure. From a data-analytics perspective, this framing follows the broader movement from descriptive monitoring toward decision-oriented operational intelligence (Choi et al.,2018). Code-focused LLM evaluations further indicate that syntactic fluency does not by itself establish software reliability (Chen et al.,2021).

This article proposes Business Oracle Divergence Analytics (BODA), a framework that converts patch-level validation outcomes into enterprise risk indicators. BODA organizes validation evidence into an assurance ladder, estimates the probability that a patch accepted by a weak oracle will fail a stronger oracle, and maps this divergence to expected business consequences. The framework is designed for use in CI/CD, DevSecOps, software product governance, and vulnerability management workflows. It does not require a new patch-generation model. Its contribution lies in decision analytics: it tells managers when an AI-generated patch is likely safe enough to release, when it requires additional review, and when weak-oracle acceptance masks unacceptable business risk. FinTech research provides a useful analogy because risk analytics in financial systems must also reconcile automation, trust, and governance (Kou and Lu,2025). Empirical work on security patching shows that remediation decisions are shaped by competition, timing, and incentives rather than by technical severity alone (Arora et al.,2010).

The article makes four contributions. First, it reframes security patch overfitting as a business analytics problem involving risk leakage from technical validation into enterprise workflows. Second, it develops a layered evidence model that converts oracle divergence into release-readiness and residual-risk indicators. Third, it presents a controlled enterprise-style data analysis of 480 AI-generated candidate patches, showing how acceptance rates change as validation moves from weak signals to release-grade evidence. Fourth, it provides managerial implications for software leaders, chief information security officers, audit teams, and platform managers who must balance remediation speed against assurance quality. The contribution also aligns with evidence that analytics capabilities create value when they are embedded in organizational routines rather than treated as isolated technical assets (Mikalef et al.,2020). Foundation code models increase the urgency of this issue because they make patch generation easier while leaving validation and accountability unresolved (Roziere et al.,2023).

The remainder of the article is organized as follows. Section 2 reviews relevant literature on AI-based software repair, vulnerability management, software risk economics, and analytics-driven governance. Section 3 presents the conceptual framework. Section 4 describes the analytical dataset and variables. Section 5 reports the results. Section 6 develops business risk and managerial implications. Section 7 discusses governance integration. Section 8 concludes the article.

2. Literature Review

Automated program repair has long aimed to generate patches that make defective software pass a set of tests. Early work demonstrated that search-based and generate-and-validate methods can produce plausible repairs, while also revealing a persistent limitation: a patch that passes available tests may not be semantically correct (Le Goues et al.,2012). This distinction between plausible and correct patches is central to the present article because enterprise release decisions often inherit the same weakness. When a test suite is incomplete, a patch may satisfy observable evidence while violating untested business or security behavior (Qi et al.,2015). Industry 4.0 research emphasizes that digital systems create value only when technical integration is paired with process governance (Lu,2025). Security metrics research similarly cautions that measurement design determines whether controls produce real assurance or only superficial compliance (Böhme,2010).

The rise of large language models has intensified this issue. LLMs are strong at producing syntactically plausible code and explaining common secure-coding patterns, but plausibility is not equivalent to assurance. In software engineering, LLMs have been used for code generation, code summarization, vulnerability explanation, and bug repair (Chen et al.,2021). However, the reliability of generated code depends on validation context, prompt design, repository state, and security requirements that may not be visible in the prompt. For security patches, the cost of a plausible but incomplete fix is greater than the cost of an ordinary incorrect suggestion because the organization may close the ticket, ship the release, and leave the attack surface active. Big data studies show that predictive value depends on the organizational capability to transform heterogeneous evidence into repeatable action (Wamba et al.,2017). Open foundation models further increase the range of possible generated repairs, which makes downstream validation pipelines more important than ever (Touvron et al.,2023).

Vulnerability management research has traditionally measured risk using severity scores, exploit availability, asset criticality, and remediation time. The Common Vulnerability Scoring System has become a widely used baseline for communicating technical severity, but it does not directly measure whether a specific generated patch has removed the root cause in a given codebase (Jacobs et al.,2020). Enterprise risk frameworks therefore need to combine vulnerability severity with patch validation evidence. A high-severity vulnerability patched by a weakly validated AI-generated fix may remain more dangerous than a moderate vulnerability patched with strong root-cause evidence. Blockchain-enabled internal auditing research is relevant because it treats evidence integrity as a governance asset rather than a purely technical property (Wu et al.,2025). Cyber supply-chain guidance likewise stresses that organizations should evaluate suppliers and software artifacts through documented, risk-informed evidence records (NIST,2022).

Cybersecurity economics provides an important lens for this problem. Firms invest in security under uncertainty, incomplete information, externalities, and bounded verification capacity (Gordon and Loeb,2002). The cost of a weakly validated patch includes not only the expected breach loss if exploitation remains possible, but also the cost of rework, incident response, compliance review, delayed release, and loss of engineering trust. In this sense, oracle divergence is a measurable form of information risk: the enterprise believes a patch is safer than it actually is because the chosen validation signal is too permissive (Anderson and Moore,2006). Customer and operational analytics research shows that strategic value emerges from integrating data into managerial decision processes (Kitchens et al.,2018). General-purpose model evaluations also show that capability claims should be interpreted cautiously when validation settings differ from deployment settings (Achiam et al.,2023).

 https://doi.org/10.63646/jbda.2023.010405

Business analytics research has increasingly emphasized the transformation of operational logs into decision metrics. In software organizations, this transformation is visible in DevOps metrics such as deployment frequency, lead time, change failure rate, and mean time to recovery (Forsgren et al.,2018). Security work adds new metrics such as vulnerability age, remediation backlog, exploit exposure, and control coverage. AI-generated patch workflows require one more class of metric: validation strength. Without a validation-strength metric, teams may optimize for remediation speed while silently increasing residual risk. Quantum machine learning research reinforces the broader point that advanced computational methods require classification frameworks, constraints, and evaluation criteria before they become manageable business tools (Lu et al.,2024). Information systems success theory similarly suggests that system quality and information quality must be evaluated together when technology affects organizational outcomes (DeLone and McLean,2003).

DevSecOps practices attempt to integrate security checks into development pipelines rather than treating security as a final gate. Static analysis, software composition analysis, dynamic testing, infrastructure-as-code scanning, and secrets detection are commonly used in automated pipelines (Rahman and Williams,2016). These tools are valuable, but they also create the possibility of analyzer-facing optimization. If teams or models learn to satisfy the tool rather than the security invariant, the dashboard improves while the root cause remains. This pattern is structurally similar to Goodhart's law: when a metric becomes the target, it can lose its value as a measure (Chen et al.,2012). Dynamic capability theory explains why the same AI patching tool can generate different business value depending on organizational sensing, learning, and reconfiguration routines (Teece,2007). Automatic repair research has long shown that generated fixes require correctness assessment beyond simple patch production (Le Goues et al.,2012).

Research on technical debt also helps explain why weakly validated AI patches are a business concern. A patch that superficially removes a symptom but fails to repair the underlying control may add security debt: the apparent issue is closed, but future maintenance, auditing, and incident response become more difficult. Security debt differs from ordinary technical debt because it may remain invisible until exploited. It therefore requires evidence-based tracking, especially in enterprises where many teams contribute to shared services, APIs, and platform components (Li et al.,2015). Quantum finance research highlights the need to translate technically complex mechanisms into decision structures that executives and regulators can evaluate (Lu and Yang,2024). Security behavior research also demonstrates that formal controls work best when perceived effectiveness and organizational pressure support compliant practice (Herath and Rao,2009).

The literature on software analytics argues that development data can be used to guide engineering decisions, identify risk patterns, and support project governance (Menzies and Zimmermann,2013). AI patch validation extends this agenda by treating every generated patch as a data point with attributes: vulnerability class, prompt setting, model source, validation outcomes, review effort, release decision, and post-release events. When these data are aggregated, managers can identify which vulnerability classes, model configurations, or workflow stages are associated with excessive oracle divergence. Digital business strategy research supports this view because analytics value depends on how data capabilities reshape cross-functional decisions (Bharadwaj et al.,2013). Learning-based repair methods further illustrate why a generated patch should not be equated with a verified business outcome (Long and Rinard,2016).

The gap addressed by this article lies at the intersection of these streams. Automated repair

research studies whether patches are technically acceptable. Cybersecurity economics studies how organizations invest under risk. DevSecOps research studies how security is integrated into pipelines. Business analytics studies how operational data becomes managerial evidence. What remains underdeveloped is a framework that uses patch validation data to quantify business risk from AI-generated security fixes. BODA is proposed to fill this gap. DeFi research is relevant because it examines how automated digital infrastructures create new assurance, transparency, and governance requirements (Xu et al.,2024). Information-security compliance research shows that awareness and perceived benefit affect whether technical controls are followed in practice (Bulgurcu et al.,2010).

3. Business Oracle Divergence Analytics Framework

Business Oracle Divergence Analytics starts from a simple premise: a validation oracle is not only a technical test; it is also a managerial signal. When a release board accepts a patch because the original exploit no longer works, it implicitly treats that oracle as sufficient evidence. When a product team closes a security ticket because a static warning disappears, it treats the analyzer as a proxy for risk reduction. BODA makes these implicit assumptions explicit and measurable. Digital transformation literature supports this layered view because organizations create value by redesigning processes around data-enabled capabilities rather than by adding tools alone (Vial,2019). Patch plausibility studies show that passing a limited validation suite can leave substantial residual correctness risk (Qi et al.,2015).

The framework contains three layers. The first is the enterprise workflow layer, where vulnerabilities are triaged, AI-generated patches are proposed, code is reviewed, and releases are approved. The second is the patch evidence layer, where validation oracles produce structured outcomes. The third is the business risk analytics layer, where oracle divergence is translated into exposure, risk leakage, and release guidance.

4. Data Design and Measurement

The empirical component of this article is a controlled enterprise-style analytical experiment. It is not intended to estimate the universal success rate of AI-generated vulnerability repair. Instead, it illustrates how an organization could use patch-level validation data to measure oracle divergence and business risk. The dataset contains 480 AI-generated candidate patches, representing 48 vulnerability repair cases, 10 generated variants per case, and four vulnerability families commonly found in enterprise software: SQL injection, path traversal, cross-site scripting, and missing authorization. Software analytics research provides methodological support for turning development artifacts into decision evidence (Menzies and Zimmermann,2013). Patch-correctness research further shows that repair validation should distinguish test satisfaction from semantic correctness (Xiong et al.,2017).

Each case is framed as a workflow item inside an enterprise software portfolio. Some cases represent customer-facing APIs; others represent internal dashboards, identity and access-control services, data export utilities, and business reporting applications. This distinction matters because the same technical vulnerability can have different business consequences depending on asset criticality, data sensitivity, user population, and release timing. A missing authorization bug in an internal dashboard may be serious, but the same pattern in a customer account service may require emergency handling. Management analytics scholarship is useful here because it frames decision support as the connection between data, models, and managerial action (Lu et al.,2024). Security behavior research shows that organizational responses depend not only on rules but also

on how actors interpret consequences and habits (Johnston and Warkentin,2010).

Patch validation is measured across six layers. Build validity records whether the patch can be applied and executed. Functional preservation records whether established benign behavior remains intact. Original proof-of-concept blocking records whether the demonstrated exploit evidence no longer succeeds. Exploit-variant resistance records whether semantically related attacks are blocked. Root-cause conformance records whether the patch satisfies the expected security mechanism for the vulnerability class. Regression safety records whether the patch creates new security-relevant side effects or business-function failures. Empirical software engineering has shown that change-level risk prediction can help organizations prioritize review attention before defects reach production (Kim et al.,2008). Recent LLM vulnerability-repair studies confirm that reasoning and validation feedback influence repair quality but do not remove the need for independent evidence checks (Kulsum et al.,2024).

Table 1 translates these technical validation layers into business analytics constructs. The purpose is to show that validation evidence is not a purely engineering artifact. It has direct implications for risk interpretation, release confidence, and managerial action.

Table 1. Validation evidence and business analytics interpretation.

Validation layer	Technical question	Business interpretation	Typical management action
Build validity	Does the patched code execute?	Minimum feasibility, not assurance	Reject invalid patch or return to model
Functional preservation	Does normal behavior still work?	Operational continuity evidence	Run regression tests and owner review
Original PoC blocking	Is the demonstrated exploit blocked?	Initial exposure reduction	Escalate to variant testing
Exploit-variant resistance	Are equivalent attacks blocked?	Generalization evidence	Approve only if asset risk is moderate or lower
Root-cause conformance	Was the control repaired at the correct mechanism?	Release-grade security evidence	Require for high-criticality systems
Regression safety	Did the patch create new weaknesses?	Residual side-effect control	Approve, monitor, or delay release

The table also clarifies why a single validation outcome is insufficient for release governance. Each layer answers a different question, and the business decision depends on which uncertainty remains after the layer is passed.

5. Results: Oracle Divergence and Business Risk Leakage

The analysis begins with the validation ladder. As expected, acceptance rates decline as the evidence requirements become stronger. Build validity is high because most AI-generated patches are syntactically plausible. Functional preservation is lower because some patches prevent exploitation by over-restricting behavior or altering business logic. Original proof-of-concept blocking is lower still, because many plausible patches do not stop the demonstrated exploit. The largest business-relevant drop occurs when original proof-of-concept blocking is compared with release acceptance after variant, root-cause, and regression checks. Decision-making research in management analytics supports the use of tiered indicators when operational choices involve multiple uncertain signals (Lu et al.,2024). Information-security studies also show that habitual compliance patterns can mask deeper control weaknesses when evidence is not independently reviewed (Vance et al.,2012).

Table 2 reports the patch-level data schema and the reason each field is needed for business risk analytics. The schema is intentionally practical. It can be implemented in a CI/CD environment

using existing artifacts such as ticket metadata, code review records, SAST outputs, test reports, and release decisions. The key is that validation outcomes must be stored at patch level rather than collapsed into a single pass/fail label.

Table 2. Patch-level data schema for business risk analytics.

Field	Example value	Why it matters for analytics
Patch ID	API-PT-017-M4	Links generation, validation, review, and release records
Asset criticality	High	Weights technical failure by business consequence
Vulnerability class	CWE-22	Supports class-specific divergence baselines
Generation source	LLM repair prompt V3	Supports model and prompt governance
Weak-oracle outcome	Original PoC blocked	Records apparent repair evidence
Strong-oracle outcome	Variant failed	Identifies hidden residual risk
Release decision	Security review required	Connects analytics to workflow action
Post-release monitoring tag	Yes	Captures evidence debt after emergency release

The second result concerns vulnerability-class heterogeneity. Oracle divergence is not evenly distributed across vulnerability families. Path traversal shows high original-exploit blocking but lower variant resistance because many patches block literal traversal strings without robust normalization and base-directory enforcement. Missing authorization shows low release acceptance because correct repair requires trust-boundary placement and preservation of legitimate access paths. SQL injection shows moderate divergence because some patches filter payloads rather than adopting parameterized query construction. Cross-site scripting shows lower divergence in this dataset because context-aware encoding is easier to identify in the constructed cases, though this should not be generalized to all XSS settings.

Table 3 summarizes the acceptance ladder by vulnerability class. The table demonstrates why enterprises should not use a single global AI patch success rate. A release policy that is adequate for one vulnerability family may be too permissive for another. Business risk analytics therefore needs class-specific baselines and escalation thresholds.

Table 3. Validation outcomes by vulnerability class.

Vulnerability class	Patches	Original PoC accepted	Variant resistant	Root-cause conformant	Release accepted
SQL injection	120	82.0%	70.0%	58.0%	51.0%
Path traversal	120	91.0%	56.0%	49.0%	43.0%
Cross-site scripting	120	88.0%	79.0%	72.0%	68.0%
Missing authorization	120	54.0%	42.0%	33.0%	29.0%
Total	480	76.7%	64.8%	56.0%	47.7%

The third result concerns business impact. Technical divergence becomes managerial risk when it changes exposure, remediation cost, or operational continuity. Weak-oracle acceptance creates apparent risk reduction because the ticket appears remediated and the exploit evidence is blocked. However, residual variant risk, residual root-cause risk, regression risk, and operational delay cost partially offset this apparent benefit.

A portfolio view provides additional insight. If 100 AI-generated security patches are accepted after original exploit blocking alone, roughly 29 may fail release-grade validation under the constructed assumptions. Not all failures have equal business impact. A variant failure in a low-

criticality internal tool may require scheduled remediation. A root-cause failure in a customer identity service may require immediate rollback and incident review. BODA therefore uses a two-dimensional policy: validation divergence estimates the probability of hidden failure, while asset criticality estimates consequence.

Table 4 reports the risk zones used in the analysis. These zones can be adapted to organizational risk appetite. The purpose is not to create a universal score but to provide an auditable decision structure. A patch in the green zone may proceed after standard review. A patch in the gray zone requires additional security review or test generation. A patch in the black zone should not be released without root-cause evidence and regression review.

Table 4. Release risk zones based on oracle divergence and asset criticality.

Zone	Evidence condition	Business risk meaning	Recommended action
Green	Variant resistant and root-cause conformant	Residual risk is within standard release tolerance	Approve with routine monitoring
Light gray	Original PoC blocked but root cause not verified	Risk may be hidden behind weak evidence	Security engineer review required
Dark gray	Variant test failed on medium asset	Exploit-specific repair likely	Generate stronger patch or compensating control
Black	Weak-oracle success on high-criticality asset only	Release would create unacceptable hidden exposure	Reject or require executive exception
Post-release debt	Emergency release without full evidence	Risk accepted temporarily but not resolved	Schedule retrospective validation

6. Managerial Implications for Enterprise Software Workflows

The first implication is that AI-generated security patches should be governed as risk-bearing assets, not merely as code suggestions. A candidate patch changes the organization's exposure profile. If accepted prematurely, it can close a ticket without closing the risk. If rejected too conservatively, it can delay remediation and keep the vulnerability open. The managerial problem is therefore not whether to use AI-generated patches, but how to place them within an evidence-based acceptance workflow. Industrial information integration research shows that new computational paradigms create value when their outputs are incorporated into operational decision architectures (Lu et al.,2023). Research on policy neutralization suggests that controls fail when teams rationalize shortcuts under time pressure, which is a plausible risk in rapid AI patching workflows (Siponen and Vance,2010).

The second implication is that validation budgets should be allocated where divergence is highest. Many enterprises apply the same review intensity to all security patches. The results suggest a more efficient policy. Path traversal and authorization patches deserve stronger variant and root-cause review because their failure modes are more likely to survive original exploit blocking. XSS patches may still need context review, but a well-designed encoding test suite can reduce uncertainty earlier. SQL injection patches should be checked specifically for parameterized query construction rather than payload-specific filtering. Technical debt mapping studies indicate that unmanaged debt requires explicit identification, measurement, and repayment strategies (Li et al.,2015). Line-level vulnerability prediction research reinforces the value of granular evidence when prioritizing security review (Fu and Tantithamthavorn,2022).

The third implication concerns release governance. Traditional change advisory boards often focus on business continuity, while security teams focus on vulnerability severity. AI patch workflows require a combined view: release approval should depend on the interaction between

severity, validation strength, asset criticality, and rollback feasibility. A high-severity vulnerability patched by a weak oracle on a high-criticality asset should not be treated as low risk merely because the ticket is marked fixed. Research on quantum science trends demonstrates how emerging technical areas benefit from structured mapping before they mature into stable practices (Ye and Lu,2022). End-user security behavior research also indicates that confidence, capability, and perceived responsibility shape whether controls are enacted consistently (Rhee et al.,2009).

The fourth implication is that organizations should track model and prompt behavior over time. Some model configurations may produce high build and functional-pass rates while generating superficial security fixes. Others may produce fewer plausible patches but higher root-cause conformance. Without patch-level analytics, these differences remain invisible. BODA supports model governance by reporting acceptance rates at every evidence layer, not merely final productivity metrics. Exploratory technical-debt studies show that debt becomes dangerous when it remains invisible to managers and is normalized as routine work (Tom et al.,2013). Large-scale code generation results show that high benchmark performance still depends on carefully specified evaluation contexts (Li et al.,2022).

The fifth implication concerns security debt. Weakly validated AI patches can accumulate into a hidden backlog. Unlike ordinary backlog items, these patches may be recorded as completed. Organizations should therefore maintain an evidence debt register: patches accepted under limited validation should be tagged for future variant generation, root-cause review, or post-release monitoring. This practice converts hidden uncertainty into a managed risk object. Blockchain information-systems research emphasizes that technical trust mechanisms must be connected to organizational processes to deliver governance value (Lu,2022). Developer studies on static analysis show that even useful tools may be underused when evidence is hard to interpret or poorly integrated into workflow (Johnson et al.,2013).

7. Governance Integration and Implementation Roadmap

Implementing BODA requires changes to data capture, pipeline design, and organizational responsibility. At the data layer, CI/CD systems should record the patch source, prompt version, validation outcomes, vulnerability class, asset criticality, and reviewer decision. At the analytics layer, dashboards should show acceptance by evidence level, oracle divergence by class, and risk leakage by asset group. At the governance layer, release policies should specify the minimum acceptable evidence for each vulnerability class and criticality tier. Change-risk prediction research supports this implementation logic because release decisions improve when risk indicators are attached to concrete software changes (Mockus and Weiss,2000). Program-synthesis research shows that model-generated code should be treated as a candidate requiring specification-aware validation rather than as an automatically accepted artifact (Austin et al.,2021).

A practical implementation can begin with three controls. First, every AI-generated security patch should receive an evidence label indicating the strongest validation layer it has passed. Second, high-criticality assets should require at least exploit-variant resistance and root-cause conformance before release. Third, patches accepted under emergency conditions should be scheduled for retrospective validation and security debt review. These controls are lightweight but prevent the most damaging failure: treating weak-oracle success as complete repair. Blockchain trend research is relevant because distributed verification mechanisms demonstrate how traceable evidence can reduce information asymmetry across technology ecosystems (Zheng and Lu,2022). Empirical work on program-analysis tools shows that developers need actionable

evidence, not simply more warnings (Christakis and Bird,2016).

Table 5 presents an implementation roadmap. The stages progress from basic evidence capture to portfolio-level governance. Organizations do not need to implement the full framework immediately. Even recording original exploit, variant, and root-cause outcomes separately would improve decision quality over a single pass/fail metric.

Table 5. Implementation roadmap for BODA in enterprise DevSecOps.

Stage	Capability	Data requirement	Governance outcome
1. Evidence capture	Store validation layer outcomes	CI logs, test reports, SAST results	Patch decisions become auditable
2. Divergence metrics	Measure weak-to-strong oracle failure	Patch-level validation history	Risk leakage becomes visible
3. Class baselines	Estimate divergence by CWE and asset type	Vulnerability taxonomy and asset registry	Review effort is targeted
4. Release policy	Define minimum evidence by criticality	Risk appetite and business owner rules	Approvals become consistent
5. Model governance	Compare model and prompt profiles	Generation metadata and validation outcomes	AI repair quality is monitored
6. Continuous learning	Predict likely strong-oracle failure	Historical records and post-release events	Validation resources are optimized

The framework also has audit implications. Regulators and customers increasingly expect organizations to show not only that vulnerabilities were remediated, but that remediation was reasonable and controlled. BODA creates an auditable record of evidence. It shows which tests were passed, where uncertainty remained, who approved release, and whether compensating controls were applied. This record is valuable for internal audit, cyber insurance, vendor assurance, and post-incident analysis. DevSecOps research supports this audit orientation because secure delivery requires integrating security checks into continuous engineering practices (Rahman and Williams,2016). Transformer architecture research explains why modern code models are powerful at pattern learning while still requiring external validation for task-specific correctness (Vaswani et al.,2017).

Finally, the framework encourages a healthier relationship between AI speed and human judgment. AI can accelerate candidate generation, but enterprises still need structured evidence to decide whether a candidate patch is release-ready. Human review should focus less on manually rereading every line and more on evaluating whether validation evidence matches the business risk of the asset. This shifts expert effort toward the cases where it has the highest marginal value. IoT security research using blockchain shows that distributed systems need layered assurance because one technical mechanism rarely covers all risk surfaces (Xu et al.,2021). Large-scale static-analysis experience further shows that assurance tools must be designed around developer workflow to avoid becoming ignored noise (Sadowski et al.,2018).

7.1 Risk Score Calibration for Release Boards

For release boards, the most useful output of oracle divergence analytics is not a raw technical percentage but a calibrated decision score. The score should combine three families of variables: validation strength, asset consequence, and workflow urgency. Validation strength captures the strongest evidence layer passed by the patch. Asset consequence captures business exposure such as customer data sensitivity, revenue dependency, legal reporting obligations, and the difficulty of rollback. Workflow urgency captures whether the vulnerability is actively exploited, whether a public exploit is available, and whether delaying the release creates a larger exposure window. This structure prevents a common governance error: treating every AI patch that blocks the

original exploit as equally safe. DevSecOps literature supports score-based release governance because security evidence must be integrated into delivery timing, operational urgency, and accountability routines (Myrbakken and Colomo-Palacios,2017). Retrieval-augmented generation research also suggests that grounding can improve model output but still depends on downstream verification (Lewis et al.,2020).

A practical score can be implemented without complex mathematics. The organization can assign ordinal levels to each evidence layer and multiply them by criticality weights approved by risk leadership. For example, original exploit blocking may be sufficient for a low-criticality internal tool when compensating monitoring is available, but it should be insufficient for a customer identity service. Likewise, a root-cause-conformant patch may still require regression review when the affected component controls authentication, billing, or data export. The value of the score is not precision for its own sake; it is consistency. It forces different teams to justify release decisions using the same evidence vocabulary. Artificial intelligence research reviews indicate that AI adoption requires matching methods to use cases, constraints, and evaluation assumptions (Zhang and Lu,2021). Security-vulnerability prediction studies similarly show that risk scoring should be calibrated to observable code and process signals rather than broad severity labels alone (Shin and Williams,2013).

7.2 Dashboard Design and Key Performance Indicators

The dashboard associated with BODA should avoid presenting AI patching as a simple productivity story. Metrics such as number of generated patches, average time to patch, and number of tickets closed are useful, but they are incomplete and potentially misleading. A better dashboard separates productivity metrics from assurance metrics. Productivity metrics include generation success, developer acceptance, and remediation lead time. Assurance metrics include original proof-of-concept blocking, variant resistance, root-cause conformance, regression-safety pass rate, and weak-to-strong oracle divergence. Only when these two groups are viewed together can managers understand whether the organization is becoming faster without becoming less safe. DevOps performance research shows that speed metrics can be misleading unless combined with reliability, quality, and recovery indicators (Forsgren et al.,2018). Prompting research indicates that model reasoning can be elicited, but the resulting output still requires external checks before organizational reliance (Wei et al.,2022).

The most important key performance indicator is the release-grade acceptance ratio: the proportion of AI-generated candidate patches that pass all evidence required for the asset criticality class. A second useful indicator is the evidence downgrade rate, which counts patches initially marked as fixed but later downgraded after stronger validation. A third indicator is security-debt carryover, which counts emergency releases that were accepted before complete validation and remain unresolved after a defined period. These indicators give senior management a way to distinguish genuine automation value from superficial closure of security tickets. Management analytics research supports dashboard designs that connect metrics to decisions rather than reporting isolated productivity counts (Lu,2021). Exploit prediction research shows that security remediation benefits when probability of real-world exploitation is incorporated into prioritization (Jacobs et al.,2020).

7.3 Integration with Existing Enterprise Controls

BODA does not need to replace existing security controls. It can be layered on top of vulnerability management systems, code review platforms, ticketing systems, and release-management boards. The key requirement is data integration. A ticket should not merely say that

a patch was generated and merged. It should record which validation layers were passed, which were skipped, who approved exceptions, and which compensating controls were applied. This information is especially important for regulated sectors, where organizations may need to demonstrate that remediation decisions were reasonable under known constraints. DevOps case-study research shows that successful adoption depends on cross-functional coordination across development, operations, and governance roles (Erich et al.,2017). Agent-style prompting research suggests that reasoning and action loops can improve task execution while expanding the need for traceable evidence records (Yao et al.,2022).

In practice, three integrations matter most. The first is integration with the asset inventory, because business consequence cannot be assessed without knowing which service, database, or user group is affected. The second is integration with CI/CD evidence, because validation outcomes must be captured automatically rather than reconstructed manually. The third is integration with the risk register, because residual risk from weakly validated patches should be visible to business owners. Without these integrations, oracle divergence remains a technical insight rather than a management control. Research on 6G governance and infrastructure highlights that interconnected digital services require security-aware integration across technical layers (Lu and Ning,2020). Exploit-prediction studies show that risk information becomes more actionable when it is modeled as likelihood and consequence rather than as an undifferentiated vulnerability list (Bozorgi et al.,2010).

7.4 Organizational Roles and Accountability

AI patch validation also changes accountability. Developers should remain responsible for code quality, but they should not be expected to carry the entire burden of security assurance when patches are generated by external models and validated by automated tools. Security engineers should define root-cause criteria and variant-generation policies. Product owners should define business criticality and acceptable downtime. Release managers should enforce evidence thresholds. Internal audit should verify whether exceptions are properly documented. This role separation is important because weak validation often becomes dangerous when responsibility is diffuse. Software supply-chain studies show that dependency ecosystems expose organizations to risks beyond the immediate code owner, which makes accountability assignment essential (Ohm et al.,2020). Cloud pricing research also illustrates that technical service quality and economic incentives should be modeled together when digital infrastructure choices affect business outcomes (Lu et al.,2020).

The framework therefore recommends an evidence owner for every high-risk AI patch. The evidence owner does not need to write the patch. Instead, this person confirms that the validation record is complete enough for the release decision. For low-risk patches, this task can be automated. For high-risk patches, it should involve a named reviewer. Naming the evidence owner creates accountability without slowing every release equally. It also helps organizations learn over time which teams, services, and vulnerability classes generate repeated evidence gaps. Recent software supply-chain taxonomies demonstrate that attack paths are diverse and require coordinated controls across artifacts, maintainers, and organizations (Gokkaya et al.,2026). Algorithmic auditing research similarly argues that accountability should be structured across the entire lifecycle rather than delegated to a single technical actor (Raji et al.,2020).

7.5 Strategic Value of Oracle Divergence Analytics

From a strategic perspective, oracle divergence analytics gives executives a more realistic view of AI value. A model that produces many patches but also produces high divergence may increase

short-term throughput while adding hidden security debt. A model that produces fewer patches but higher root-cause conformance may be more valuable for critical systems. This distinction matters for procurement, vendor evaluation, and internal AI platform investment. Organizations should therefore compare AI repair systems not only by code-generation quality or developer satisfaction, but also by release-grade validation performance. Supply-chain integrity research demonstrates that provenance and verification records can materially change downstream trust decisions (Torres-Arias et al.,2019). Explainable AI research supports the need to present evidence in forms that decision makers can inspect and challenge (Ribeiro et al.,2016).

The same logic applies to prompt engineering and workflow design. A prompt that asks the model to fix the shown exploit may produce high original proof-of-concept blocking but poor variant resistance. A prompt that asks for root-cause repair and functional preservation may produce fewer immediate passes but better business outcomes. BODA enables enterprises to quantify this trade-off. Instead of relying on anecdotal developer impressions, managers can compare prompt families using evidence-layer metrics and select the workflow that minimizes residual release risk. Model-interpretability research further indicates that local explanations and evidence attribution can strengthen review processes when automated systems influence decisions (Lundberg and Lee,2017).

7.6 Practical Example of a Release Decision

Consider an AI-generated patch for a path traversal flaw in a file download endpoint used by business customers. The patch blocks the original traversal string and passes existing functional tests. Under a weak-oracle policy, the ticket might be closed. Under BODA, the patch remains in a gray or black zone until variant testing and root-cause review are complete. If encoded traversal strings bypass the patch, the apparent remediation is downgraded to exploit-specific blocking. The business decision changes immediately: the release should be delayed, a stronger patch should be generated, or a compensating control should be activated at the gateway. This example shows how the framework turns technical disagreement among validation layers into a clear workflow action. This case illustrates why path-level fixes need evidence beyond one observed exploit because release decisions may otherwise understate business exposure (Jacobs et al.,2020).

A second example is a missing authorization repair in an internal reporting dashboard. The AI-generated patch adds an authentication check to one route but does not enforce object-level authorization. The original proof-of-concept may fail because the demonstrated request no longer reaches the vulnerable route, yet a user with a valid login may still access another user's report through a different parameter. Root-cause conformance prevents this error by asking whether the trust boundary and object permission were repaired, not merely whether the original request was blocked. For business managers, this distinction is crucial because authorization failures often create compliance and confidentiality exposure even when service availability is unaffected. The authorization example also shows why organizational accountability must distinguish authentication success from object-level permission enforcement (NIST,2022).

8. Limitations and Future Research

This article has several limitations. The analytical dataset is controlled and enterprise-style rather than derived from a production software portfolio. The numerical values are designed to illustrate risk analytics rather than to estimate universal repair rates. Real organizations will observe different divergence levels depending on programming language, test maturity, asset architecture, model configuration, prompt design, and review culture. The generalizable contribution is

therefore methodological: patch validation should be measured as layered evidence and translated into business risk. These limitations are consistent with the broader analytics literature, which treats constructed datasets as useful for method demonstration but not as substitutes for external validation (Chen et al.,2012).

The business impact estimates are also simplified. Real breach cost depends on detection speed, data sensitivity, customer notification obligations, contractual penalties, regulatory environment, and brand trust. Future research should connect patch-level divergence data with incident loss models, cyber insurance pricing, and empirical post-release vulnerability outcomes. This would allow organizations to estimate not only validation failure rates but expected monetary loss. Future incident-cost modeling should build on security-economics research that treats breach exposure as a function of probability, loss magnitude, and control effectiveness (Gordon and Loeb,2002).

Future work should also study dynamic learning. As organizations collect patch validation records, they can train risk models that predict which AI-generated patches are likely to fail variant or root-cause checks. These models could prioritize human review and automatically generate additional tests for high-risk cases. Another promising direction is governance benchmarking across firms: organizations could compare oracle divergence patterns without sharing sensitive source code by exchanging sanitized validation metrics. Future learning systems should also draw on software analytics research by using accumulated validation records to improve review prioritization over time (Menzies and Zimmermann,2013).

9. Conclusion

AI-generated security patches create a new opportunity and a new risk for enterprise software organizations. They can shorten remediation cycles and reduce engineering workload, but they can also create premature confidence when weak validation signals are mistaken for complete security repair. This article developed Business Oracle Divergence Analytics, a framework for measuring the gap between apparent technical success and release-grade assurance in enterprise workflows.

The analysis showed that validation evidence becomes progressively stronger as organizations move from build validity and functional preservation toward exploit-variant resistance, root-cause conformance, and regression-safety review. In the constructed enterprise dataset, original proof-of-concept blocking substantially overstated release readiness. This gap is not simply a technical error; it is business risk leakage. It affects exposure, remediation cost, auditability, release confidence, and security debt.

The managerial message is clear. Enterprises adopting LLM-based vulnerability repair should not ask only whether a generated patch passes a test or blocks one exploit. They should ask which oracle accepted the patch, which stronger oracles remain untested, what asset would be exposed if the acceptance is wrong, and what governance action is appropriate. By converting oracle divergence into business risk analytics, organizations can use AI-generated patches responsibly while preserving the assurance discipline required for secure digital operations.

Acknowledgement

The authors thank the anonymous reviewers and editorial team for their constructive suggestions. The article was prepared as a conceptual and analytical study using constructed enterprise-style validation records for methodological demonstration.

Author Contributions

Table 6. Author contributions.

Author	Contribution
Isabel Navarro	Conceptualization, writing - original draft, business risk model design
Miguel Ángel Rojas	Data curation, formal analysis, validation, visualization
Clara Benítez	Supervision, methodology, writing - review & editing, project administration

Declarations

Conflicts of interest: The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this manuscript.

Data availability: The constructed aggregate data used for the illustrative analysis are available from the corresponding author upon reasonable request. No executable exploit artifacts, attack payloads, or sensitive operational records are included in this manuscript.

Funding: This research received no external funding.

Ethics statement: The manuscript does not involve human participants, animal experiments, identifiable personal records, or live-system security testing.

About the Authors

Isabel Navarro is affiliated with the University of Murcia, Spain. Her research focuses on information systems risk, enterprise analytics, and software governance.

Miguel Ángel Rojas is affiliated with the University of Vigo, Spain. His work examines business analytics, technology management, and organizational decision models.

Clara Benítez is affiliated with the University of Castilla-La Mancha, Spain. Her research addresses cybersecurity management, AI governance, and software assurance analytics.

References

Yang, L., Hou, Q., Zhu, X., Lu, Y., & Xu, L. D. (2025). Potential of large language models in blockchain-based supply chain finance. Enterprise Information Systems, 19(11), 2541199. https://doi.org/10.1080/17517575.2025.2541199

Gordon, L. A., & Loeb, M. P. (2002). The economics of information security investment. ACM Transactions on Information and System Security, 5(4), 438-457. https://doi.org/10.1145/581271.581274

Chen, H., Chiang, R. H. L., & Storey, V. C. (2012). Business intelligence and analytics: From big data to big impact. MIS Quarterly, 36(4), 1165-1188. https://doi.org/10.2307/41703503

Xia, C. S., Wei, Y., & Zhang, L. (2023). Automated program repair in the era of large pre-trained language models. Proceedings of the 45th IEEE/ACM International Conference on Software Engineering,1482-1494. https://doi.org/10.1109/ICSE48619.2023.00129

Zhang, H., & Lu, Y. (2025). Web 3.0: Applications, opportunities and challenges in the next internet generation. Systems Research and Behavioral Science, 42(4), 996-1015. https://doi.org/10.1002/sres.3151

Anderson, R., & Moore, T. (2006). The economics of information security. Science, 314(5799), 610-613. https://doi.org/10.1126/science.1130992

Choi, T. M., Wallace, S. W., & Wang, Y. (2018). Big data analytics in operations management. Production

and Operations Management, 27(10), 1868-1883. https://doi.org/10.1111/poms.12838

Chen, M., Tworek, J., Jun, H., Yuan, Q., Pinto, H. P. O., Kaplan, J., et al. (2021). Evaluating large language models trained on code. arXiv. https://doi.org/10.48550/arXiv.2107.03374

Kou, G., & Lu, Y. (2025). FinTech: A literature review of emerging financial technologies and applications. Financial Innovation, 11(1), 1-34. https://doi.org/10.1186/s40854-024-00668-6

Arora, A., Forman, C., Nandkumar, A., & Telang, R. (2010). Competition and patching of security vulnerabilities: An empirical analysis. Information Economics and Policy, 22(2), 164-177. https://doi.org/10.1016/j.infoecopol.2009.10.002

Mikalef, P., Krogstie, J., Pappas, I. O., & Pavlou, P. (2020). Exploring the relationship between big data analytics capability and competitive performance. Technological Forecasting and Social Change, 154, 119767. https://doi.org/10.1016/j.techfore.2019.119767

Roziere, B., Gehring, J., Gloeckle, F., Sootla, S., Gat, I., Tan, X. E., et al. (2023). Code Llama: Open foundation models for code. arXiv. https://doi.org/10.48550/arXiv.2308.12950

Lu, Y. (2025). The current status and developing trends of Industry 4.0: A review. Information Systems Frontiers, 27(1), 215-234. https://doi.org/10.1007/s10796-021-10221-w

Böhme, R. (2010). Security metrics and security investment models. In Advances in Information and Computer Security (pp. 10-24). Springer. https://doi.org/10.1007/978-3-642-16825-3_2

Wamba, S. F., Gunasekaran, A., Akter, S., Ren, S. J., Dubey, R., & Childe, S. J. (2017). Big data analytics and firm performance: Effects of dynamic capabilities. Journal of Business Research, 70, 356-365. https://doi.org/10.1016/j.jbusres.2016.08.009

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., et al. (2023). Llama 2: Open foundation and fine-tuned chat models. arXiv. https://doi.org/10.48550/arXiv.2307.09288

Wu, H. P., Liu, Z., Dong, H. Y., Lu, Y., & Xu, L. D. (2025). Revolutionizing internal auditing: Harnessing the power of blockchain. Enterprise Information Systems, 19(1-2). https://doi.org/10.1080/17517575.2024.2448003

National Institute of Standards and Technology. (2022). Cybersecurity supply chain risk management practices for systems and organizations (NIST SP 800-161 Rev. 1). https://doi.org/10.6028/NIST.SP.800-161r1

Kitchens, B., Dobolyi, D., Li, J., & Abbasi, A. (2018). Advanced customer analytics: Strategic value through integration of relationship-oriented big data. Journal of Management Information Systems, 35(2), 540-574. https://doi.org/10.1080/07421222.2018.1451957

Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., et al. (2023). GPT-4 technical report. arXiv. https://doi.org/10.48550/arXiv.2303.08774

Lu, W., Lu, Y., Li, J., Sigov, A., Ratkin, L., & Ivanov, L. A. (2024). Quantum machine learning: Classifications, challenges, and solutions. Journal of Industrial Information Integration, 42, 100736. https://doi.org/10.1016/j.jii.2024.100736

DeLone, W. H., & McLean, E. R. (2003). The DeLone and McLean model of information systems success: A ten-year update. Journal of Management Information Systems, 19(4), 9-30. https://doi.org/10.1080/07421222.2003.11045748

Teece, D. J. (2007). Explicating dynamic capabilities: The nature and microfoundations of sustainable enterprise performance. Strategic Management Journal, 28(13), 1319-1350. https://doi.org/10.1002/smj.640

Le Goues, C., Nguyen, T., Forrest, S., & Weimer, W. (2012). GenProg: A generic method for automatic software repair. IEEE Transactions on Software Engineering, 38(1), 54-72. https://doi.org/10.1109/TSE.2011.104

Lu, Y., & Yang, J. (2024). Quantum financing system: A survey on quantum algorithms, potential scenarios and open research issues. Journal of Industrial Information Integration, 41, 100663. https://doi.org/10.1016/j.jii.2024.100663

Herath, T., & Rao, H. R. (2009). Encouraging information security behaviors in organizations: Role of penalties, pressures and perceived effectiveness. Decision Support Systems, 47(2), 154-165.

https://doi.org/10.1016/j.dss.2009.02.005

Bharadwaj, A., El Sawy, O. A., Pavlou, P. A., & Venkatraman, N. (2013). Digital business strategy: Toward a next generation of insights. MIS Quarterly, 37(2), 471-482. https://doi.org/10.25300/MISQ/2013/37.2.03

Long, F., & Rinard, M. (2016). Automatic patch generation by learning correct code. Proceedings of the 43rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, 298-312. https://doi.org/10.1145/2837614.2837617

Xu, R., Zhu, J., Yang, L., Lu, Y., & Xu, L. D. (2024). Decentralized finance (DeFi): A paradigm shift in the FinTech. Enterprise Information Systems, 18(9). https://doi.org/10.1080/17517575.2024.2397630

Bulgurcu, B., Cavusoglu, H., & Benbasat, I. (2010). Information security policy compliance: An empirical study of rationality-based beliefs and information security awareness. MIS Quarterly, 34(3), 523-548. https://doi.org/10.25300/MISQ/2010/34.3.04

Vial, G. (2019). Understanding digital transformation: A review and a research agenda. Journal of Strategic Information Systems, 28(2), 118-144. https://doi.org/10.1016/j.jsis.2019.01.003

Qi, Z., Long, F., Achour, S., & Rinard, M. (2015). An analysis of patch plausibility and correctness for generate-and-validate patch generation systems. Proceedings of the 2015 International Symposium on Software Testing and Analysis, 24-36. https://doi.org/10.1145/2771783.2771791

Chen, Y., Lu, Y., Bulysheva, L., & Kataev, M. Y. (2024). Applications of blockchain in Industry 4.0: A review. Information Systems Frontiers, 26(5), 1715-1729. https://doi.org/10.1007/s10796-022-10248-7

Safa, N. S., Von Solms, R., & Furnell, S. (2016). Information security policy compliance model in organizations. Computers & Security, 56, 70-82. https://doi.org/10.1016/j.cose.2015.10.006

Menzies, T., & Zimmermann, T. (2013). Software analytics: So what? IEEE Software, 30(4), 31-37. https://doi.org/10.1109/MS.2013.86

Xiong, Y., Liu, X., Zeng, M., Zhang, L., & Huang, G. (2017). Identifying patch correctness in test-based program repair. Proceedings of the 39th International Conference on Software Engineering, 789-799. https://doi.org/10.1109/ICSE.2017.57

Lu, Y., Ivanov, L. A., Wang, F., Pisarenko, Z. V., & Ye, C. (2024). Management analytics: A bibliometric analysis. Nanotechnologies in Construction, 16(3), 257-266. https://doi.org/10.15828/2075-8545-2024-16-3-257-266

Johnston, A. C., & Warkentin, M. (2010). Fear appeals and information security behaviors: An empirical study. MIS Quarterly, 34(3), 549-566. https://doi.org/10.25300/MISQ/2010/34.3.05

Kim, S., Whitehead, E. J., & Zhang, Y. (2008). Classifying software changes: Clean or buggy? IEEE Transactions on Software Engineering, 34(2), 181-196. https://doi.org/10.1109/TSE.2007.70773

Kulsum, U., Zhu, H., & Brown, A. (2024). A case study of LLM for automated vulnerability repair: Assessing impact of reasoning and patch validation feedback. Proceedings of the 1st ACM International Conference on AI-Powered Software, 63-74. https://doi.org/10.1145/3664646.3664770

Lu, Y., Pisarenko, Z. V., Yang, L., & Ye, C. (2024). Advancing decision-making: The role of management analytics in modern business practices. Nanotechnologies in Construction, 16(5), 431-440. https://doi.org/10.15828/2075-8545-2024-16-5-431-440

Vance, A., Siponen, M., & Pahnila, S. (2012). Motivating IS security compliance: Insights from habit and protection motivation theory. Information & Management, 49(3-4), 190-198. https://doi.org/10.1016/j.im.2012.04.002

Kruchten, P., Nord, R. L., & Ozkaya, I. (2012). Technical debt: From metaphor to theory and practice. IEEE Software, 29(6), 18-21. https://doi.org/10.1109/MS.2012.167

de-Fitero-Dominguez, D., Garcia-Lopez, E., Garcia-Cabot, A., & Martinez-Herraiz, J. J. (2024). Enhanced automated code vulnerability repair using large language models. Engineering Applications of Artificial Intelligence, 138, 109291. https://doi.org/10.1016/j.engappai.2024.109291

Lu, Y., Sigov, A. S., Ratkin, L., Ivanov, L. A., & Zuo, M. (2023). Quantum computing and industrial information integration: A review. Journal of Industrial Information Integration, 35, 100511. https://doi.org/10.1016/j.jii.2023.100511

Siponen, M., & Vance, A. (2010). Neutralization: New insights into the problem of employee information systems security policy violations. MIS Quarterly, 34(3), 487-502. https://doi.org/10.25300/MISQ/2010/34.3.02

Li, Z., Avgeriou, P., & Liang, P. (2015). A systematic mapping study on technical debt and its management. Journal of Systems and Software, 101, 193-220. https://doi.org/10.1016/j.jss.2014.12.027

Fu, M., & Tantithamthavorn, C. (2022). LineVul: A transformer-based line-level vulnerability prediction. Proceedings of the 19th International Conference on Mining Software Repositories, 608-620. https://doi.org/10.1145/3524842.3528455

Ye, Z., & Lu, Y. (2022). Quantum science: A review and current research trends. Journal of Management Analytics, 9(3), 383-402. https://doi.org/10.1080/23270012.2022.2089064

Rhee, H. S., Kim, C., & Ryu, Y. U. (2009). Self-efficacy in information security: Its influence on end users' information security practice behavior. Computers & Security, 28(8), 816-826. https://doi.org/10.1016/j.cose.2009.05.008

Tom, E., Aurum, A., & Vidgen, R. (2013). An exploration of technical debt. Journal of Systems and Software, 86(6), 1498-1516. https://doi.org/10.1016/j.jss.2012.12.052

Li, Y., Choi, D., Chung, J., Kushman, N., Schrittwieser, J., Leblond, R., et al. (2022). Competition-level code generation with AlphaCode. Science, 378(6624), 1092-1097. https://doi.org/10.1126/science.abq1158

Lu, Y. (2022). Implementing blockchain in information systems: A review. Enterprise Information Systems, 16(12), 1876-1907. https://doi.org/10.1080/17517575.2021.2008513

Johnson, B., Song, Y., Murphy-Hill, E., & Bowdidge, R. (2013). Why don't software developers use static analysis tools to find bugs? Proceedings of the 35th International Conference on Software Engineering, 672-681. https://doi.org/10.1109/ICSE.2013.6606583

Mockus, A., & Weiss, D. M. (2000). Predicting risk of software changes. Bell Labs Technical Journal, 5(2), 169-180. https://doi.org/10.1002/bltj.2229

Austin, J., Odena, A., Nye, M., Bosma, M., Michalewski, H., Dohan, D., et al. (2021). Program synthesis with large language models. arXiv. https://doi.org/10.48550/arXiv.2108.07732

Zheng, X. R., & Lu, Y. (2022). Blockchain technology: Recent research and future trend. Enterprise Information Systems, 16(12), 1939895. https://doi.org/10.1080/17517575.2021.1939895

Christakis, M., & Bird, C. (2016). What developers want and need from program analysis: An empirical study. Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering, 332-343. https://doi.org/10.1145/2970276.2970347

Rahman, A. A. U., & Williams, L. (2016). Software security in DevOps: Synthesizing practitioners' perceptions and practices. Proceedings of the International Workshop on Continuous Software Evolution and Delivery, 70-76. https://doi.org/10.1145/2896941.2896946

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention is all you need. arXiv. https://doi.org/10.48550/arXiv.1706.03762

Xu, L. D., Lu, Y., & Li, L. (2021). Embedding blockchain technology into IoT for security: A survey. IEEE Internet of Things Journal, 8(13), 10452-10473. https://doi.org/10.1109/JIOT.2021.3060508

Sadowski, C., Aftandilian, E., Eagle, A., Miller-Cushon, L., & Jaspan, C. (2018). Lessons from building static analysis tools at Google. Communications of the ACM, 61(4), 58-66. https://doi.org/10.1145/3188720

Myrbakken, H., & Colomo-Palacios, R. (2017). DevSecOps: A multivocal literature review. In Software Process Improvement and Capability Determination (pp. 17-29). Springer. https://doi.org/10.1007/978-3-319-67383-7_2

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., et al. (2020). Retrieval-augmented generation for knowledge-intensive NLP tasks. arXiv. https://doi.org/10.48550/arXiv.2005.11401

Zhang, C., & Lu, Y. (2021). Study on artificial intelligence: The state of the art and future prospects. Journal of Industrial Information Integration, 23, 100224. https://doi.org/10.1016/j.jii.2021.100224

Shin, Y., & Williams, L. (2013). An empirical model to predict security vulnerabilities using code complexity metrics. IEEE Transactions on Dependable and Secure Computing, 10(5), 291-304. https://doi.org/10.1109/TDSC.2012.15

Forsgren, N., Humble, J., & Kim, G. (2018). Accelerate: The science of lean software and DevOps. IT Revolution Press. https://doi.org/10.4324/9780429463839

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., et al. (2022). Chain-of-thought prompting elicits reasoning in large language models. arXiv. https://doi.org/10.48550/arXiv.2201.11903

Lu, Y. (2021). Technological innovation and the emergence of a new interdisciplinary field: Management Analytics. Nanotechnologies in Construction, 13(3), 181-192. https://doi.org/10.15828/2075-8545-2021-13-3-181-192

Jacobs, J., Romanosky, S., Edwards, B., Adjerid, I., & Roytman, M. (2020). Improving vulnerability remediation through better exploit prediction. Journal of Cybersecurity, 6(1), tyaa015. https://doi.org/10.1093/cybsec/tyaa015

Erich, F. M. A., Amrit, C., & Daneva, M. (2017). A qualitative study of DevOps usage in practice. Journal of Software: Evolution and Process, 29(6), e1885. https://doi.org/10.1002/smr.1885

Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., & Cao, Y. (2022). ReAct: Synergizing reasoning and acting in language models. arXiv. https://doi.org/10.48550/arXiv.2210.03629

Lu, Y., & Ning, X. (2020). A vision of 6G-5G's successor. Journal of Management Analytics, 7(3), 301-320. https://doi.org/10.1080/23270012.2020.1802622

Bozorgi, M., Saul, L. K., Savage, S., & Voelker, G. M. (2010). Beyond heuristics: Learning to classify vulnerabilities and predict exploits. Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 105-114. https://doi.org/10.1145/1835804.1835818

Ohm, M., Plate, H., Sykosch, A., & Meier, M. (2020). Backstabber's knife collection: A review of open source software supply chain attacks. In Detection of Intrusions and Malware, and Vulnerability Assessment (pp. 23-43). Springer. https://doi.org/10.1007/978-3-030-52683-2_2

Lu, Y., Zheng, X., Li, L., & Xu, L. D. (2020). Pricing the cloud: A QoS-based auction approach. Enterprise Information Systems, 14(3), 334-351. https://doi.org/10.1080/17517575.2019.1669827

Gokkaya, B., Aniello, L., & Halak, B. (2026). Software supply chain: A taxonomy of attacks, mitigations and risk assessment strategies. Journal of Information Security and Applications, 97, 104324. https://doi.org/10.1016/j.jisa.2025.104324

Raji, I. D., Smart, A., White, R. N., Mitchell, M., Gebru, T., Hutchinson, B., et al. (2020). Closing the AI accountability gap: Defining an end-to-end framework for internal algorithmic auditing. Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency, 33-44. https://doi.org/10.1145/3351095.3372873

Torres-Arias, S., Afzali, H., Kuppusamy, T. K., Curtmola, R., & Cappos, J. (2019). in-toto: Providing farm-to-table guarantees for bits and bytes. Proceedings of the 28th USENIX Security Symposium,1393-1410. https://doi.org/10.1145/3238147.3238171

Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). Why should I trust you? Explaining the predictions of any classifier. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining,1135-1144. https://doi.org/10.1145/2939672.2939778

Lundberg, S. M., & Lee, S. I. (2017). A unified approach to interpreting model predictions. arXiv. https://doi.org/10.48550/arXiv.1705.07874