

Requirements-Related Software Failures

Harouna Wendpanga Yann Christian Sankara^{1*}, Aisha I. Abdulwahab², Toussida Fatah Tanguy Minoungou³, Olumide Kumuyi⁴

¹ Five9, San Ramon, California, USA

² Cummins Inc., Indiana, USA

³ Nationwide, Columbus, Ohio, USA

⁴ New York University Abu Dhabi, Abu Dhabi, United Arab Emirates

* Corresponding author: christiansankara@gmail.com

Abstract

Purpose: Software failures that reach operation are often traced, after the event, to decisions made while requirements were elicited, written, linked and validated, yet defect records usually describe the symptom rather than that origin. This article asks which classes of requirements defect produce escaped failures, why late detection multiplies their cost, and which lifecycle controls contain them. **Method:** A critical integrative review of empirical software engineering research, requirements and testing standards, and public investigation findings, synthesised through fault-chain and latent-condition theory and a phase-containment cost model. **Findings:** Five defect classes (elicitation failure, ambiguity, inconsistency, traceability loss and validation failure) differ in where they are injected, how far they travel before detection and which controls intercept them. Well-documented public failures, including Ariane 5 Flight 501, the Mars Climate Orbiter and the Therac-25 accidents, are dominated by unexamined operating assumptions and validation against the wrong reference rather than by wording defects. **Contribution:** The Requirements Defect Origin and Containment (RDOC) framework places five containment gates across the lifecycle, defines phase containment, requirements leakage and severity-weighted escape distance as metrics, and states six falsifiable propositions with tests. **Implications:** Prevention depends on revalidating reused assumptions, auditing trace links whenever requirements change, and feeding blame-free causation analysis of escaped defects back into gate controls.

Keywords: requirements defects, failure causation analysis, phase containment, requirements traceability, validation failure, software failure investigation

DOI: <https://doi.org/10.21590/ijtmh20241004422>

1 Introduction

Software systems fail in operation for many reasons, but a persistent share of the most consequential failures originate before a line of code is written, in the statements that define what the system must do, under which conditions and for whom. When that definition is incomplete, admits more than one reading, contradicts itself or another artifact, loses its links to the design and tests that implement it, or is validated

against a reference that does not represent the real operating world, the resulting fault can pass every verification activity, because verification checks conformance to the specification rather than fitness for use (IEEE Computer Society, 2014; ISO/IEC/IEEE, 2018). The testing literature separates a fault (a static defect in an artifact), an error (an incorrect internal state) and a failure (externally observable deviation from required behaviour), and it shows that a fault is revealed only if a test reaches it, corrupts state, propagates that corruption to an output and meets an oracle able to recognise it (Ammann et al., 2016; Myers et al., 2011). A requirements defect is a fault in the specification itself, and the oracles derived from that specification inherit it (Kaner et al., 1999).

Current defect management practice obscures this origin. Issue trackers typically classify defects by manifestation (crash, wrong output, degraded performance), by component and by severity, and they often record the phase in which a defect was found rather than the phase in which it was introduced. The consequence is a systematic bias toward downstream remedies: more test cases, better test prioritisation, and machine-learning assistance for test generation and selection (Li et al., 2007; Catal et al., 2013; Borah et al., 2022). These techniques improve the efficiency with which specified behaviour is exercised, but they cannot expose behaviour that the specification never described. Service dashboards that report closure rather than origin reinforce the bias (Edivri et al., 2024; Aliliele et al., 2023b). Quality management in regulated industries has long distinguished correction of a nonconformity from prevention of its recurrence, and has shown that preventive action is only as good as the causal analysis behind it (Aneke et al., 2022a; Yusuf et al., 2018; Asiedu et al., 2024b). Software engineering has been slower to make that distinction operational for requirements.

Three gaps motivate the present review. First, requirement quality attributes are well codified in standards, and traceability research has produced reference models and link recovery techniques, but these bodies of work are rarely connected to the causation analysis of failures that actually escaped (Gotel et al., 1994; Ramesh et al., 2001; Winkler et al., 2010). Second, the claim that requirements defects become more expensive the later they are found is repeated in almost every textbook, yet it is seldom expressed in a form that an organisation could estimate from its own data, and the size of the escalation varies considerably with development model and domain. Third, public failure cases are cited anecdotally, often compressed to a single cause, and used to support whichever defect class an author wishes to promote. Even the vocabulary of non-functional requirements lacks agreed definitions, which complicates classification (Mairiza et al., 2010).

The article therefore addresses four research questions. RQ1: Which classes of requirements defect can be distinguished by their injection mechanism and by the control that would intercept them? RQ2: Under what conditions does late detection multiply the cost of a requirements defect, and how can the multiplier be expressed so that it can be estimated rather than asserted? RQ3: What do well-documented public failures reveal about the relative contribution of each class when they are read against a consistent causation protocol? RQ4: Which lifecycle controls and feedback mechanisms contain requirements defects, and which claims about them can be tested empirically? The questions follow the distinction between knowledge questions and design problems that is standard in empirical software engineering, so that descriptive synthesis and proposed artifacts are kept apart (Easterbrook et al., 2008; Wieringa, 2014).

The article makes four contributions. (1) It develops a causation taxonomy of five requirements defect classes (elicitation failure, ambiguity, inconsistency, traceability loss and validation failure), each defined by its injection mechanism, carrier artifact, typical escape route and observable signal (Table 2). (2) It formalises the cost-of-late-detection argument as a phase-containment model with an escalation ratio and

gate containment probabilities, derives the marginal value of upstream containment, and identifies the conditions under which the argument weakens (Figure 1). (3) It re-reads eight public failure cases, including one deliberate boundary case with a non-requirements origin, through a single coding protocol and an escape-distance representation (Table 3, Figure 3). (4) It proposes the Requirements Defect Origin and Containment (RDOC) framework, which places five containment gates across the lifecycle, specifies three containment metrics and a causation feedback loop, and states six falsifiable propositions with tests (Figure 4, Table 4). The framework is offered as a design science artifact whose utility remains to be demonstrated in the field (Hevner et al., 2004).

The scope is enterprise and safety-related software in which requirements are documented to some degree, whether in a formal specification, a backlog of user stories or a set of interface control documents. Automated language analysis of requirements text, the management of requirements volatility and the design of stakeholder collaboration are treated only where they bear on failure causation, since each is a substantial topic in its own right. Throughout, claims drawn from the literature are separated from the propositions that this article advances and that still require empirical test (Kitchenham et al., 2009).

2 Theoretical Grounding: Fault Chains, Latent Conditions and Containment Economics

Three bodies of theory explain why requirements defects behave differently from implementation defects once a system is built. Dependability and testing theory describes the chain from fault to failure and the conditions under which a test reveals a fault. Organisational accident theory from safety science describes how decisions taken far from the point of operation lie dormant until an operating condition activates them. Engineering economics, applied to defect removal, describes how the cost of repair grows with the distance between the phase in which a defect is introduced and the phase in which it is found. The first explains why verification misses requirements defects, the second why they surface as surprising operational failures, and the third why late detection is expensive. Reliability engineering already combines the first and third perspectives for physical systems through failure mode analysis and fault trees (Gideon et al., 2019), and research on human error causation in high-risk industrial work shows the value of the second (Obriki et al., 2020).

2.1 From Requirement Defect to Field Failure

Let N denote the set of needs held by stakeholders and imposed by the operating environment E , S the specified requirement set, and I the implemented system. Verification establishes that I satisfies S ; validation establishes that S , and therefore I , satisfies N in E (IEEE Computer Society, 2014). A requirements-originated failure is an operational event in which I satisfies S but I does not satisfy N under the conditions of E that actually occur. No increase in verification strength can detect such a failure, because the verification reference is itself the defective artifact. The testing literature expresses the same idea through the reachability, infection, propagation and revelation conditions for fault detection (Ammann et al., 2016). Where a requirement omits a condition, tests derived from it never reach the relevant state; where a requirement states the wrong behaviour, the oracle accepts the wrong output, so revelation fails even when the other three conditions hold.

Coverage criteria sharpen the argument. Adequacy criteria measure the thoroughness of testing relative to a reference, either the program structure or the specification (Zhu et al., 1997). Structural coverage cannot detect missing functionality, since absent code has no statements or branches to cover, and specification coverage is only as complete as the specification. Survey evidence on unit testing practice

shows that developers write tests from their own understanding of intended behaviour and report difficulty in deciding what the correct output should be (Daka et al., 2014), which means that developer tests reproduce the developer's reading of the requirement, including any misreading. Independent test design helps only when testers can access the needs and environment the specification failed to capture (Kaner et al., 1999). These observations define the first design principle of the framework developed in Section 6: containment of requirements defects must rely on references other than the specification, such as stakeholders, operating profiles, interface partners and physical constraints.

2.2 Latent Conditions and Organisational Accident Theory

Safety science explains major accidents as the alignment of latent conditions, created by designers, managers and procedures, with active failures at the point of operation, where each defensive layer contains holes that are harmless until they line up. Requirements decisions are archetypal latent conditions. They are taken early, by people distant from operation, and they can remain dormant for years until a trajectory, load, duration or user behaviour activates them. Research on incident investigation in industrial operations shows that investigations which stop at the proximate act produce corrective actions that do not prevent recurrence, whereas investigations that trace organisational and design contributors support prevention-oriented learning (Obriki et al., 2024b; Obogo et al., 2021). Systematic reviews of hazard identification practice make the complementary point that hazards not identified at design time are rarely controlled later (Obogo et al., 2019).

Two further implications follow for requirements. The first concerns near misses. Requirements defects that are caught late but before a failure, for instance during acceptance testing or through a user complaint, are the software analogue of near-miss reports, and safety research shows that systematically analysed near-miss data are among the most informative sources for prevention (Arumosoye et al., 2019). The second concerns reporting culture. Maturity models for safety learning associate sustained improvement with organisations that treat reports as learning inputs rather than disciplinary evidence (Arumosoye et al., 2021). Compliance research in fiscal settings distinguishes enforced compliance, which follows perceived power, from voluntary cooperation, which follows trust (Kirchler, 2007), and shows that intrinsic motivation and social norms matter alongside enforcement (Alm, 2019). Applied to requirements, a review regime perceived as coercive tends to become ceremonial: checklists are completed, but omissions are not reported because reporting them invites blame.

2.3 The Economics of Late Detection

The cost-of-late-detection argument can be stated precisely. Index lifecycle phases $k = 0, 1, \dots, n$, with phase 0 the requirements phase and phase n operation. Let $c(k)$ be the cost of repairing a requirements-origin defect detected in phase k , and assume a geometric escalation $c(k) = c(0) r^k$ with escalation ratio $r \geq 1$. Let $p(k)$ be the containment probability of phase k , that is, the probability that a requirements defect still present at the start of the phase is detected during it, with $p(n) = 1$ because operation eventually reveals any defect that matters. The expected repair cost of one requirements defect is then given by Equation (1). Testing texts tie the economics of testing to the stage at which defects are found (Myers et al., 2011), and cost governance research shows that such relationships are useful only as estimable quantities (Adesuyi et al., 2023).

$$E[C] = c(0) \sum_{k=0}^n r^k p(k) \prod_{j < k} (1 - p(j)) \quad (1)$$

Separating the requirements phase from the rest gives $E[C] = c(0) [p(0) + (1 - p(0)) R]$, where R is the expected relative cost of a defect that escapes the requirements phase. The marginal value of raising

requirements-phase containment is therefore $dE[C]/dp(0) = -c(0)(R - 1)$. An organisation should invest in upstream containment whenever the cost of raising $p(0)$ by a small amount Δ is lower than $\Delta \cdot c(0) \cdot (R - 1)$ multiplied by the expected number of requirements defects. The expression makes the conditions of the argument explicit. R increases steeply with r and decreases with downstream containment, but, as Section 2.1 showed, downstream containment is structurally weak for requirements defects because tests share the defective oracle. Simulation modelling of process changes and their financial impact offers a practical way to estimate such parameters from historical project data rather than from textbook ratios (Tonoyan et al., 2022b).

A worked illustration, using parameters chosen for exposition rather than estimated from data, shows the magnitudes involved (Figure 1). With five phases after requirements and a downstream containment probability of 0.35 at each intermediate phase, R is about 10.0 when $r = 2$ and about 78 when $r = 3.2$. Raising $p(0)$ from 0.3 to 0.6 then reduces the expected cost per requirements defect from about 7.3 to 4.6 units of $c(0)$ in the first case and from about 55 to 32 units in the second. For a flat escalation of $r = 1.35$, typical of iterative delivery with cheap rollback, R is only about 2.4 and the same improvement saves less than half a unit. The argument for upstream investment is therefore conditional rather than universal. It is strongest where r is large because consequences are irreversible, hardware or contracts are committed early, or regulatory approval must be repeated, and it is weakest where delivery is incremental, feedback is fast and failure consequences can be reversed. Analyses of linked economic value chains make an analogous point: losses introduced at an early stage compound through every subsequent stage that depends on it (Jones et al., 2021).

Two refinements are needed before the model is used. First, repair cost understates the full cost of an escaped requirements defect because operational failures carry consequence costs (lost missions, injuries, regulatory sanctions, contractual penalties) that are not proportional to repair effort. Where such consequences are possible, $c(n)$ should be replaced by the expected consequence cost. Second, the widely quoted escalation ratios derive largely from studies of large, sequential projects in earlier decades, and later experience with iterative methods suggests flatter curves in some settings (IEEE Computer Society, 2014). The model therefore treats r as a parameter to be estimated for each organisation and defect class, the form in which it enters proposition P5.

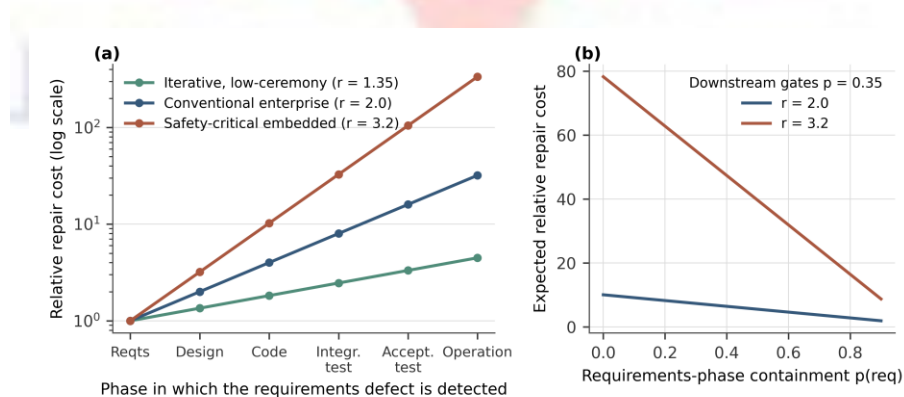


Figure 1. Illustrative late-detection economics for a requirements-origin defect: (a) relative repair cost by detection phase under three escalation ratios; (b) expected repair cost as a function of requirements-phase containment, with downstream containment of 0.35 per phase

3 Review Design and Analytical Method

The study is a critical integrative review. An integrative design was chosen over a conventional systematic review with meta-analysis because the relevant evidence is heterogeneous in kind: controlled experiments on inspection techniques, industrial case studies of traceability, mapping studies of link recovery, normative standards, and public investigation reports on individual failures. These sources cannot be pooled statistically, but they can be compared on the mechanisms they describe and the conditions under which those mechanisms operated. The protocol borrows the transparency requirements of systematic reviewing in software engineering (Kitchenham et al., 2009) and the reporting logic of the PRISMA statement (Page, 2021), while acknowledging that the aim is theory development rather than an effect estimate. Method selection follows guidance that the research question, not the availability of a technique, should determine the empirical approach (Easterbrook et al., 2008).

3.1 Sources, Search Strings and Time Window

Academic literature was sought in IEEE Xplore, the ACM Digital Library, Scopus, Web of Science Core Collection and SpringerLink, covering publications up to December 2024. Foundational works published before 1995 were retained where they define a construct used in the taxonomy, such as the analysis of the requirements traceability problem (Gotel et al., 1994). Four search strings were used and adapted to each database's syntax. The first targeted defect causation: (requirement* NEAR/3 (defect* OR error* OR fault* OR quality)) AND ("root cause" OR causation OR "failure analysis" OR "defect prevention" OR containment). The second targeted traceability: traceab* AND ("change impact" OR "link recovery" OR "trace maintenance" OR "safety case"). The third targeted validation: (validation OR "acceptance test*" OR "operational profile") AND requirement* AND (failure OR escape* OR field). The fourth targeted cases: software AND (failure OR mishap OR accident) AND (inquiry OR "investigation board" OR "lessons learned").

Backward snowballing from reference lists and forward snowballing from citing works were applied to the most central sources, following the practice of earlier systematic reviews in the field (Dybå et al., 2008). Standards were obtained from the issuing bodies, notably the requirements engineering standard ISO/IEC/IEEE 29148 and the test process standard ISO/IEC/IEEE 29119-2 (ISO/IEC/IEEE, 2018), together with the software engineering body of knowledge (IEEE Computer Society, 2014). Public failure cases were identified from inquiry board reports, agency mishap investigation reports, government audit reports and regulatory findings. Secondary retellings were used only to locate the primary investigation and were never coded directly.

3.2 Eligibility, Screening and Appraisal

Table 1 sets out the protocol decisions. Screening proceeded in two stages: titles and abstracts were screened against the inclusion criteria, and retained records were then read in full and screened again. Appraisal used instruments matched to the type of evidence. Empirical case studies were appraised against published guidelines for case study research in software engineering, with attention to the clarity of the unit of analysis, triangulation of data sources and the presence of a chain of evidence from data to conclusion (Runeson et al., 2009). Experiments were appraised for defect realism and participant representativeness. Standards were appraised for their normative scope and were used to define expected controls, not as evidence that those controls work. Investigation reports were appraised for independence of the investigating body, access to the original artifacts and the explicitness of the causal chain. Screening counts are not reported because the aim was conceptual saturation rather than enumeration.

Table 1. Review protocol for requirements-related failure causation

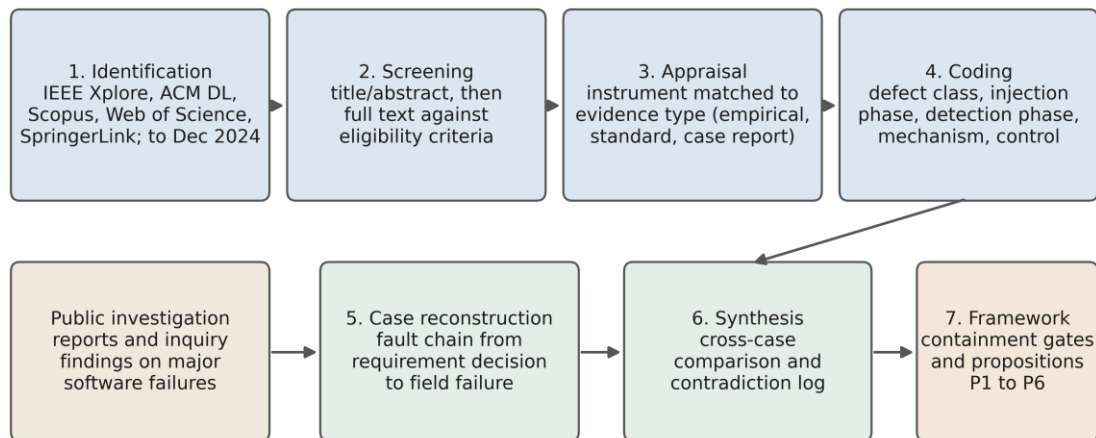
Protocol element	Decision rule for this review	Safeguard against bias
Sources	Five academic databases to December 2024; ISO/IEC/IEEE standards; independent investigation and audit reports on software failures	Primary investigation reports coded; secondary retellings used only to locate them
Inclusion	Studies reporting requirements defect classes, origin or detection phase, causation analysis, trace link maintenance or recovery, or validation practice with a failure or escape outcome	Inclusion judged on the reported outcome, not on keywords
Exclusion	Vendor material; opinion without a described mechanism; defect studies coded only by symptom; language-tool accuracy studies with no link to failure outcomes	Excluded full texts logged with the reason
Appraisal	Case studies against case study guidelines; experiments for defect realism; standards for normative scope; investigations for independence and explicit causal chain	Standards never treated as efficacy evidence
Case selection	Publicly investigated failure; identifiable requirement decision; documented consequence; spread across domains and decades	One boundary case with non-requirements origin retained as a rival test
Coding	Dominant and contributing defect class, injection phase, detection phase, missed containment point, control implied	Classification requires an explicit statement in the investigation record
Synthesis	Mechanism comparison across sources with a contradiction log	Conflicting readings retained and adjudicated in the text

3.3 Case Reconstruction and Synthesis Logic

Figure 2 summarises the workflow from identification to framework construction. For each public case, the fault chain was reconstructed backward from the observed failure to the proximate technical fault, then to the specification clause or its absence, and finally to the requirement decision that allowed it. A case was coded to a defect class only when the investigation record identified that decision, the containment point at which it could have been caught, and the resulting failure. Where the record supported more than one class, one dominant class and one or more contributing classes were recorded. The rule prevents labelling every failure a requirements failure in hindsight. Credit assignment across several contributing causes is known to be sensitive to the attribution rule chosen, as studies of multi-touch attribution models demonstrate in another field (Sanni et al., 2022), so the dominant class is reported together with contributing classes rather than as a single label.

Synthesis followed a mechanism, context and outcome logic. For each defect class, the review compared the mechanism by which the defect is injected, the contexts in which it escapes and the outcomes it produces, and it recorded contradictions between sources in a log rather than resolving them by exclusion. The most important contradiction, concerning the relative prevalence of ambiguity versus

omission, is adjudicated explicitly in Section 4.6. The resulting taxonomy and gate design were then treated as a design artifact whose requirements are derived from the synthesis and whose validation is deferred to the empirical agenda, in keeping with design science methodology for software engineering (Wieringa, 2014).



Contradictory readings of a case are logged, not resolved by exclusion; a case is coded to a defect class only when the investigation record identifies the requirement decision, the missed containment point and the resulting failure.

Boundary cases (failures with a non-requirements origin) are retained as rival-explanation tests.

Figure 2. Review, case reconstruction and synthesis workflow used to derive the taxonomy and the RDOC framework

4 A Causation Taxonomy of Requirements Defects

The taxonomy classifies requirements defects by the mechanism through which they enter the requirement set rather than by the symptom they eventually produce. Two criteria governed its construction. Each class must correspond to a distinct question that, if asked at the right time, would have exposed the defect, and each class must imply a distinct primary control. Symptom-based classes fail the second criterion, because one symptom can arise from an omitted stakeholder, a vague statement or a lost change link. The quality characteristics of individual requirements (necessary, unambiguous, complete, singular, feasible, verifiable, correct) and of requirement sets (complete, consistent, feasible, comprehensible, able to be validated) defined in ISO/IEC/IEEE 29148 provide the normative baseline (ISO/IEC/IEEE, 2018). The five classes below map onto those characteristics but add the lifecycle dimension that the standard does not address: when each defect is typically injected and why it escapes.

4.1 Elicitation Failure: Missing Stakeholders, Tacit Knowledge and Unexamined Operating Envelopes

Elicitation failure occurs when a need held by a stakeholder, or a condition imposed by the operating environment, never enters the requirement set. Three mechanisms dominate. The first is an omitted stakeholder: operators, maintainers, auditors, regulators and adversaries are routinely absent from elicitation, and their needs surface only when they use or attack the system. Homogeneous elicitation

groups share blind spots, which is why team diversity matters (Amayo et al., 2023c). The second is tacit knowledge: experts omit what they consider obvious, so the requirements capture exceptions and leave out the rules everyone knows. The third, and most dangerous in the cases reviewed, is an unexamined operating envelope: assumptions about duration of continuous operation, load, rate, physical range, units, data quality and environmental conditions that are held implicitly by the original designers and never written down. Risk models developed for industrial work in extreme environments illustrate the value of making the envelope explicit, since hazards grow nonlinearly as conditions move outside the range for which controls were designed (Arumosoye et al., 2018).

Security and privacy requirements are a recurrent category of omission. Reviews of access control policy in bring-your-own-device environments show that privacy requirements are frequently added after the architecture is fixed (Ladapo et al., 2023b), and field evidence on device adoption in administrative offices documents user practices that fall outside the configurations designers assumed (Onche et al., 2023). Privacy-by-design frameworks for customer relationship platforms and for corporate transactions both argue that privacy constraints must be elicited as requirements rather than retrofitted as controls (Badmus et al., 2024; Ominyi, 2021), and legal and ethical risk models for data protection make the same point for governance obligations (Mbonu et al., 2018). Network protocol abuse offers a particularly clear illustration: amplification attacks against time synchronisation services exploit features that behave exactly as specified, so the defect lies in the absence of a misuse requirement rather than in the implementation (Jimoh et al., 2024). Agentic artificial intelligence systems raise a newer version of the problem, since excessive agency is at root a missing constraint on the authority an agent may exercise (Nwakamma et al., 2024b). Zero trust architectures respond to the same class of defect by requiring that trust assumptions be stated explicitly rather than inherited from network position (Ahmed et al., 2021).

Elicitation failures are hard to detect because their carrier artifact is the absence of text. Their observable signals are indirect: new requirements appearing during acceptance testing, change requests labelled as enhancements, incidents concentrated in unusual operating conditions, and workarounds adopted by users. Effective controls work by introducing a reference other than the specification. Stakeholder maps and role-based walkthroughs introduce missing perspectives; misuse and abuse cases introduce the adversary; scenario analysis introduces conditions that designers did not consider, as scenario-based modelling does for strategic planning (Filani et al., 2023); and an explicit operating-envelope register records, for each requirement, the ranges and durations within which it has been validated. Facility design research offers a cross-domain parallel: laboratories designed to technical standards but without analysis of real workflows suffer throughput and error problems that only appear in use (Ogbete et al., 2018; Ogbete et al., 2022). Aligning requirements with strategic intent, rather than with the preferences of whichever stakeholders are most available, is a further safeguard, as research on aligning operational decisions with enterprise strategy suggests (Akin-Oluyomi et al., 2023b).

4.2 Ambiguity: Lexical, Syntactic, Referential and Vague Statements

A requirement is ambiguous when it admits more than one reasonable interpretation by the readers who must act on it, including designers, developers, testers and acceptance authorities (ISO/IEC/IEEE, 2018). Five forms recur. Lexical ambiguity arises from a word with several senses, such as record, account or close. Syntactic ambiguity arises from structure, as in a statement that the system shall notify the supervisor and the operator of alarms in zone A, where it is unclear whether the zone restriction applies to both recipients. Referential ambiguity arises from pronouns and anaphora whose antecedent is uncertain. Quantifier and scope ambiguity arises from words such as all, each and any, particularly in

combination with negation. Vagueness differs from ambiguity proper: a vague term such as fast or intuitive has borderline cases rather than discrete readings, and no test can decide the borderline. The software engineering body of knowledge treats both as threats to verifiability (IEEE Computer Society, 2014).

The harmful case is unrecognised ambiguity, in which each reader is confident of a different interpretation. Recognised ambiguity prompts a question; unrecognised ambiguity produces two consistent but incompatible artifacts, for example code that implements one reading and tests that check another, or, worse, code and tests that share one reading while the customer holds the other. Contract research provides a close analogue: disputes in high-value service contracts cluster around terms that the parties read differently without noticing at signature (Uchechi et al., 2019), and cross-functional legal education for engineering teams has been associated with fewer contract errors, which suggests that exposing engineers to the reader's perspective reduces misinterpretation (Anichukwueze et al., 2022). The practical controls follow the same logic. Fit criteria attach a measurable acceptance condition to each requirement; a controlled glossary fixes the sense of domain terms; perspective-based inspection asks reviewers to read as a tester, a designer and a user in turn; and concrete examples of expected behaviour, agreed with stakeholders, convert statements into checkable cases (Kaner et al., 1999). Business analysis frameworks that move requirements from ambiguity to automation-ready form apply the same discipline upstream, structuring the business problem, its rules and its exceptions before any specification is written, which matters most where the implementing system will execute rules without human interpretation (Alieme et al., 2024). Automated language analysis can flag candidate ambiguities but cannot decide which reading the stakeholder intended.

4.3 Inconsistency: Intra-Specification Conflict and Cross-Artifact Drift

A requirement set is inconsistent when no behaviour can satisfy all of its members simultaneously, or when two artifacts assign incompatible properties to the same entity. Four forms are distinguished. Logical conflict occurs within one specification, for instance a response-time requirement that cannot be met together with a mandatory encryption step on constrained hardware. Interface inconsistency occurs between the specification and an interface definition, where two parties disagree on units, coordinate frames, encodings, value ranges or timing. Regulatory inconsistency arises when obligations from different jurisdictions or regimes conflict, a problem well documented in digital trade and in critical infrastructure compliance (Jones et al., 2024b; Adegbite et al., 2020). Temporal drift occurs when artifacts that were consistent at baseline diverge because one was updated and the other was not. Formal methods detect logical conflict only in formalised specifications; in natural-language requirements, inconsistencies are found, if at all, by human cross-checking.

Inconsistency concentrates at organisational boundaries. Governance research on multi-contractor construction projects shows that risks cluster at the interfaces between contractors, where each party assumes the other controls the hazard (Obogo et al., 2019b), and architecture research on converging information and operational technology networks makes a similar case for aligning design principles across organisational units (Adegbite et al., 2022). Cross-organisational data management raises the same issue for shared data definitions (Sanni et al., 2023). Controls drawn from finance are instructive: reconciliation procedures detect inconsistency by independently recomputing the same quantity from two sources and investigating any difference (Dogbatsey et al., 2019), and continuous monitoring for cloud misconfiguration applies the same principle to deployed configurations that have drifted from their intended state (Aliliele et al., 2023a). In requirements terms, the corresponding controls are a canonical

data dictionary that fixes units and ranges for every shared quantity, executable interface contract tests run by both parties, API governance that treats interface definitions as controlled artifacts (Aliliele et al., 2023b), and a conflict register in which each detected inconsistency has an owner and a resolution deadline. Standardisation of instruments across a portfolio, as in the harmonisation of documentary tax practice, reduces the number of places where inconsistency can arise (Uchechi, 2018).

4.4 Traceability Loss: Broken Links and Invisible Change Impact

Traceability is the ability to follow the life of a requirement both backward to its origins and forward to the design, code, tests and safety analyses that realise it. The classic analysis of the traceability problem distinguished pre-requirements-specification traceability, which records where a requirement came from and why, from post-specification traceability, which links it to downstream artifacts, and argued that most practical problems lie in the former (Gotel et al., 1994). Reference models derived from industrial practice showed that low-end users record simple links for compliance while high-end users also record rationale, decisions and assumptions (Ramesh et al., 2001). Later work surveyed traceability in model-driven development (Winkler et al., 2010) and framed it as a strategic investment (Cleland-Huang et al., 2012). Traceability loss, as a defect class, occurs when these links are absent, wrong or out of date at the moment they are needed, which is almost always when something changes.

Information retrieval approaches recover candidate links automatically by comparing the text of artifacts, using vector space and latent semantic indexing models (De Lucia et al., 2007). A systematic mapping of a decade of this work found that most evaluations used small datasets, reported precision and recall on artifacts from a handful of projects, and seldom examined whether engineers could use the recovered links in practice (Borg et al., 2014). Precision, the share of recovered links that are correct, and recall, the share of true links that are recovered, trade against each other, and their relative importance depends on the use. For change impact analysis in safety-related systems recall dominates, because a missing link hides an affected analysis, while low precision imposes a vetting burden that engineers may not sustain. Detection problems in fraud analytics face the same asymmetry between missed cases and false alarms (Atakpa et al., 2022; Adelanwa et al., 2023b).

Traceability loss converts a correct change into a failure. When a requirement or design parameter is revised, every artifact that depended on the old value (hazard analyses, test cases, user documentation, training material, interface partners) must be reviewed. If the links are missing, the change appears local. Temporary project organisations exacerbate the risk because the people who hold the undocumented links disperse when the project ends (Arumosoye et al., 2024). Regulated domains have developed change-triggered assessments that are instructive: data protection impact assessments must be revisited when processing changes (Mbonu et al., 2022a), and data sensitivity classification schemes are paired with regulatory traceability mechanisms so that obligations follow data through transformations (Aliliele et al., 2024a). The corresponding requirements controls are a change-impact gate that blocks approval of a change until the affected links have been reviewed, suspect-link flagging that marks downstream artifacts whenever an upstream artifact changes, and trace completeness measured on critical requirements rather than averaged over all.

4.5 Validation Failure: Verifying the Wrong Thing Correctly

Validation failure occurs when the requirement set, or the system built from it, is checked against a reference that does not represent the real needs in the real operating environment. It differs from elicitation failure in that the opportunity to discover the gap, by exercising the system against realistic

conditions, existed but was not taken or was taken against the wrong reference. Four mechanisms recur: validation against an assumed rather than measured operating profile; validation in a simulated environment that omits conditions present in the field; reuse of a component validated in one context without revalidation in the new one; and acceptance by proxies, such as managers or product owners, instead of the people who will operate the system. The software testing process standard defines test planning, design and completion processes (ISO/IEC/IEEE, 2018), but conformance to these processes does not guarantee that the test basis reflects the operating world.

Evidence from user acceptance testing in small businesses shows that structured validation frameworks can be adapted to lean environments, but that acceptance compressed into the final days before deployment tends to confirm rather than challenge the specification (Eyetsemitan et al., 2023b). Research on accelerated regulatory approval in pharmaceuticals formalises the trade-off between speed, safety and evidence quality that underlies every compressed validation phase (Eze et al., 2024b), and release gating research for machine learning models shows that automated evaluators can share the biases of the systems they judge (Nwakamma et al., 2024a), which is the validation analogue of a test oracle that shares the specification's defect. Effective controls therefore introduce independent references: operational profiles measured from comparable systems, end-to-end testing with production-like data, shadow operation in parallel with the incumbent system, staged rollout with monitored exposure, and independent validation teams with access to users. Test adequacy measured against the specification remains necessary, but it answers a different question (Zhu et al., 1997).

Table 2 consolidates the five classes. For each class it states the defining question that would expose the defect, the phase in which it is typically injected, the reason downstream testing tends to miss it, the signal through which it becomes observable and the primary containment control. Rows follow lifecycle position rather than frequency, for reasons given in Section 4.6.

Table 2. Causation taxonomy of requirements defects: injection, escape and containment

Defect class	Defining question	Typical injection	Why testing misses it	Observable signal	Primary containment control
Elicitation failure	Whose need or which operating condition is absent?	Elicitation; reuse decisions	No requirement, so no test and no oracle	Late new requirements; incidents in unusual conditions; user workarounds	Stakeholder map, misuse cases, operating-envelope register (G1)
Ambiguity	Can two competent readers act on this differently?	Specification writing	Tests encode the tester's reading	Clarification requests; divergent tests for one requirement	Fit criteria, glossary, perspective-based inspection (G2)
Inconsistency	Can all statements and interfaces hold at once?	Specification; interface definition	Each party tests against its own artifact	Integration anomalies; unit or range mismatches	Data dictionary, contract tests, conflict register (G2)
Traceability loss	Which artifacts depend on this,	Change after	Regression tests linked to	Post-change incidents; stale	Change-impact gate, suspect-link

Defect class	Defining question	Typical injection	Why testing misses it	Observable signal	Primary containment control
	and were they updated?	baseline	the old value	hazard analyses	flags (G3)
Validation failure	Was the system checked against the real operating world?	Validation planning; reuse	Test basis equals the specification	Field failures under conditions never tested	Operational profile, shadow operation, staged rollout (G5)

4.6 Adjudicating Conflicting Evidence on Prevalence

The literature contains an apparent contradiction. Studies of requirements inspections and document analysis frequently report ambiguous, vague or poorly structured statements as the most numerous defect type, while analyses of serious field failures emphasise omissions, unexamined assumptions and inadequate validation. Both findings can be correct, and the reason is a selection effect in how defects are sampled. Inspections sample defects that are visible on the page, and ambiguity is by definition present in the text, whereas an omission leaves nothing to find. Field failure investigations sample defects that escaped every earlier filter, so they over-represent exactly the classes that inspections cannot see. Research on non-destructive testing of welds shows the same structure in physical inspection: the probability of detection depends on the type and orientation of the flaw, so the population of detected flaws is not a representative sample of the flaws present (Atta et al., 2021). Analytical chemistry offers a further analogy, since contamination below an assay's detection limit is absent from the results rather than measured as zero (Tawose et al., 2023).

The adjudication has two consequences. First, prevalence figures for requirements defect classes are not comparable across studies unless the sampling point, inspection output or field escape, is stated. Second, the classes that dominate inspection counts are not necessarily the classes that dominate failure cost, so allocating prevention effort by inspection counts would systematically under-invest in elicitation and validation. The case analysis in Section 5 is consistent with this reading, and proposition P4 in Section 6 turns it into a testable claim about inspection yield. A conservative reading of the case study literature supports treating the two sampling points as complementary sources of evidence rather than as competing measures of the same quantity (Runeson et al., 2009).

5 Public Failure Cases Read Through the Taxonomy

This section applies the coding protocol of Section 3.3 to eight widely discussed failures. The descriptions are deliberately held at the general level supported by the public investigation findings, because popular retellings often compress a multi-causal finding into one sentence and occasionally misstate it. For each case the dominant and contributing defect classes, the injection phase and the phase in which the defect was first acted on are recorded in Table 3, and Figure 3 represents the resulting escape distances. The purpose is to test whether the taxonomy classifies them consistently and whether the cross-case pattern matches Section 2. Investigation research in industrial safety stresses that the value of case analysis lies in extracting transferable prevention lessons rather than in assigning individual fault (Obriki et al., 2024b), and that principle governs the treatment below.

5.1 Reused Components and Unvalidated Assumptions: Ariane 5 Flight 501 and the Patriot Clock Drift

On its maiden flight in June 1996, the Ariane 5 launcher veered off course shortly after lift-off and was destroyed. The inquiry board found that the inertial reference software had been reused from Ariane 4. An alignment function that served no purpose after lift-off continued to run, and it converted a floating-point value related to horizontal velocity into a 16-bit signed integer. Because Ariane 5 followed a different early trajectory from Ariane 4, the value exceeded the representable range, the resulting exception was not handled, and both inertial reference units, running identical software, shut down. The board also found that Ariane 5 trajectory data had not been included in the functional requirements of the inertial reference system and had not been used in its testing, and that protection of the conversion had been omitted on the basis of Ariane 4 analysis. The dominant class is validation failure (reuse without revalidation against the new envelope), with elicitation failure contributing because that envelope was never stated as a requirement. Redundancy offered no protection, since two copies of the same software share the same specification defect, a common-mode pathway familiar from reliability engineering (Gideon et al., 2019).

During the 1991 Gulf War, a Patriot air defence battery at Dhahran failed to intercept an incoming Scud missile, which struck a barracks and killed 28 soldiers. The government audit found that the system tracked time in tenths of a second and converted it to seconds using a fixed-point approximation whose truncation error grew with continuous operation. After roughly 100 hours of uninterrupted running the accumulated error was about a third of a second, enough to shift the tracking gate so that the target was not engaged. The system had been designed around mobile operation of limited duration, and although a software correction was being distributed, it arrived the day after the attack. The dominant class is elicitation failure: the duration of continuous operation, a core parameter of the operating envelope, was an implicit assumption rather than a stated and validated requirement. Research on emergency response readiness in process facilities makes the general point that capability must be assessed under the conditions in which it will actually be demanded (Arumosoye et al., 2023).

5.2 Interface Contracts Across Organisations: The Mars Climate Orbiter

The Mars Climate Orbiter was lost in September 1999 during orbit insertion. The mishap investigation board identified the root cause as the failure to use metric units in a ground software file used in trajectory modelling: thruster impulse data were produced in imperial units (pound-force seconds) where the interface specification required metric units (newton-seconds). The navigation team's models therefore underestimated the effect of small thruster firings, and the spacecraft approached Mars on a much lower trajectory than planned. The board also found that anomalies in the navigation data had been noticed during cruise but were not resolved through formal processes, and that end-to-end verification of the interface was inadequate. In taxonomy terms, the requirement existed and was correct; the defect was an inconsistency between an implementation artifact and the interface specification at a boundary between organisations, with traceability loss and validation failure contributing. Its lesson is that a correct requirement that is not enforced across an organisational interface provides no containment, which is why multi-party governance research emphasises explicit ownership of interfaces (Obogo et al., 2019b). Cooperation models for security across national borders make a comparable point: shared objectives fail without agreed operating protocols at the boundary (Liadi, 2024c).

5.3 Safety Functions Allocated to Software: Therac-25 and the 737 MAX MCAS

Between 1985 and 1987 the Therac-25 radiation therapy machine was involved in a series of accidents in which patients received massive overdoses, several of them fatal. Earlier machines in the product line had used hardware interlocks to prevent the high-energy beam from operating without the correct beam-shaping components in place; the Therac-25 relied on software for this safety function, and part of its software was derived from the earlier machines, where hardware interlocks had masked latent software faults. Investigations identified a race condition triggered when experienced operators edited treatment parameters quickly, together with other software faults, error messages that gave operators little information and were routinely overridden, and an incident response that fixed individual faults without addressing the design. The dominant class is validation failure: the reallocation of a safety requirement from hardware to software was not accompanied by validation of the software against realistic operator behaviour. Elicitation failure contributes because the interaction patterns of expert operators were not captured. Human error causation research is relevant here because it locates such operator actions within the design conditions that made them hazardous rather than treating them as the cause (Obriki et al., 2020).

The two fatal accidents involving the Boeing 737 MAX in October 2018 and March 2019 have been examined by accident investigators, regulators and legislative inquiries. At a general level those investigations describe a flight control function, the Manoeuvring Characteristics Augmentation System, that commanded nose-down stabiliser trim on the basis of angle-of-attack input. During development the function's authority was expanded, including operation at low speed, and it relied on input from a single sensor. Subsequent reviews found that the revised function was not fully reflected in the safety assessment and in the information provided to flight crews, and that assumptions about how quickly pilots would recognise and respond to an erroneous activation were not borne out. The dominant class is traceability loss: a design change was not propagated to the hazard classification, the crew information and the validation assumptions that depended on the earlier version. Validation failure contributes through the untested assumption about human response time.

5.4 Organisational-Scale Failures: London Ambulance Dispatch and Denver Airport Baggage

In 1992 the London Ambulance Service introduced a computer-aided dispatch system that, shortly after going live, produced growing delays and lost calls, and the service reverted first to partial and then to manual operation. The subsequent inquiry identified an unrealistic implementation timetable, a procurement process that favoured the lowest bid, limited involvement of ambulance crews and control room staff, an assumption that vehicle location and status data would be complete and accurate, inadequate testing under full load, and insufficient training. The dominant class is elicitation failure, specifically the failure to capture how crews and dispatchers actually worked and the real quality of the data the system depended on, with validation failure contributing through the absence of realistic load testing. Procurement criteria that reward price over demonstrated capability shift risk onto the operator (Okonkwo et al., 2021), and partnership governance research stresses accountability for outcomes as well as deliverables (Aminu-Ibrahim et al., 2024).

The automated baggage system at Denver International Airport was intended to serve the entire airport with an integrated network of carts and tracks. Requirements were defined late relative to the construction schedule and changed repeatedly as the airport and airlines revised their needs, and public tests in 1994 revealed misrouted and damaged baggage. The opening of the airport was delayed by about sixteen months, it opened with the automated system serving only part of the operation, and the system

was later abandoned in favour of conventional handling. The dominant class is elicitation failure, since the requirements for a system of unprecedented scale were not stabilised or reconciled with physical constraints before construction, and the defect was first acted on during integration testing rather than in operation, giving a shorter escape distance than most other cases. Optimisation research on automated material handling shows how sensitive such systems are to layout and routing constraints that must be fixed early (Akanbi et al., 2024).

5.5 A Boundary Case and the Risk of Retrospective Relabelling: Knight Capital

In August 2012 Knight Capital deployed new trading software to its servers, but one server did not receive the new code. A configuration flag that had been repurposed for the new functionality activated obsolete code on that server, which sent a large volume of unintended orders over roughly 45 minutes and produced a loss that regulatory findings put at about 460 million US dollars. The regulator's findings concentrated on deployment procedures, the absence of independent review of the deployment, the retention of dead code and the lack of automated controls to halt erroneous activity. The case is included as a rival-explanation test. Its dominant origin is deployment and configuration management, not requirements; the only requirements-related element is the reuse of a flag whose earlier meaning had not been retired, which is a weak form of traceability loss. A taxonomy that classified this case as a requirements failure would be over-inclusive. Retaining it guards against the retrospective relabelling that affects many analyses of software failure, in which investigators who know the outcome find a missing requirement behind every incident. Evidence on enterprise cybersecurity trends similarly shows that operational and configuration failures form a distinct class of risk with their own controls (Okoruwa et al., 2024).

5.6 Cross-Case Findings

Table 3 summarises the coding, and Figure 3 plots each case as an arrow from its injection phase to the phase in which it was first acted on. Four findings emerge. First, the dominant classes are elicitation failure, validation failure, inconsistency at an interface and traceability loss at a change; ambiguity is not the dominant class in any of the cases, which agrees with the selection-effect argument of Section 4.6. Second, escape distances are long: in seven of the eight cases the defect was first acted on only in operation, and in the one exception (Denver) it surfaced at integration testing, still late in the lifecycle. Third, verification strength and redundancy did not help, because the defect was common to every redundant channel and to the oracles used in testing. Fourth, warning signals existed before most failures: the Patriot duration advisory, the navigation anomalies on the Mars Climate Orbiter, earlier Therac-25 incident reports and staff concerns about the London dispatch system. In each case the near-miss channel existed but did not change the requirement or the validation plan, as Section 2.2 predicts. Log analytics research shows that such signals are often collected but not routed to anyone with authority to act (Mbonu et al., 2019).

Table 3. Public failure cases coded by dominant and contributing requirements defect class

Case (year)	Failure at a general level	Dominant class	Contributing classes	Missed containment point
Therac-25 (1985 to 1987)	Radiation overdoses after a safety function moved from hardware to software	Validation failure	Elicitation failure	G5: validation against real operator behaviour

Case (year)	Failure at a general level	Dominant class	Contributing classes	Missed containment point
Patriot, Dhahran (1991)	Clock drift over long continuous operation shifted tracking; missile not intercepted	Elicitation failure	Validation failure	G1: operating duration not stated or validated
London Ambulance CAD (1992)	Dispatch system degraded and was withdrawn soon after launch	Elicitation failure	Validation failure	G1 and G5: staff practice, data quality, full-load testing
Denver airport baggage (1994 to 1995)	Integrated baggage system failed tests; opening delayed; system later abandoned	Elicitation failure	Traceability loss	G1: unstable scope not reconciled with physical constraints
Ariane 5 Flight 501 (1996)	Reused inertial software overflowed on new trajectory; launcher destroyed	Validation failure	Elicitation failure	G5: reuse without revalidation against new envelope
Mars Climate Orbiter (1999)	Unit mismatch at a ground software interface; spacecraft lost	Inconsistency	Traceability loss; validation failure	G2 and G4: interface contract not enforced end to end
Knight Capital (2012)	Partial deployment activated obsolete code; large trading loss	Non-requirements origin	Weak traceability loss (repurposed flag)	Deployment control, outside RDOC scope
Boeing 737 MAX MCAS (2018 to 2019)	Expanded control function not fully reflected in safety assessment and crew information	Traceability loss	Validation failure	G3: change impact on hazard analysis and crew information

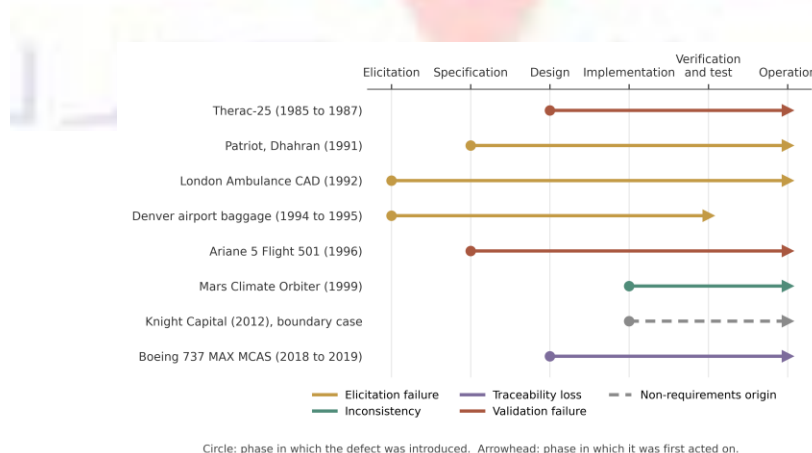


Figure 3. Escape distance of eight public failure cases, from the lifecycle phase in which the defect was introduced to the phase in which it was first acted on, coloured by dominant defect class

6 The RDOC Framework: Containment Gates, Causation Feedback and Propositions

The Requirements Defect Origin and Containment (RDOC) framework translates the taxonomy and the case findings into a lifecycle control design. It rests on three principles derived from the synthesis. Containment of requirements defects must use references other than the specification (Section 2.1). Containment effort should be allocated according to the escalation ratio and consequence profile of the system (Section 2.3). And escaped defects must be traced back to their origin class and used to revise the gate that missed them, because the case analysis shows that warning signals were available but did not change upstream controls (Section 5.6). Figure 4 presents the framework. Following design science practice, it is stated with its intended use, components and refuting evidence (Hevner et al., 2004; Wieringa, 2014).

6.1 Five Containment Gates Across the Lifecycle

Gate G1, at the end of elicitation, reviews stakeholder coverage and the operating envelope. Its evidence is a stakeholder map showing which roles, including maintainers, regulators and adversaries, contributed requirements, and an envelope register listing for each critical requirement the durations, loads, ranges, units and environmental conditions it assumes. Gate G2, at the end of specification, applies perspective-based inspection for ambiguity and a consistency check across the requirement set and its interface definitions, using the data dictionary and conflict register described in Section 4.3. Gate G3 operates whenever a baselined requirement or design parameter changes: it blocks approval until suspect links have been reviewed and affected hazard analyses, tests and user information have been updated. Gate G4, during verification, evaluates the adequacy of requirement-based tests and the independence of their oracles. Gate G5, before and during release, validates the system against a measured operational profile through end-to-end tests with realistic data, shadow operation or staged rollout. The gates map onto the requirements and test processes defined in the relevant standards (ISO/IEC/IEEE, 2018) but add explicit exit evidence for each defect class.

Two design choices guard against the ceremonial review identified in Section 2.2. First, gate performance is measured by finding yield and coverage rather than by the number of reviews held. Research on management safety walkthroughs found that the coverage and quality of walkthroughs, not merely their frequency, relate to safety culture outcomes (Obriki et al., 2019), and inspection standards in heavy engineering specify what must be examined and how findings are dispositioned rather than simply mandating inspection (Atta et al., 2022). Second, gate effort is scaled to risk. A risk-based approach to assurance allocates effort to the assets whose failure has the largest consequence (Fadayomi et al., 2019); in RDOC terms, G1 and G5 receive the most effort where the estimated escalation ratio and consequence cost are high, while in low-consequence, rapidly iterated components they may be reduced to lightweight checklists.

6.2 Causation Analysis of Escaped Defects

Every defect that escapes to a late phase or to operation enters a causation analysis with four steps. First, the fault chain is reconstructed backward from the failure to the proximate technical fault, to the specification clause or its absence, and to the requirement decision, using logs and artifacts in the way forensic analytics reconstruct event sequences (Mbonu et al., 2021). Second, the origin is classified into one dominant and any contributing classes by two coders working independently, with agreement measured by Cohen's kappa, $\kappa = (p_o - p_e) / (1 - p_e)$, where p_o is observed agreement and p_e is agreement expected by chance; low agreement signals that class definitions need refinement. Third, the

gate that should have contained the defect is identified. Fourth, a control change is proposed, assigned an owner and verified after implementation. A standardised procedure matters because procedural standardisation has been shown to reduce variation and delay in high-volume case handling (Uchechi et al., 2018), and corrective and preventive action research in pharmaceutical quality systems shows that preventive actions are effective only when root-cause methods are applied consistently and their effect is verified (Aneke et al., 2022a).

The choice of analytical technique affects what is found. The five-whys method is fast but follows a single causal chain and tends to stop at a convenient cause, often a person. Fault tree analysis represents combinations of conditions through AND and OR gates and therefore exposes common-mode pathways such as the identical software in both Ariane 5 inertial units (Gideon et al., 2019). Risk pathway models developed for industrial operations offer a similar structure for tracing how latent conditions combine with triggering events (Arumosoye et al., 2022). A requirements failure mode and effects analysis applies FMEA to the requirement set, rating each potential requirements defect for severity S, occurrence O and detectability D and ranking by the risk priority number $RPN = S \times O \times D$. Its known weaknesses, multiplication of ordinal scales and ties between very different risk profiles, argue for reporting the three ratings separately and for treating severity as a filter before ranking, as risk scoring research recommends for high-stakes decisions (Adesuyi et al., 2024). Whatever technique is used, the analysis must be blame-free, since, as Section 2.2 argued, attribution of fault to individuals suppresses the reporting on which the whole loop depends (Obriki et al., 2020; Kirchler, 2007).

6.3 Containment Metrics

Three metrics make the framework measurable. Phase containment effectiveness for phase k, $PCE(k)$, is the number of defects originated in phase k and removed in phase k divided by the number of defects originated in phase k. Requirements leakage, $L(req)$, is the number of requirements-origin defects found after gate G2 divided by all requirements-origin defects found. Severity-weighted escape distance, D , is defined in Equation (2), where $o(i)$ and $d(i)$ are the phase indices of injection and detection for defect i and $w(i)$ is its severity weight. D summarises how far requirements defects travel before detection and weights the long escapes that matter most. These metrics follow the principle, established in performance management research, that each indicator requires a defined owner, data source and response (Sakyi et al., 2022b), and they suit the automated monitoring that real-time KPI systems make possible (Tonoyan et al., 2024b; Edivri et al., 2024).

$$D = \frac{\sum_i w(i) (d(i) - o(i))}{\sum_i w(i)} \quad (2)$$

Three measurement problems must be addressed. First, origin phase recorded at triage is biased toward the detection phase, because the person triaging sees the symptom, so origin should be coded retrospectively during causation analysis. Second, the denominator of $PCE(req)$, the total number of requirements defects originated, is unknown. It can be estimated with a capture and recapture design: if two independent inspections of the same specification find n_1 and n_2 defects with m in common, the Lincoln and Petersen estimator gives an estimated total $N = n_1 \cdot n_2 / m$, with wide uncertainty when m is small, which should be reported (Wedraogo et al., 2024). Third, aggregate metrics can conceal poor containment on critical requirements, just as aggregate forecast accuracy can conceal systematic error for particular groups (Komi, 2023), so PCE and D should be reported separately for safety-critical, security-relevant and interface requirements. Performance measurement in regulatory oversight and in data-driven safety risk control shows that such stratified indicators are more informative than a single composite

(Anioke et al., 2023a; Obriki et al., 2018). Financial valuation of prevention investment, finally, should use the organisation's own estimate of r rather than published ratios, in line with valuation practice that integrates observed unit-level performance into pricing models (Tonoyan et al., 2023).

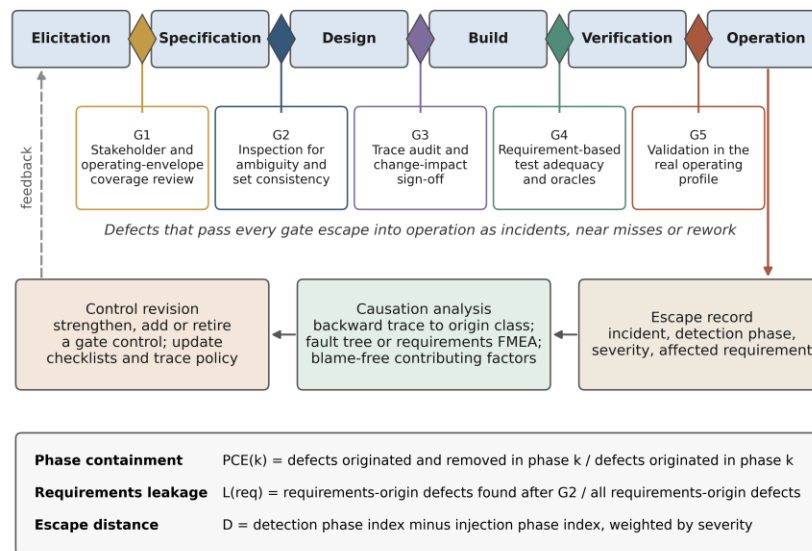


Figure 4. The Requirements Defect Origin and Containment (RDOC) framework: five containment gates across the lifecycle, causation feedback from escaped defects, and the three containment metrics

6.4 Propositions and Empirical Tests

Six propositions follow from the synthesis. They are stated so that each could be shown false by feasible data, and Table 4 pairs each with a test design and the result that would refute it. P1 concerns the limits of downstream detection and can be illustrated through the average percentage of faults detected, $APFD = 1 - (TF1 + \dots + TF_m) / (n \cdot m) + 1 / (2n)$, a standard measure for test prioritisation in which n is the number of tests, m the number of known faults and TF_i the position of the first test that reveals fault i (Li et al., 2007; Catal et al., 2013). APFD is computed only over faults that some test can reveal, so a requirement omission contributes nothing to it and remains invisible however the tests are ordered. P1 therefore predicts that, as specification-based coverage rises, the share of requirements-origin defects among escapes rises too.

P2 predicts that reused components lacking an explicit operating-envelope revalidation will show higher rates of field failures attributable to elicitation or validation defects than newly developed components of comparable size and criticality, generalising the Ariane 5 and Patriot findings. P3 predicts that trace completeness measured at change time, the share of changed requirements whose downstream links were reviewed before approval, will predict post-change incidents better than trace completeness measured at baseline, following the 737 MAX analysis. P4 predicts that inspection with operating-envelope and stakeholder-perspective checklists will raise the yield of omission findings more than the yield of ambiguity findings, following the selection-effect argument of Section 4.6. P5 predicts that the escalation ratio r , estimated from an organisation's own defect records, will be larger for defects that cross an organisational boundary, such as an interface, a supplier contract or a regulatory submission, than for defects confined within one team, following the Mars Climate Orbiter analysis. P6 predicts that organisations in which causation findings lead to a documented gate change within a defined interval will

show declining requirements leakage over subsequent releases, while organisations that record root causes without changing controls will show no trend. Case study guidelines provide the design standards for testing P2, P3 and P6 in industrial settings (Runeson et al., 2009).

Table 4. RDOC propositions, test designs and refuting results

Proposition	Test design	Result that would refute it
P1. Rising specification-based coverage raises the share of requirements-origin defects among escapes	Multi-release panel; regress origin-coded escape share on specification coverage, controlling for size and change volume	Share flat or falling as coverage rises
P2. Reuse without envelope revalidation raises elicitation and validation field failures	Matched comparison of reused and new components by size and criticality over equal exposure	No difference, or lower rate for reused components
P3. Trace review at change time predicts post-change incidents better than baseline trace completeness	Longitudinal change-level dataset; compare predictive fit of the two measures	Baseline completeness predicts equally well or better
P4. Envelope and perspective checklists raise omission yield more than ambiguity yield	Controlled inspection experiment with seeded omissions and ambiguities; checklist versus generic reading	Equal or smaller gain for omissions
P5. Escalation ratio is larger for defects crossing an organisational boundary	Estimate r by fitting repair cost against escape distance, stratified by boundary crossing	Overlapping or reversed estimates
P6. Causation findings that change gates within a set interval reduce requirements leakage	Interrupted time series across releases before and after adopting the feedback rule, with comparison units	No level or slope change relative to comparison units

7 Discussion

The review set out to explain why requirements defects produce a disproportionate share of serious failures, why their late detection is costly, and which controls contain them. The answer developed here is that requirements defects are latent conditions that verification cannot see, because verification takes the specification as its reference, and that their cost rises with escape distance at a rate that depends on the reversibility of consequences and on organisational boundaries. This section draws out theoretical and practical implications and then examines the strongest objections. This view aligns with fields that treat failures as products of design and organisational conditions (Gideon et al., 2019; Obriki et al., 2024b).

7.1 Theoretical Implications

Four implications follow for theory. First, the fault-chain account of testing and the latent-condition account of accidents combine naturally once the specification is recognised as a possible fault carrier. The testing literature explains why the defect is invisible to verification, and the safety literature explains why it remains dormant and why its activation looks like a surprise (Ammann et al., 2016; Obriki et al., 2020). Second, the selection-effect explanation of conflicting prevalence findings implies that studies of

requirements quality should report the sampling point as part of the construct definition, since inspection counts and field escapes measure different things. Third, the escalation ratio should be treated as a contingent parameter that varies with development model, domain and boundary crossing rather than as a universal constant; this reframing reconciles the classic cost argument with the experience of iterative development (Dybå et al., 2008). Fourth, validation should be theorised as a construct separate from verification strength, because the case evidence shows that highly verified systems failed through validation against the wrong reference. The traceability literature supports this separation by distinguishing the backward links that carry rationale and assumptions from the forward links that support verification (Gotel et al., 1994; Ramesh et al., 2001). Lifecycle evaluation of built facilities offers a parallel, since fitness for use is judged only in operation (Ogbete et al., 2023).

7.2 Practical Implications

For engineering and quality managers, the most immediate implication concerns data. Defect records should carry an origin phase and an origin class, coded at closure through causation analysis rather than guessed at triage, because without them PCE, requirements leakage and escape distance cannot be computed. A second implication concerns reuse: every reused component should carry an operating-envelope record stating the conditions under which it was validated, and reuse in a new context should trigger gate G5 for those conditions. A third concerns change: gate G3 should be mandatory for safety-relevant, security-relevant and interface requirements, with suspect-link flags generated by the requirements tool. These measures fit within enterprise risk management that integrates human resource strategy with operational analytics (Falemi et al., 2024) and within cybersecurity governance frameworks that already require documented controls and owners (Jimoh et al., 2023). Leadership matters for all three, since research on safety culture shows that leaders' responses to reported problems determine whether reports continue (Obogo et al., 2023; Obogo et al., 2019a).

Regulation is extending requirements obligations to new classes of system. Organisational readiness for the European Union Artificial Intelligence Act requires documented risk management, data governance and traceability for high-risk systems (Ominyi et al., 2024b), and regulatory readiness models in pharmaceuticals show how organisations can assess their capability against such obligations before market entry (Eze et al., 2022b). Firms entering heavily regulated service sectors face comparable barriers in which documentation of requirements and their verification is itself a condition of entry (Sanni et al., 2023c). Capability building is the final lever. Requirements analysts benefit from structured professional development (Joseph et al., 2024), transfer of operational knowledge from information technology operations into design roles brings the operating envelope into elicitation (Anyanwu, 2022), and employee readiness shapes whether new review practices are adopted or resisted (Akhigbe et al., 2024). Tools alone do not sustain improvement without strategy, operational discipline and workforce capability (Ilodigwe et al., 2024b).

7.3 Counter-Arguments and Boundary Conditions

The first objection comes from iterative development: if releases are small and feedback is fast, late detection is cheap, and upstream gates waste effort. The objection is correct within its boundary, and the cost model accommodates it, since a low escalation ratio implies a low marginal value of upstream containment. The boundary is reversibility. Where consequences cannot be undone, as in physical harm, irreversible financial transfers, lost spacecraft or regulatory sanction, the effective escalation ratio remains high regardless of release cadence, and gates G1 and G5 remain justified. The second objection is hindsight bias: once the outcome is known, every failure appears to reveal a missing requirement. The

coding rule of Section 3.3 and the Knight Capital boundary case mitigate but cannot eliminate this risk, which favours prospective tests. The third objection is that not every operating condition is knowable in advance, especially for systems facing adaptive adversaries whose tactics evolve (Okoruwa et al., 2024; Adebayo et al., 2023). The framework concedes this by placing G5 and operational monitoring at the end of the chain: intrusion detection and prevention, integrated network and security operations, anomaly detection in time series, digital twin fault detection and interpretable early failure prediction are the runtime counterparts of requirements containment (Dosunmu et al., 2020; Ladapo et al., 2024; Atakpa et al., 2023; Shittu et al., 2023; Komi et al., 2021).

The fourth objection is bureaucracy. Gates can become rituals in which checklists are completed without scrutiny, and the evidence on review frequency versus coverage suggests that this risk is real (Obriki et al., 2019). The framework's safeguards are yield-based measurement and risk-scaled effort, but their effectiveness is itself an empirical question. The fifth objection is measurement gaming: once leakage and escape distance become targets, teams may reclassify defects to improve them. Quantitative evaluation of organisational investments shows how sensitive such measures are to model choices (Uchechi et al., 2022), and monitoring coverage in physical infrastructure shows that a metric is only as good as the inspection regime behind it (Dagodzo et al., 2020). Independent coding of origin, reported agreement statistics and separation of learning metrics from performance appraisal are the minimum protections. RDOC is therefore most applicable where consequences are irreversible, systems cross organisational boundaries and requirements are documented, and least applicable to small, reversible, single-team products.

8 Limitations and Future Research

The review has five main limitations. First, it is integrative rather than exhaustive; despite a structured protocol, relevant studies may have been missed, and the absence of numerical screening counts reflects the conceptual aim rather than a completed enumeration (Page, 2021). Second, the public cases are biased toward catastrophic, well-investigated failures, most of them from aerospace, defence, medicine and large public projects, and most several decades old, because these are the cases for which independent investigation reports are available; routine enterprise failures, which are far more numerous, are rarely investigated in public. Third, the classification of cases relies on judgement, and although the coding rule is conservative, another analyst might assign different contributing classes. Fourth, the parameters of the cost model are illustrative and do not constitute estimates. Fifth, analogies from safety, finance, contracts and quality management are explanatory parallels, not evidence that the mechanisms operate identically in software. Evaluations of capability-building interventions in other fields have found modest and heterogeneous effects (Miller et al., 2015), which counsels caution against assuming that training or process change will transfer as expected.

A research agenda follows from these limitations and from the propositions. The first priority is a multi-organisation dataset of escaped defects coded for origin phase, origin class, detection phase and repair and consequence cost, which would allow r , PCE and D to be estimated and $P1$, $P5$ and $P6$ to be tested. The second is a set of controlled inspection experiments with seeded omissions and ambiguities to test $P4$ and to estimate the reliability of the taxonomy through inter-coder agreement. The third is longitudinal study of change-time traceability to test $P3$, including the usability of automated trace recovery in real change workflows (Borg et al., 2014). The fourth concerns systems whose behaviour is partly learned

from data. Real-world evidence and predictive analytics in health systems show that requirements on data provenance and representativeness now carry much of the risk that functional requirements once carried (Ilodigwe et al., 2024a), and threat modelling for adversarial machine learning shows that misuse requirements must cover the training pipeline as well as the deployed model (Adebayo et al., 2022). The fifth concerns small enterprises and public-facing services in developing economies, where digital financial inclusion is expanding rapidly and formal requirements practice is often thin (World Bank, 2014; Eyetsemitan et al., 2023b).

9 Conclusion

Requirements-related software failures are not a single phenomenon. They arise through five distinguishable mechanisms, elicitation failure, ambiguity, inconsistency, traceability loss and validation failure, each injected at a characteristic point in the lifecycle and each escaping for a characteristic reason. What unites them is that verification cannot see them, because the specification against which verification checks is itself the carrier of the defect. That is why their cost rises with the distance they travel, and why that cost rises fastest where consequences are irreversible and where defects cross organisational boundaries.

The public failures examined here confirm that the most damaging requirements defects are unexamined operating assumptions, components reused without revalidation, interface contracts left unenforced and design changes whose impact was never traced. Prevention therefore depends less on writing better sentences than on bringing independent references to bear: the stakeholders and adversaries who were missing, the operating envelope that was assumed, the interface partner on the other side of the boundary and the real conditions of use. The RDOC framework organises those references into five gates, measures containment through phase containment, requirements leakage and escape distance, and closes the loop by tracing every escaped defect back to the gate that missed it. Its propositions are stated so that they can be refuted, and testing them is the next step.

Funding: This research received no specific grant from any funding agency in the public, commercial or not-for-profit sectors.

Competing interests

The authors declare no competing interests.

Data availability

No new data were generated or analysed in this study; all sources are cited in the reference list.

References

1. Adebayo, A., Adegbite, M. P., & Ahmed, M. O. (2022). Adversarial machine learning in critical infrastructure: A conceptual framework for threat modeling AI enabled OT systems. *World Journal of Innovation and Modern Technology*, 6(1), 184-234. <https://doi.org/10.56201/wjimt.v6.no1.2022.pg184.234>

2. Adebayo, A., Adegbite, M. P., & Ahmed, M. O. (2023). AI augmented threat detection in industrial control systems: A systematic review of machine learning approaches for ICS anomaly detection. *International Journal of Engineering and Modern Technology*, 9(3), 287-340. <https://doi.org/10.56201/ijcsmt.v9.no3.2023.pg287.340>
3. Adegbite, M. P., Adebayo, A., & Ahmed, M. O. (2022). A security architecture model for IT and OT convergence in regulated energy networks: Design principles and governance alignment. *International Journal of Computer Science and Mathematical Theory*, 8(2), 81-132. <https://doi.org/10.56201/ijcsmt.v8.no2.2022.pg81.132>
4. Adegbite, M. P., Adebayo, A., & Ahmed, M. O. (2020). Securing operational technology networks in electric utilities: A systematic review of NERC CIP compliance and architectural threat mitigation. *International Journal of Engineering and Modern Technology*, 6(3), 103-146. <https://doi.org/10.56201/ijemt.vol.6.no3.2020.pg103.146>
5. Adelanwa, A., Basnet, A., & Anene, U. N. (2023b). Predictive analytics models for financial risk detection and fraud prevention in public systems. *International Journal of Advanced Multidisciplinary Research and Studies*, 3(6). <https://doi.org/10.62225/2583049X.2023.3.6.5969>
6. Adesuyi, M. O., Akomolafe, O., Olaogun, B. O., Ndukwe, V. U., & Sakyi, J. K. (2024). AI-driven risk scoring model for global cross-border trade payment transactions. *International Journal of Advanced Multidisciplinary Research and Studies*, 4(1), 1569-1581. <https://doi.org/10.62225/2583049X.2024.4.1.5281>
7. Adesuyi, M. O., Kalu, A., & Walawalkar, G. (2023). Data-led cost governance in technology-intensive enterprises. *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, 9(3), 877-896. <https://doi.org/10.32628/IJSRCSEIT>
8. Ahmed, M. O., Adegbite, M. P., & Adebayo, A. (2021). Zero trust architecture for operational technology in North American critical infrastructure: A framework for implementation and resilience optimization. *International Journal of Engineering and Modern Technology*, 7(1), 66-113. <https://doi.org/10.56201/ijemt.vol.7.no1.2021.pg66.113>
9. Akanbi, O., & Sunday, E. A. (2024). An integrated path-planning and slotting optimization model for AMR-enabled high-density warehousing. *International Journal of Advanced Multidisciplinary Research and Studies*, 4(6), 3226-3243. <https://doi.org/10.62225/2583049X.2024.4.6.6181>
10. Akhigbe, R., Falemi, A., & Akin-Oluyomi, O. T. (2024). An advanced framework for technology adoption that supports employee readiness in modern organizations. *International Journal of Advanced Multidisciplinary Research and Studies*, 4(1), 1637-1659. <https://doi.org/10.62225/2583049X.2024.4.1.5344>
11. Akin-Oluyomi, O. T., & Akhigbe, R. (2023b). A conceptual supply chain leadership model for aligning logistics decisions with enterprise strategy. *International Journal of Advanced Multidisciplinary Research and Studies*, 3(1), 1725-1744. <https://doi.org/10.62225/2583049X.2023.3.1.5340>
12. Alieme, J. I., Sule, A., & Obukadata, V. O. (2024). From ambiguity to AI-ready requirements: A business analysis framework for structuring business problems for intelligent process automation. *International Journal of Multidisciplinary Research and Growth Evaluation*, 5(6), 2088-2117. <https://doi.org/10.54660/IJMRGE.2024.5.6.2088-2117>

13. Aliliele, C., Mbonu, I. S., & Iwuanyanwu, U. (2023a). A conceptual framework for continuous cloud misconfiguration monitoring and enterprise risk mitigation strategies. *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, 9(10), 373-394. <https://doi.org/10.32628/CSEIT2361071>
14. Aliliele, C., Mbonu, I. S., & Iwuanyanwu, U. (2024a). A conceptual framework for enterprise data sensitivity classification and regulatory traceability mechanisms. *International Journal of Advanced Multidisciplinary Research and Studies*, 4(6), 3103-3124. <https://doi.org/10.62225/2583049X.2024.4.6.5991>
15. Aliliele, C., Mbonu, I. S., & Iwuanyanwu, U. (2023b). A review of API governance and risk prioritization frameworks in modern financial institutions. *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, 9(10), 395-433. <https://doi.org/10.32628/CSEIT2361072>
16. Alm, J. (2019). What motivates tax compliance? *Journal of Economic Surveys*, 33(2), 353-388. <https://doi.org/10.1111/joes.12272>
17. Amayo, E. B., Owulade, O. A., & Isi, L. R. (2023c). The role of diversity and inclusion in enhancing project team performance and delivery in multinational environments. *Journal of Frontiers in Multidisciplinary Research*, 4(1), 48-58. <https://doi.org/10.54660/IJFMR.2023.4.1.48-58>
18. Aminu-Ibrahim, A. Y., Ogbete, J. C., & Ambali, K. B. (2024). Governance and accountability models for public private partnerships in healthcare infrastructure development. *International Journal of Advanced Multidisciplinary Research and Studies*, 4(6), 2943-2960. <https://doi.org/10.62225/2583049X.2024.4.6.5699>
19. Ammann, P., & Offutt, J. (2016). *Introduction to Software Testing* (2nd ed.). Cambridge University Press.
20. Aneke, O. B., Adesemoye, A. C., & Sydney, E. (2022a). Corrective and preventive action effectiveness in pharmaceutical quality management systems: A systematic review of root-cause methodologies. *International Journal of Health and Pharmaceutical Research*, 7(2), 78-111.
21. Anichukwueze, C. C., Ominyi, M., Uzougbo, N. S., & Jones, B. C. (2022). Cross-functional legal education as risk mitigation: Empirical analysis of contract error reduction in engineering teams. *International Journal of Engineering and Modern Technology*, 8(5), 142-180. <https://doi.org/10.56201/ijemt.v8.no5.2022.pg142.180>
22. Anioke, S. C., & Atima, M. E. (2023a). Public health governance models using process optimization and performance metrics for regulatory oversight. *International Journal of Advanced Multidisciplinary Research and Studies*, 3(6), 2534-2548. <https://doi.org/10.62225/2583049X.2023.3.6.5491>
23. Anyanwu, K. C. E. (2022). Bridging enterprise IT operations and digital entrepreneurship: A systematic review of skill transfer mechanisms and pathways. *World Journal of Entrepreneurial Development Studies*, 7(1), 114-134. <https://doi.org/10.56201/wjeds.v7.no1.2022.pg114.134>
24. Arumosoye, O. M., & Obriki, O. D. (2023). Conceptual model for emergency response readiness and capability in energy and process facilities. *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, 9(3), 897-917. <https://doi.org/10.32628/CSEIT25112791>

25. Arumosoye, O. M., & Obriki, O. D. (2024). Conceptual model of safety leadership influence in large temporary project organizations. *International Journal of Advanced Multidisciplinary Research and Studies*, 4(6), 3034-3047. <https://doi.org/10.62225/2583049X.2024.4.6.5895>
26. Arumosoye, O. M., & Obriki, O. D. (2022). Conceptual risk pathway model for lifting and rigging operations in heavy industrial construction. *Gyanshauryam, International Scientific Refereed Research Journal*, 5(5), 346-369. <https://doi.org/10.32628/GISRRJ2256237>
27. Arumosoye, O. M., & Obriki, O. D. (2018). Development of an integrated heat stress risk conceptual model for industrial operations in extreme environments. *Iconic Research and Engineering Journals*, 1(12), 141-160. <https://doi.org/10.64388/IREV1I12-1714415>
28. Arumosoye, O. M., & Obriki, O. D. (2021). Organizational learning-based conceptual maturity model for continuous safety performance improvement. *International Journal of Multidisciplinary Research and Growth Evaluation*, 2(1), 970-980. <https://doi.org/10.54660/IJMRGE.2021.2.1.970-980>
29. Arumosoye, O. M., & Obriki, O. D. (2019). Systematic review of near-miss and hazard observation data utilization in industrial safety management. *Iconic Research and Engineering Journals*, 3(2), 981-999. <https://doi.org/10.64388/IREV3I2-1714417>
30. Asiedu, C. S., & Asiedu, A. A. (2024b). Reconceptualizing quality assurance in pharmaceutical manufacturing as a determinant of supply reliability. *International Journal of Health and Pharmaceutical Research*, 9(5), 148-173. <https://doi.org/10.56201/ijhpr.v9.no5.2024.pg148.173>
31. Atakpa, M. I., & Abetoh, N. F. (2022). A systematic review of machine learning advances in financial fraud detection for banking systems. *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, 8(2), 771-801. <https://doi.org/10.32628/CSEIT23906220>
32. Atakpa, M. I., & Fobellah, A. N. (2023). Anomaly detection in financial time-series data: A conceptual model for healthcare and banking applications. *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, 9(2), 967-997. <https://doi.org/10.32628/CSEIT2342441>
33. Atta, S., Adjaklo, E., Asomani, E., & Adenuga, O. M. (2022). Advances in welding inspection standards and quality assurance practices in large-scale power generation projects. *International Journal of Engineering and Modern Technology*, 8(5), 101-124. <https://doi.org/10.56201/ijemt.v8.no5.2022.pg101.124>
34. Atta, S., Tofah, P., & Adenuga, O. M. (2021). Advances in non-destructive testing methods for weld integrity evaluation in high-capacity power infrastructure. *International Journal of Engineering and Modern Technology*, 7(1), 20-47. <https://doi.org/10.56201/ijemt.vol.7.no1.2021.pg20.47>
35. Badmus, O., Jooda, D., & Anunagba, C. O. (2024). A privacy-by-design framework for role-based security architecture in Salesforce environments handling sensitive personal data. *International Journal of Advanced Multidisciplinary Research and Studies*, 4(6), 3244-3254. <https://doi.org/10.62225/2583049X.2024.4.6.6243>
36. Borah, S., Aliliele, K. C., Rakshit, S., & Vajjhala, N. R. (2022). Applications of artificial intelligence in software testing. In P. K. Mallick et al. (Eds.), *Cognitive informatics and soft computing (Lecture Notes in Networks and Systems, Vol. 375, pp. 727-736)*. Springer. https://doi.org/10.1007/978-981-16-8763-1_60

37. Borg, M., Runeson, P., & Ardö, A. (2014). Recovering from a decade: A systematic mapping of information retrieval approaches to software traceability. *Empirical Software Engineering*, 19, 1565-1616. <https://doi.org/10.1007/s10664-013-9255-y>
38. Catal, C., & Mishra, D. (2013). Test case prioritization: A systematic mapping study. *Software Quality Journal*, 21, 445-478. <https://doi.org/10.1007/s11219-012-9181-z>
39. Cleland-Huang, J., Gotel, O., & Zisman, A. (Eds.). (2012). *Software and Systems Traceability*. Springer. <https://doi.org/10.1007/978-1-4471-2239-5>
40. Dagodzo, D., & Ahiaeke Patrick, M. C. (2020). UAV-based pipeline and corridor monitoring: A review of current practices and emerging technologies. *Iconic Research and Engineering Journals*, 3(10), 574-597. <https://doi.org/10.64388/IREV3I10-1716084>
41. Daka, E., & Fraser, G. (2014). A survey on unit testing practices and problems. *IEEE International Symposium on Software Reliability Engineering Workshops*, 201-211. <https://doi.org/10.1109/ISSREW.2014.11>
42. De Lucia, A., Fasano, F., Oliveto, R., & Tortora, G. (2007). Recovering traceability links in software artifact management systems using information retrieval methods. *ACM Transactions on Software Engineering and Methodology*, 16(4), Article 13. <https://doi.org/10.1145/1276933.1276934>
43. Dogbatsey, E. A., Ebhojie, O., & Oyeleye, A. O. (2019). Reconciliation control and financial workflow redesign in emerging market organizations: A conceptual framework. Zenodo. <https://doi.org/10.5281/zenodo.20447103> \[Originally numbered 3(5); journal name not captured in source.\]
44. Dosunmu, A. A., & Ogundele, P. O. (2020). Intrusion detection and prevention models for enhancing organizational cyber defense effectiveness. *Iconic Research and Engineering Journals*, 4(6), 310-324. <https://doi.org/10.64388/IREV4I6-1713226>
45. Dybå, T., & Dingsøyr, T. (2008). Empirical studies of agile software development: A systematic review. *Information and Software Technology*, 50(9-10), 833-859. <https://doi.org/10.1016/j.infsof.2008.01.006>
46. Easterbrook, S., Singer, J., Storey, M., & Damian, D. (2008). Selecting empirical methods for software engineering research. In *Guide to Advanced Empirical Software Engineering* (pp. 285-311). Springer.
47. Edivri, J., & Oteri, O. (2024). A conceptual KPI-driven decision and optimization framework for IT service delivery, portfolio performance, and adoption. *Gyanshauryam, International Scientific Refereed Research Journal*, 7(1), 167-187. <https://doi.org/10.32628/GISRRJ246643>
48. Eyetsemitan, R. A., Ambali, K. B., Oyeleye, A. O., & Fadayomi, O. (2023b). User acceptance testing in small business technology deployment: A structured validation framework for lean operational environments. *International Journal of Multidisciplinary Research and Growth Evaluation*, 4(6), 1512-1531. <https://doi.org/10.54660/IJMRGE.2023.4.6.1512-1531>
49. Eze, F. I., Akinleye, O. K., & Anene, U. N. (2024b). Development of a risk-based accelerated approval decision model balancing speed, safety, and evidence quality in pharmaceutical regulatory science. *Research Journal of Pure Science and Technology*, 7(6), 120-161. <https://doi.org/10.56201/rjpst.v7.no6.2024.pg120.161>
50. Eze, F. I., Anene, U. N., & Akinleye, O. K. (2022b). Design of an integrated regulatory readiness maturity model for pharmaceutical companies seeking global market entry. *International Journal of*

- Health and Pharmaceutical Research, 7(1), 77-112.
<https://doi.org/10.56201/ijhpr.v7.no1.2022.pg77.112>
51. Fadayomi, O., Abolaji, T. O., Edivri, J., Ogbole, J. I., Okoruwa, P. O., & Akeju, B. (2019). Risk-based cybersecurity assurance and data availability: Limitations, advances and future research opportunities. *Iconic Research and Engineering Journals*, 2(12), 602-617.
<https://doi.org/10.64388/IREV2I12-1713779>
52. Fadayomi, O., Abolaji, T. O., Edivri, J., Ogbole, J. I., Okoruwa, P. O., & Akeju, B. (2019). Risk-based cybersecurity assurance and data availability limitations, advances and future research opportunities. *Iconic Research and Engineering Journals*, 2(12), 602-617.
<https://doi.org/10.64388/IREV2I12-1713779>
53. Fadayomi, O., Abolaji, T. O., Edivri, J., Ogbole, J. I., Okoruwa, P. O., & Akeju, B. (2019). Risk-based cybersecurity assurance and data availability: Limitations, advances and future research opportunities. *IRE Journals*, 2(12). <https://doi.org/10.64388/IREV2I12-1713779>
54. Falemi, A., Akhigbe, R., & Akin-Oluyomi, O. T. (2024). A conceptual enterprise risk management model that integrates HR strategy with operational analytics. *International Journal of Multidisciplinary Research and Growth Evaluation*, 5(6), 1730-1753.
<https://doi.org/10.54660/.ijmrge.2024.5.6.1730-1753>
55. Filani, O. M., Nnabueze, S. B., Sakyi, J. K., & Okojie, J. S. (2023). Scenario-based financial modelling for enhancing strategic decision-making and organizational long-term planning. *Journal of Frontiers in Multidisciplinary Research*, 4(2), 251-265.
<https://doi.org/10.54660/.JFMR.2023.4.2.251-265>
56. Filani, O. M., Nnabueze, S. B., Sakyi, J. K., & Okojie, J. S. (2023). Scenario-based financial modelling for enhancing strategic decision-making and organizational long-term planning. *Journal of Frontiers in Multidisciplinary Research*, 4(2), 251-265. <https://doi.org/10.54660/.JFMR.2023.4.2.251-265>
57. Gideon, E. N., Adaramola, T. S., & Fadero, S. (2019). Reliability engineering and failure analysis in complex engineering systems. *Iconic Research and Engineering Journals*, 2(11), 703-727.
<https://doi.org/10.64388/IREV2I11-1722490>
58. Gotel, O. C. Z., & Finkelstein, A. C. W. (1994). An analysis of the requirements traceability problem. *IEEE International Conference on Requirements Engineering*, 94-101.
<https://doi.org/10.1109/ICRE.1994.292398>
59. Hevner, A. R., March, S. T., Park, J., & Ram, S. (2004). Design science in information systems research. *MIS Quarterly*, 28(1), 75-105. <https://doi.org/10.2307/25148625>
60. IEEE Computer Society. (2014). Guide to the Software Engineering Body of Knowledge Version 3.0. IEEE.
61. Ilodigwe, L., & Adesemoye, A. C. (2024a). Advances in real-world evidence, predictive analytics, and data-driven decision making for improving health system performance and patient outcomes. *International Journal of Health and Pharmaceutical Research*, 9(5), 174-197.
<https://doi.org/10.56201/ijhpr.v9.no5.2024.pg174.197>
62. Ilodigwe, L., & Adesemoye, A. C. (2024b). Beyond technology: A strategic framework for building sustainable and resilient health systems through organizational strategy, operational excellence, and

- workforce capability. *International Journal of Medical Evaluation and Physical Report*, 8(6), 299-319. <https://doi.org/10.56201/ijmepr.v8.no6.2024.pg299.319>
63. ISO/IEC/IEEE. (2018). ISO/IEC/IEEE 29119-2 Software and systems engineering: Software testing processes. International Organization for Standardization.
64. ISO/IEC/IEEE. (2018). ISO/IEC/IEEE 29148 Systems and software engineering: Life cycle processes and requirements engineering. International Organization for Standardization.
65. Jimoh, H. O., & Ahmed, M. O. (2024). Analyzing Network Time Protocol (NTP) based amplification DDoS attack and its mitigation techniques. *Journal of Digital Innovations and Contemporary Research in Science, Engineering and Technology*, 12(2), 17-24. <https://doi.org/10.22624/AIMS/DIGITAL/V11N2P2x>
66. Jimoh, H. O., Ahmed, M. O., & Fagbade, M. O. (2023). The role of frameworks in cybersecurity governance. *Journal of Behavioural Informatics, Digital Humanities and Development Research*, 9(4), 7-16. <https://doi.org/10.22624/AIMS/BHI/V9N4P2>
67. Jones, B. C., & Ominyi, M. (2024b). A conceptual framework for harmonizing digital trade regulations in cross-border e-commerce between the United States and Nigeria. *International Journal of Economics and Financial Management*, 9(9), 337-364. <https://doi.org/10.56201/ijefm.v9.no9.2024.pg337.364>
68. Jones, B. C., & Ominyi, M. (2021). Developing a conceptual framework linking West African agricultural value chains to household economic stability. *International Journal of Economics and Financial Management*, 6(2), 89-114. <https://doi.org/10.56201/ijefm.v6.no2.2021.pg89.114>
69. Joseph, O. B., & Nwankwo, C. U. (2024). Professional development for STEM educators: Enhancing teaching effectiveness through continuous learning. *International Journal of Applied Research in Social Sciences*, 6(8), 1558-1574. <https://doi.org/10.51594/ijarss.v6i8.1370>
70. Kaner, C., Falk, J., & Nguyen, H. Q. (1999). *Testing Computer Software* (2nd ed.). Wiley.
71. Kirchler, E. (2007). *The Economic Psychology of Tax Behaviour*. Cambridge University Press.
72. Kitchenham, B., Brereton, O. P., Budgen, D., Turner, M., Bailey, J., & Linkman, S. (2009). Systematic literature reviews in software engineering: A systematic literature review. *Information and Software Technology*, 51(1), 7-15. <https://doi.org/10.1016/j.infsof.2008.09.009>
73. Komi, N. M., & Adamolekun, A. (2021). Interpretable machine learning for early failure prediction in distributed renewable energy assets. *International Journal of Multidisciplinary Research and Growth Evaluation*, 2(6), 1015-1038. <https://doi.org/10.54660/IJMRGE.2021.2.6.1015-1038>
74. Komi, N. M. (2023). When accurate forecasts are not fair forecasts: Deep learning load prediction and equity across districts. *Shodhshauryam, International Scientific Refereed Research Journal*, 6(6), 476-546. <https://doi.org/10.32628/SHISRRJ247102>
75. Ladapo, O. O., Dosunmu, A. A., Jooda, D., & Abolaji, T. O. (2024). Integrated network and security operation center: A systematic analysis. *International Journal of Multidisciplinary Futuristic Development*, 5(1), 65-80. <https://doi.org/10.54660/IJMFD.2024.5.1.65-80>
76. Ladapo, O. O., Jooda, D., Dosunmu, A. A., & Abolaji, T. O. (2023b). Systematic literature review on security access control policies and techniques based on privacy requirements in a BYOD environment. *International Journal of Advanced Multidisciplinary Research and Studies*, 3(6), 2890-2904. <https://doi.org/10.62225/2583049X.2023.3.6.6206>

77. Liadi, K. O. (2024c). Designing a cross-border security cooperation model: Nigeria's foreign policy response to regional terrorism. *Gyanshauryam, International Scientific Refereed Research Journal*, 7(3), 311-338. <https://doi.org/10.32628/GISRRJ247324>
78. Li, Z., Harman, M., & Hierons, R. M. (2007). Search algorithms for regression test case prioritization. *IEEE Transactions on Software Engineering*, 33(4), 225-237. <https://doi.org/10.1109/TSE.2007.38>
79. Mairiza, D., Zowghi, D., & Nurmuliani, N. (2010). An investigation into the notion of non-functional requirements. *ACM Symposium on Applied Computing*, 311-317. <https://doi.org/10.1145/1774088.1774153>
80. Mbonu, I. S., Aliliele, C., Iwuanyanwu, U., & Oluoha, O. M. (2018). A conceptual framework for legal and ethical risk modeling in enterprise data protection governance systems. *Iconic Research and Engineering Journals*, 2(2), 207-226. <https://doi.org/10.64388/IREV2I2-1714911>
81. Mbonu, I. S., Iwuanyanwu, U., Aliliele, C., & Uzoka, E. (2022a). A review of data protection impact assessment models in multi cloud financial infrastructure systems. *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, 8(1), 589-623. <https://doi.org/10.32628/CSEIT25442>
82. Mbonu, I. S., Iwuanyanwu, U., Aliliele, C., & Uzoka, E. (2021). A review of VoIP forensic analytics models for financial fraud detection and regulatory compliance monitoring. *International Journal of Multidisciplinary Research and Growth Evaluation*, 2(6), 711-730. <https://doi.org/10.54660/IJMRGE.2021.2.6.711-730>
83. Mbonu, I. S., Iwuanyanwu, U., Uzoka, E., & Oluoha, O. M. (2019). Advances in enterprise log analytics and automated incident response architectures using Python and SIEM platforms. *Iconic Research and Engineering Journals*, 3(2), 1000-1019. <https://doi.org/10.64388/IREV3I2-1714915>
84. Miller, M., Reichelstein, J., Salas, C., & Zia, B. (2015). Can you help someone become financially capable? *World Bank Research Observer*, 30(2), 220-246. <https://doi.org/10.1093/wbro/lkv009>
85. Myers, G. J., Sandler, C., & Badgett, T. (2011). *The Art of Software Testing* (3rd ed.). Wiley.
86. Nwakamma, S., Ojukwu, J., & Oyesiji, S. O. (2024a). LLM-as-a-judge for automated model governance: A review of methods, biases, and release-gating practices. *World Journal of Innovation and Modern Technology*, 8(6), 185-216. <https://doi.org/10.56201/wjimt.v8.no6.2024.pg185.216>
87. Nwakamma, S., Ojukwu, J., & Oyesiji, S. O. (2024b). Securing agentic AI enterprise workflows against prompt injection, tool poisoning, memory manipulation, and excessive agency threats. *World Journal of Innovation and Modern Technology*, 8(6), 217-248. <https://doi.org/10.56201/wjimt.v8.no6.2024.pg217.248>
88. Obogo, S. F., Nwafor, M. I., & Ozobu, C. O. (2023). Conceptual leadership model for safety culture development in construction and engineering projects. *International Journal of Scientific Research in Civil Engineering*, 7(6), 121-153. <https://doi.org/10.32628/IJSRCE237554>
89. Obogo, S. F., Obriki, O. D., & Arumosoye, O. M. (2021). Conceptual model for incident prevention in industrial maintenance engineering environments. *International Journal of Multidisciplinary Research and Growth Evaluation*, 2(6), 847-858. <https://doi.org/10.54660/IJMRGE.2021.2.6.847-858>
90. Obogo, S. F., Ozobu, C. O., & Uduokhai, D. O. (2019a). Advances in leadership driven safety culture transformation in large construction workforces. *Iconic Research and Engineering Journals*, 3(5), 507-523. <https://doi.org/10.64388/IREV3I5-1715497>

91. Obogo, S. F., Ozobu, C. O., & Uduokhai, D. O. (2019b). Development of a construction safety risk governance model for multi contractor high rise projects. *Iconic Research and Engineering Journals*, 2(9), 502-522. <https://doi.org/10.64388/IREV2I9-1715495>
92. Obogo, S. F., Uduokhai, D. O., & Ozobu, C. O. (2019). Systematic review of hazard identification and risk control practices in urban construction projects. *Iconic Research and Engineering Journals*, 2(12), 666-681. <https://doi.org/10.64388/IREV2I12-1715496>
93. Obriki, O. D., & Arumosoye, O. M. (2019). A conceptual framework linking management safety walkthrough frequency and coverage to safety culture outcomes in mega projects. *Iconic Research and Engineering Journals*, 2(8), 355-374. <https://doi.org/10.64388/IREV2I8-1714416>
94. Obriki, O. D., & Arumosoye, O. M. (2024b). Systematic review of incident investigation approaches and prevention-oriented learning in industrial operations. *Shodhshauryam, International Scientific Refereed Research Journal*, 7(4), 239-263. <https://doi.org/10.32628/SHISRRJ247163>
95. Obriki, O. D., & Arumosoye, O. M. (2020). Conceptual framework for human error causation in high-risk construction and industrial activities. *International Journal of Multidisciplinary Research and Growth Evaluation*, 1(5), 715-727. <https://doi.org/10.54660/IJMRGE.2020.1.5.715-727>
96. Obriki, O. D., & Arumosoye, O. M. (2018). Conceptual modeling of data-driven occupational safety risk control in large-scale energy infrastructure projects. *Iconic Research and Engineering Journals*, 1(7), 169-189. <https://doi.org/10.64388/IREV1I7-1714414>
97. Ogbete, J. C., Aminu-Ibrahim, A. Y., & Ambali, K. B. (2022). Design standards and operational planning frameworks for scalable blood collection networks. *Gyanshauryam, International Scientific Refereed Research Journal*, 5(6), 267-300. <https://doi.org/10.32628/GISRRJ225635>
98. Ogbete, J. C., Aminu-Ibrahim, A. Y., & Ambali, K. B. (2023). Lifecycle performance evaluation of purpose built diagnostic laboratories supporting long term healthcare delivery. *International Journal of Advanced Multidisciplinary Research and Studies*, 3(6), 2605-2621. <https://doi.org/10.62225/2583049X.2023.3.6.5698>
99. Ogbete, J. C., Aminu-Ibrahim, A. Y., & Ambali, K. B. (2018). Optimizing laboratory spatial planning strategies to improve diagnostic accuracy, safety, and clinical throughput. *Iconic Research and Engineering Journals*, 2(1), 87-113. <https://doi.org/10.64388/IREV1I7-1713587>
100. Okonkwo, C. S., Ogunwale, O., Mayo, W., & Okeke, O. T. (2021). Framework for regulatory-compliant procurement in high-risk energy environments. *International Journal of Multidisciplinary Research and Growth Evaluation*, 2(6), 595-605. <https://doi.org/10.54660/IJMRGE.2021.2.6.595-605>
101. Okoruwa, P. O., Fadayomi, O., Abolaji, T. O., Edivri, J., Ogbole, J. I., & Akeju, B. (2024). Enterprise cybersecurity trends and threat evolution, advances and emerging research directions. *Global Multidisciplinary Perspectives Journal*, 1(6), 194-204. <https://doi.org/10.54660/GMPJ.2024.1.6.194-204>
102. Ominyi, M., Anichukwueze, C. C., & Uzougbo, N. S. (2024b). A systematic review of organizational readiness for the EU Artificial Intelligence Act in multinational enterprises. *World Journal of Innovation and Modern Technology*, 8(6), 185-203. <https://doi.org/10.56201/wjimt.v8.no6.2024.pg185.203>
103. Ominyi, M. (2021). Toward a conceptual framework for embedding privacy by design in corporate restructuring and transaction structuring. *Journal of Accounting and Financial Management*, 7(5), 152-173. <https://doi.org/10.56201/jafm.vol.7.no5.2021.p152.173>

104. Onche, V. O., Adegbite, M. P., & Ogbonna, C. S. (2023). BYOD adoption and information-security risk in university administrative offices: Evidence from Nigerian federal universities. *International Journal of Multidisciplinary Research and Growth Evaluation*, 4(6), 1549-1563. <https://doi.org/10.54660/IJMRGE.2023.4.6.1549-1563>
105. Page, M. J., et al. (2021). The PRISMA 2020 statement: An updated guideline for reporting systematic reviews. *BMJ*, 372, n71. <https://doi.org/10.1136/bmj.n71>
106. Ramesh, B., & Jarke, M. (2001). Toward reference models for requirements traceability. *IEEE Transactions on Software Engineering*, 27(1), 58-93. <https://doi.org/10.1109/32.895989>
107. Runeson, P., & Höst, M. (2009). Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering*, 14, 131-164. <https://doi.org/10.1007/s10664-008-9102-8>
108. Sakyi, J. K., Filani, O. M., Nnabueze, S. B., Okojie, J. S., & Ogedengbe, A. O. (2022b). Developing KPI frameworks to enhance accountability and performance across large-scale commercial organizations. *Journal of Frontiers in Multidisciplinary Research*, 3(2), 81-93. <https://doi.org/10.54660/IJFMR.2022.3.2.81-93>
109. Sanni, J. O., Atima, M. E., & Attah, A. (2022). Systematic review of attribution modeling methods resolving bias in multi-touch journeys. *Shodhshauryam, International Scientific Refereed Research Journal*, 5(1), 304-334. <https://doi.org/10.32628/SHISRRJ247135>
110. Sanni, J. O., & Attah, A. (2023c). Market research frameworks addressing entry barriers within highly regulated industrial service sectors. *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, 10(2), 904-930. <https://doi.org/10.32628/CSEIT2342439>
111. Sanni, J. O., Iwuanyanwu, U. A., Essien, M. A., & Attah, A. (2023). Lifecycle-aware marketing automation using federated learning for secure cross-organizational data management. *Gyanshauryam, International Scientific Refereed Research Journal*, 6(6), 337-364. <https://doi.org/10.32628/GISRRJ236642>
112. Shittu, M. A., Shittu, H. A., Adeleke, O. J., & Adedokun, O. J. (2023). Digital twin modeling for real time monitoring and fault detection in smart substations. *International Journal of Industrial Engineering Research and Development*, 14(2), 25-44. https://doi.org/10.34218/IJIERD_14_02_003
113. Tawose, O. M., Oluwadele, J. F., Ekeocha, A. H., Odumboni, A. A., & Egbeyemi, O. S. (2023). Aflatoxin detection and quantification in poultry feeds available in selected areas in Ekiti State. *FUOYE Journal of Agriculture and Human Ecology*, 7(2). <https://doi.org/10.62923/fuojahe.v7i2.302>
114. Tonoyan, A., Dada, O., & Ayivi-Donkor, S. S. (2023). A conceptual framework for retail asset valuation: Integrating quantitative pricing and multi-unit financial performance prediction. *Journal of Accounting and Financial Management*, 9(12), 218-250. <https://doi.org/10.56201/jafm.v9.no12.2023.pg218.250>
115. Tonoyan, A., Dada, O., & Ayivi-Donkor, S. S. (2022b). Algorithmic process optimization and cost reduction: A review of simulation modeling and financial impact assessment. *Journal of Accounting and Financial Management*, 8(8), 139-169. <https://doi.org/10.56201/jafm.vol.8.no8.2022.p139.169>
116. Tonoyan, A., Dada, O., & Ayivi-Donkor, S. S. (2024b). Real-time KPI tracking systems: A review of automated performance monitoring and data-driven decision making. *World Journal of*

- Innovation and Modern Technology, 8(6), 247-281.
<https://doi.org/10.56201/wjimt.v8.no6.2024.pg247.281>
117. Uchechi, N. (2018). Mitigating enterprise financial exposure: Standardization of stamp duty policies and class action defense protocols. *Iconic Research and Engineering Journals*, 2(5).
<https://doi.org/10.64388/IREV2I5-1722788>
118. Uchechi, N., Ominyi, M., Anichukwueze, C. C., & Uzougbo, N. S. (2022). Advances in quantitative evaluation models for measuring corporate diversity and inclusion investments. *World Journal of Finance and Investment Research*, 6(2), 56-101.
<https://doi.org/10.56201/wjfir.v6.no2.2022.pg56.101>
119. Uchechi, N., Ominyi, M., & Jones, B. C. (2018). Procedural standardization and workflow optimization in high-volume dispute resolution clinics: A systematic review. *Iconic Research and Engineering Journals*, 2(6). <https://doi.org/10.64388/IREV2I6-1722789>
120. Uchechi, N., & Ominyi, M. (2019). Liability mitigation and dispute prevention in high-value energy service contracts. *Iconic Research and Engineering Journals*, 3(6).
<https://doi.org/10.64388/IREV3I6-1722790>
121. Wedraogo, L., & Sanni, J. O. (2024). Machine learning models addressing uncertainty in cross-channel campaign performance forecasting accuracy. *International Journal of Advanced Multidisciplinary Research and Studies*, 4(6), 2875-2890.
<https://doi.org/10.62225/2583049X.2024.4.6.5648>
122. Wieringa, R. J. (2014). *Design Science Methodology for Information Systems and Software Engineering*. Springer. <https://doi.org/10.1007/978-3-662-43839-8>
123. Winkler, S., & von Pilgrim, J. (2010). A survey of traceability in requirements engineering and model-driven development. *Software and Systems Modeling*, 9, 529-565.
<https://doi.org/10.1007/s10270-009-0145-0>
124. World Bank. (2014). *Global Financial Development Report 2014: Financial Inclusion*. World Bank. <https://doi.org/10.1596/978-0-8213-9985-9>
125. Yusuf, H. D., & Abdulrazaq, K. (2018). A conceptual framework for strengthening diagnostic quality systems in resource-limited clinical laboratories. *Iconic Research and Engineering Journals*, 1(11), 145-179. <https://doi.org/10.64388/IREV1I11-1723090>
126. Zhu, H., Hall, P. A. V., & May, J. H. R. (1997). Software unit test coverage and adequacy. *ACM Computing Surveys*, 29(4), 366-427. <https://doi.org/10.1145/267580.267590>