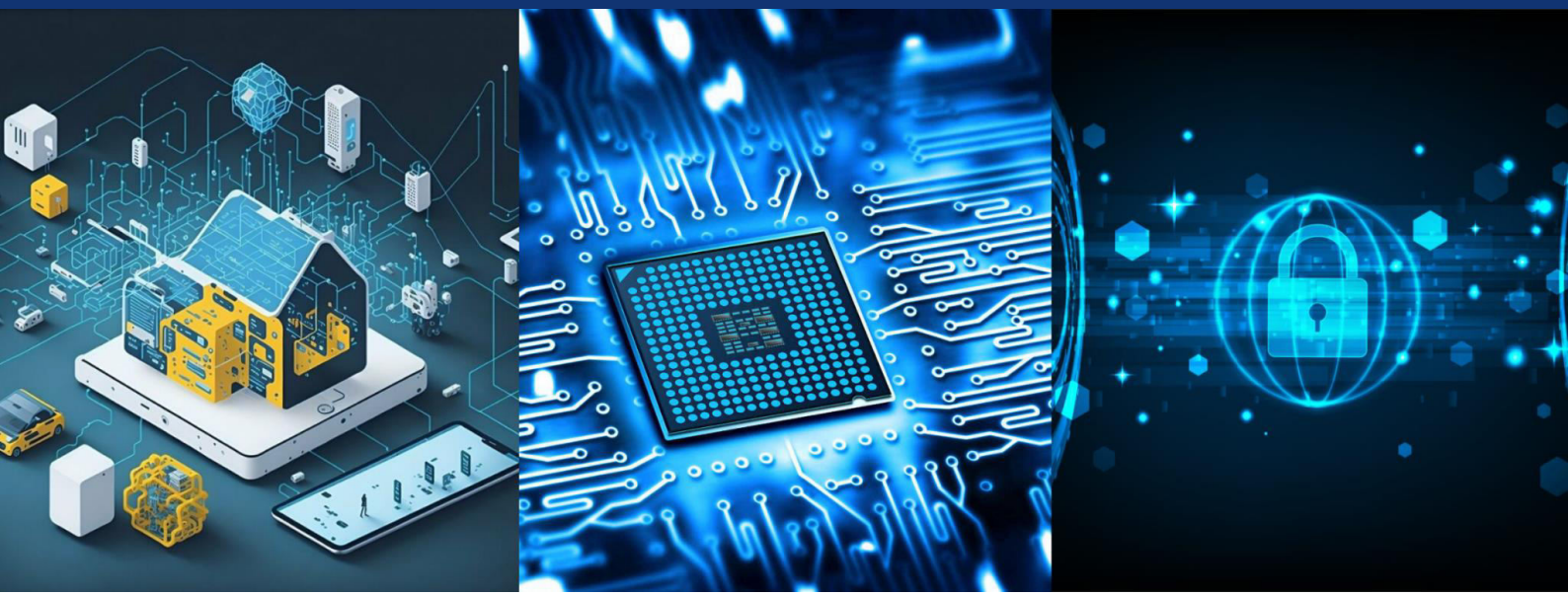
IJIRCCE

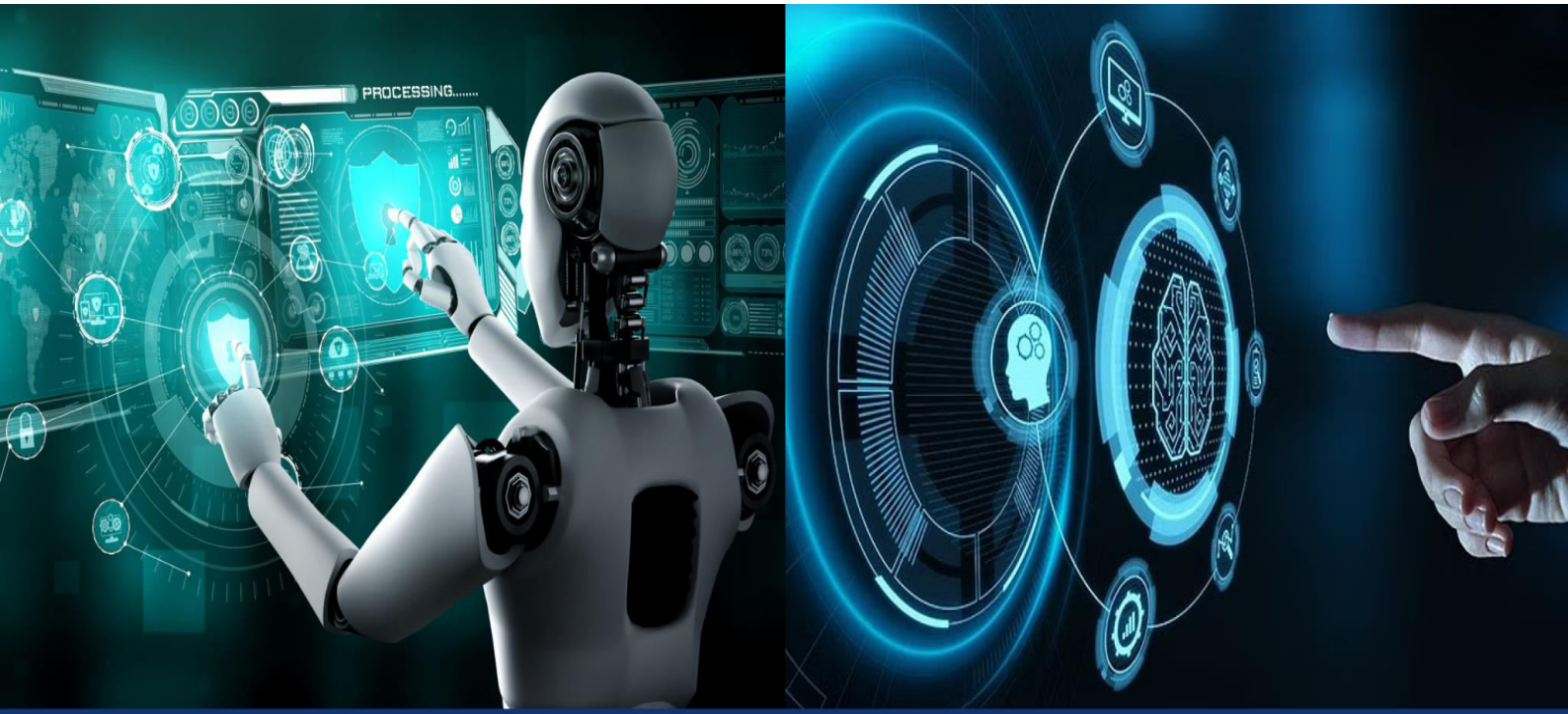
ISSN(O): 2320-9801

ISSN(P): 2320-9798



International Journal of Innovative Research in Computer and Communication Engineering

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)



Impact Factor: 8.771

Volume 13, Issue 5, May 2025

© 2025 IJIRCCE | Volume 13, Issue 5, May 2025| DOI:10.15680/IJIRCCE.2025.1305305

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 8.771| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

Operationalizing DevSecOps in Enterprise ERP Landscapes: Continuous Integration, Automated Testing, and Security Gate Frameworks for Multi-Release Transport Management

Srinivas Kakarla

SAP Architect, USA

ABSTRACT: Enterprise resource planning (ERP) landscapes have historically promoted change through transport requests reviewed largely by manual process, a model that scales poorly once an organization runs multiple concurrent release trains. This article presents an alternative operating model - a continuous integration and delivery (CI/CD) pipeline with security gates embedded at defined checkpoints - and extends the analysis with an economic lens largely absent from prior treatments of this topic: an illustrative cost waterfall, a hidden-cost iceberg model, and a sensitivity (tornado) analysis of the value drivers behind a DevSecOps transport-gating investment. We further introduce a statistical process control view of gate performance over time, a bump-chart trend of shifting root causes, and a RACI governance tree clarifying accountability for gate decisions. All monetary figures presented are explicitly illustrative estimates, flagged throughout, intended to demonstrate analytical method rather than to report measured costs from any specific organization.

KEYWORDS: DevSecOps, Enterprise Resource Planning, CI/CD, Transport Management, Security Gates, Statistical Process Control, Cost Sensitivity Analysis, RACI Governance, Release Management, Continuous Delivery

I. INTRODUCTION: WHY TRANSPORT MANAGEMENT NEEDS DEVSECOPS

In most enterprise resource planning (ERP) landscapes, a "transport" is the unit of change: a packaged set of configuration or code objects moved between systems - typically development, quality assurance, staging, and production - via a scheduled, often manually reviewed process. This transport model has served ERP change management reliably for decades, but it was designed for a release cadence measured in months, not the multiple concurrent release trains that many enterprises now run to support faster-moving finance, logistics, and compliance requirements.

This article is a companion treatment to prior work on this subject, extending the architectural discussion with an economic lens: what does a DevSecOps transport-gating program actually cost, where do the hidden costs live, and which value drivers matter most to the business case. We also introduce statistical process control and root-cause trend analysis as ongoing measurement disciplines, rather than one-time adoption metrics.

A security gate that cannot demonstrate its own cost and value over time is a control that governance will eventually be asked to justify - better to have the answer ready than to reconstruct it under audit pressure.

The remainder of this article proceeds as follows. Section 2 frames the multi-release transport challenge. Section 3 presents a tool-ecosystem view of the pipeline. Section 4 traces transport flow using a Sankey-style visualization. Section 5 covers gate design and governance accountability. Section 6 is the economic core of the article: cost waterfall, hidden-cost iceberg, and sensitivity analysis. Section 7 introduces statistical process control for gate performance. Section 8 examines root-cause trends and calendar-level patterns. Section 9 offers implementation considerations. Section 10 addresses risks and open questions, and Section 11 concludes.

© 2025 IJIRCCE | Volume 13, Issue 5, May 2025| DOI:10.15680/IJIRCCE.2025.1305305

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 8.771| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

II. THE MULTI-RELEASE TRANSPORT PROBLEM IN ERP LANDSCAPES

Enterprises running more than one concurrent ERP release train - for example, a stabilization branch for the current production release alongside active development for the next two quarterly releases - face a transport management challenge that a single-release model does not: transports from different release trains can target the same landscape systems in overlapping windows, creating sequencing dependencies and a higher risk that a security or quality issue in one train's transport is masked by, or conflated with, activity from another.

2.1 Why Manual Review Does Not Scale

Volume. A landscape running multiple release trains can generate hundreds of transports per quarter, far exceeding what a manual basis-team review process can meaningfully scrutinize for security issues on a per-transport basis.
Consistency. Manual review quality varies by reviewer availability, familiarity with the specific module being changed, and time pressure near a release deadline.
Traceability. Without automated, logged gate results, demonstrating to an auditor that every production transport received consistent security review becomes a manual, evidence-gathering exercise rather than a query against a pipeline's own records.

2.2 What Makes ERP Transport Security Different

Several characteristics of ERP transport objects complicate directly reusing CI/CD security tooling built for cloud-native application code: mixed object types spanning code, configuration, and data dictionary changes; cross-transport dependencies that complicate gate design; and comparatively less mature vendor tooling for ERP-specific scanning relative to the broader application security tooling market.

Table.1. How Multi-Release Transport Management Changes the Security and Governance Burden.

Factor	Single-Release Model	Multi-Release Model
Concurrent transports in flight	Low - sequential by default	High - multiple trains overlap
Manual review feasibility	Feasible at moderate volume	Breaks down past a few dozen transports/month
Audit evidence burden	Manageable via change logs	Requires automated, queryable gate records
Cross-train conflict risk	Minimal	Material - sequencing and dependency conflicts

III. A TOOL-ECOSYSTEM VIEW OF THE TRANSPORT PIPELINE

Rather than presenting the pipeline purely as a left-to-right sequence, Figure 1 organizes the same information around a central four-stage spine - Source & Build, Test & Scan, Approval Gate, Deploy & Monitor - with each stage flanked by the tool categories that support it. This framing is useful when planning tool procurement or consolidation, since it groups tools by the pipeline stage they serve rather than by vendor or product category alone.

© 2025 IJIRCCE | Volume 13, Issue 5, May 2025| DOI:10.15680/IJIRCCE.2025.1305305

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 8.771| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

Figure 1. Tool-Ecosystem Mesh: Category Coverage Along the ERP Transport Pipeline Spine

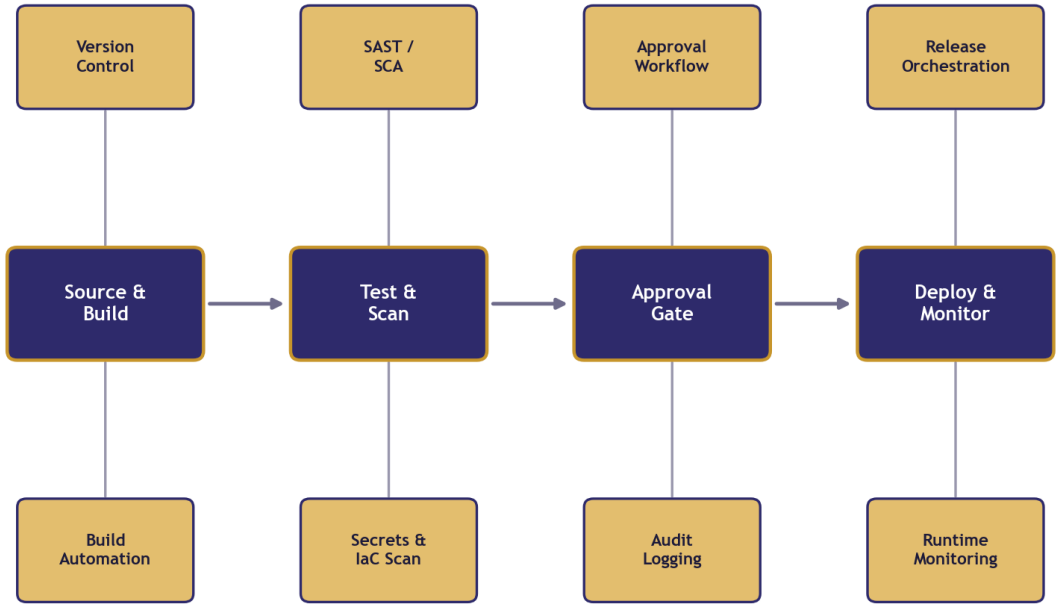


Figure.1. Tool-ecosystem mesh showing category coverage - version control, scanning, approval workflow, and monitoring tooling - along the four-stage pipeline spine.

3.1 Reading the Mesh

Each spine stage has exactly two flanking tool categories in this illustration, reflecting a typical minimum viable tool set; larger organizations often run additional categories per stage (for example, multiple SAST tools covering different languages). The mesh framing makes a gap immediately visible: if a stage has no flanking category populated, that stage is running without the tooling support its neighbors have, a useful lens for identifying underinvested pipeline stages during a tooling audit.

IV. TRANSPORT FLOW: A SANKEY VIEW OF QUARTERLY OUTCOMES

Figure 2 presents an illustrative quarterly transport flow using a Sankey-style branching visualization, tracing how an initial pool of requested transports narrows and splits across three stages: intake, automated gates, and final disposition. Unlike the funnel visualization used in prior treatments of this topic, the Sankey framing makes explicit that transports which pass automated gates still branch into two distinct final outcomes - approved for production, or held for manual review - rather than assuming every passing transport reaches production automatically.

© 2025 IJIRCCE | Volume 13, Issue 5, May 2025| DOI:10.15680/IJIRCCE.2025.1305305

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 8.771| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

Figure 2. Illustrative Quarterly Transport Flow with Branching Outcomes (Sankey-Style)

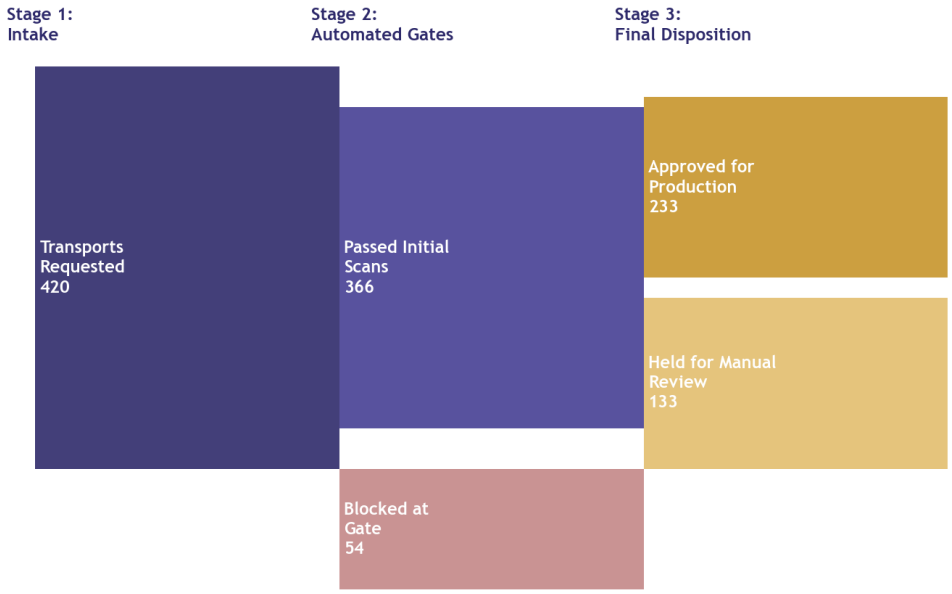


Figure.2. Illustrative quarterly transport flow with branching outcomes: transports split at each stage into passed/blocked and approved/held categories.

This distinction matters for capacity planning: a pipeline that routes a large share of gate-passing transports to manual review has not actually reduced its manual review burden proportionally to its automation investment - it has simply moved the review trigger later in the pipeline. Tracking the held-for-review share over time (Section 7) is one way to verify whether automation investment is translating into genuine review-burden reduction.

V. SECURITY GATE DESIGN AND GOVERNANCE ACCOUNTABILITY

Beyond the gate mechanics addressed in prior treatments of this topic, the governance question - who is accountable when a gate decision is disputed, or when a gate is bypassed via an exception process - deserves explicit treatment. Figure 7 presents a RACI (Responsible, Accountable, Consulted, Informed) governance tree for transport gate decisions.

© 2025 IJIRCCE | Volume 13, Issue 5, May 2025| DOI:10.15680/IJIRCCE.2025.1305305

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 8.771| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

Figure 7. RACI Governance Tree for DevSecOps Transport Gate Decisions

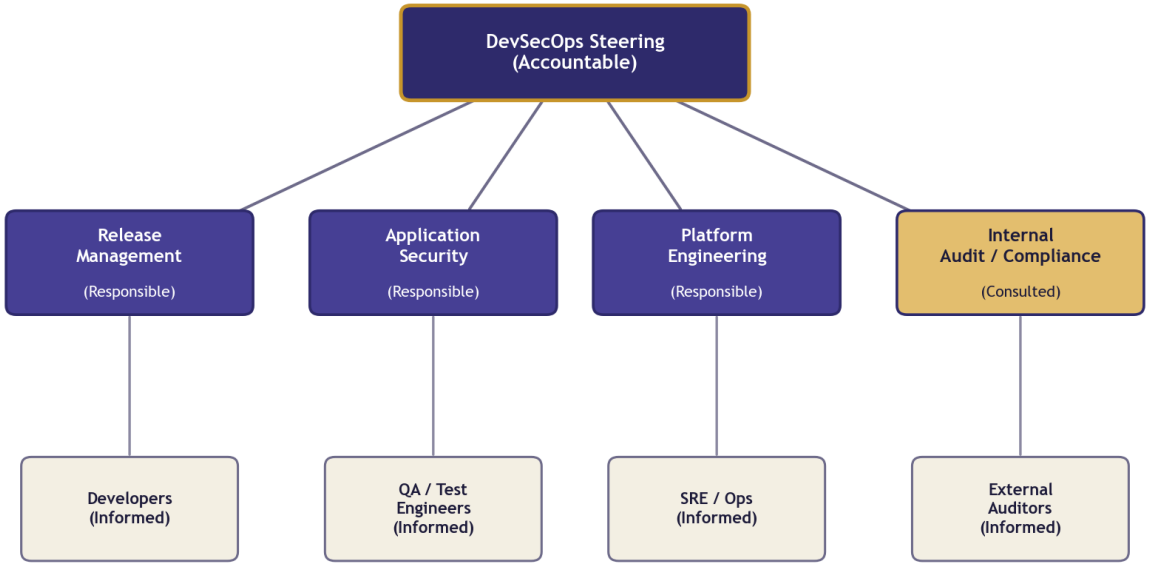


Figure.7. RACI governance tree for DevSecOps transport gate decisions, rooted in a DevSecOps steering function accountable for the overall gate framework.

5.1 Why Accountability Should Sit Above Any Single Team

A recurring governance failure mode is placing gate accountability entirely within release management or entirely within application security, rather than at a steering level that spans both. When accountability sits with a single operational team, that team faces a structural conflict of interest between its own throughput goals (release management) or its own risk-aversion incentives (application security) and the balanced judgment a gate exception decision requires. The steering-level structure in Figure 7 is designed to avoid this by making release management, application security, and platform engineering jointly responsible, with a dedicated accountable steering function above them.

5.2 The Role of Internal Audit and Compliance

Internal audit and compliance functions are positioned as consulted, not responsible, in Figure 7 - they should have visibility into gate design and a voice in setting thresholds, but should not be the operational decision-maker for individual gate exceptions, which would create both a bottleneck and a blurring of the audit function's independence from the process it is meant to assess.

VI. THE ECONOMICS OF DEVSECOPS: COST, HIDDEN COST, AND SENSITIVITY

This section presents the economic core of the article: three complementary illustrative financial views intended to demonstrate an analytical method for building a DevSecOps transport-gating business case, not to report measured figures from any specific organization.

© 2025 IJIRCCE | Volume 13, Issue 5, May 2025| DOI:10.15680/IJIRCCE.2025.1305305

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 8.771| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

$ All dollar figures in this section are illustrative estimates for discussion purposes only. They are not drawn from a specific organization's actual costs, and should not be used as a benchmark or budgeting input without independent validation.

6.1 Cost Waterfall

Figure 3 presents an illustrative annual cost waterfall, starting from a baseline manual-review cost, adding incremental tooling and licensing costs for each gate type, and then subtracting illustrative savings from reduced manual review hours and avoided breach or rework cost, arriving at a net program cost estimate.

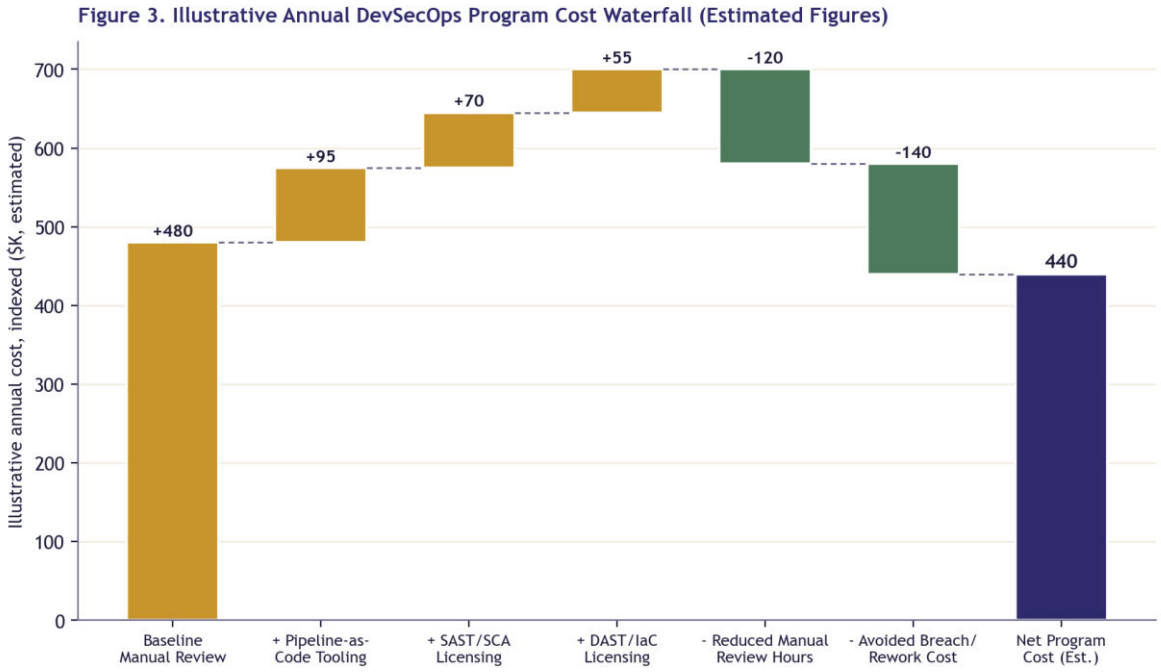


Figure.3. Illustrative annual DevSecOps program cost waterfall, moving from baseline manual-review cost through incremental tooling costs to a net program cost estimate.

6.2 The Hidden-Cost Iceberg

A cost waterfall built only from visible, invoiced costs (tooling licenses, scanning infrastructure) understates the true program cost. Figure 8 presents an iceberg model separating visible costs above the waterline from illustrative hidden or indirect costs below it - developer time lost to false-positive triage, delayed releases from unresolved gate failures, cross-train coordination overhead, and audit preparation effort.

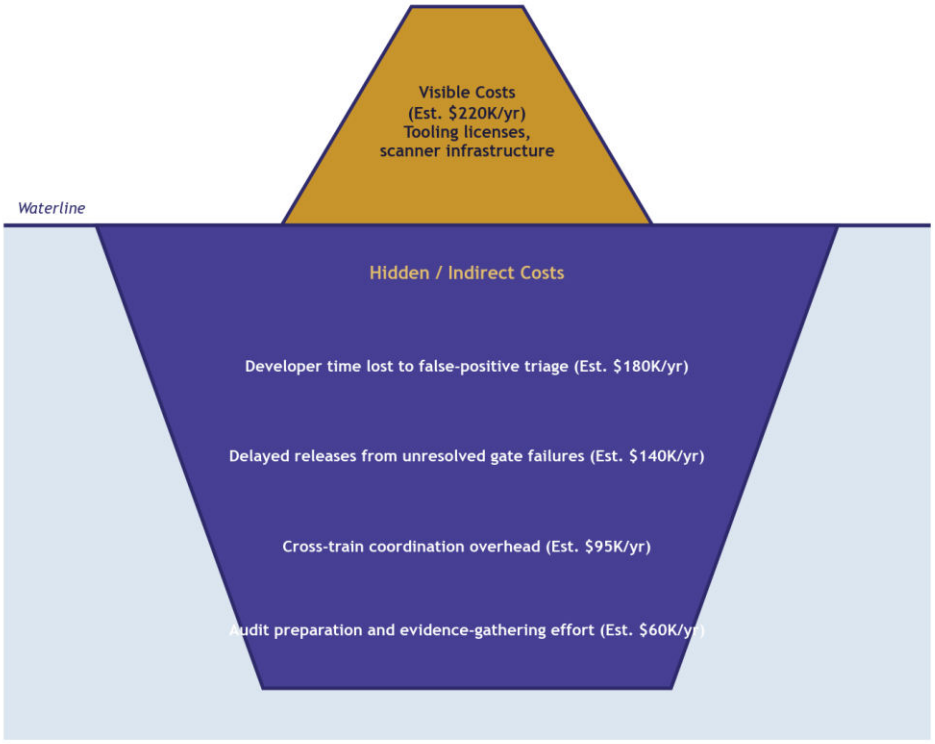
© 2025 IJIRCCE | Volume 13, Issue 5, May 2025| DOI:10.15680/IJIRCCE.2025.1305305

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 8.771| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

Figure 8. Illustrative Iceberg Cost Model: Visible vs. Hidden DevSecOps Program Costs



Note: all dollar figures are illustrative estimates for discussion purposes only and are not drawn from a specific organization's actual costs.

Figure.8. Illustrative iceberg cost model separating visible tooling costs from hidden or indirect costs such as developer triage time and coordination overhead.

In the illustrative figures shown, hidden costs (an estimated $475K per year in aggregate) substantially exceed visible costs (an estimated $220K per year) - a pattern consistent with general DevSecOps cost literature, which frequently observes that indirect costs are underweighted in initial business cases precisely because they are harder to itemize on an invoice.

$ The $475K and $220K aggregate figures cited above, and all other dollar amounts in this section, are illustrative estimates for discussion purposes only. They are not drawn from a specific organization's actual costs and should not be used as a benchmark or budgeting input without independent validation.

6.3 Sensitivity (Tornado) Analysis

Figure 9 presents an illustrative tornado chart ranking the value drivers behind the net program cost estimate by the width of their plausible impact range, from largest (avoided breach or incident cost) to smallest (audit preparation savings). This kind of analysis is useful for directing further data-gathering effort: drivers with the widest uncertainty range are the ones where additional measurement would most improve the business case's reliability.

© 2025 IJIRCCE | Volume 13, Issue 5, May 2025| DOI:10.15680/IJIRCCE.2025.1305305

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 8.771| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

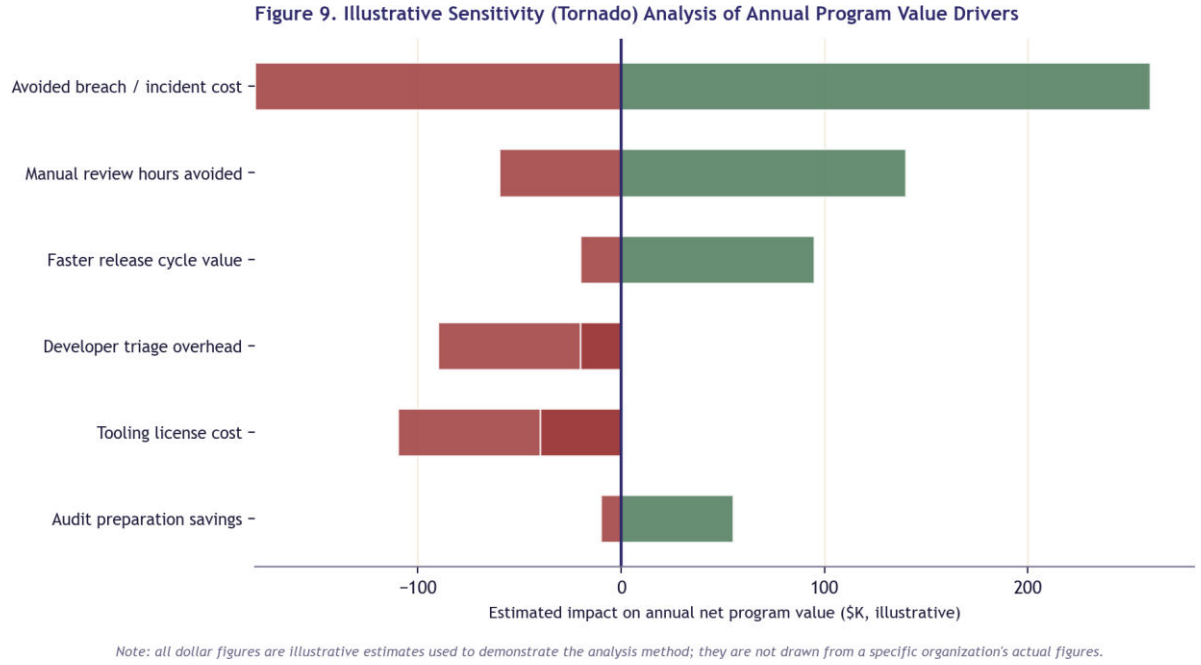


Figure. 9. Illustrative sensitivity (tornado) analysis, ranking value drivers by the width of their plausible impact range on annual net program value.

6.4 Finding Classes by Cost and Frequency

Figure 5 presents an illustrative bubble chart plotting finding classes by how frequently they occur, their estimated mean remediation cost, and their severity (encoded as bubble color). This view complements the tornado analysis by identifying which specific finding classes are contributing most to the developer-triage and remediation cost components of the iceberg model.

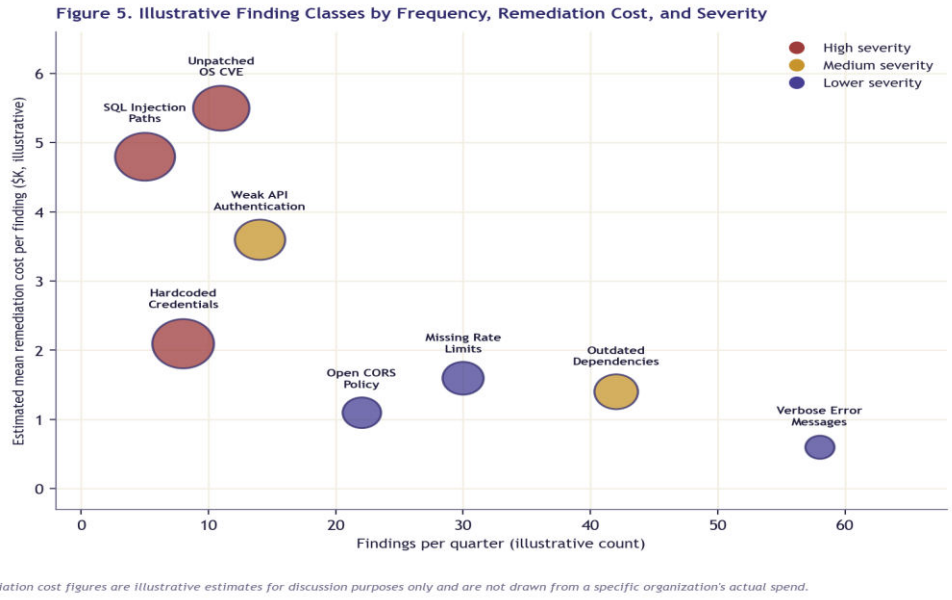


Figure 5. Illustrative finding classes by frequency, estimated remediation cost, and severity - bubble size and color both encode severity.

© 2025 IJIRCCE | Volume 13, Issue 5, May 2025| DOI:10.15680/IJIRCCE.2025.1305305

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 8.771| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

$ Remediation cost figures in Figure 5 are illustrative estimates for discussion purposes only and are not drawn from a specific organization's actual spend.

VII. MEASURING GATE PERFORMANCE OVER TIME

Cost and value estimates (Section 6) are only as credible as the operational data feeding them. Figure 4 presents a statistical process control (SPC) chart tracking weekly gate first-pass rate against a center line and upper/lower control limits, a technique borrowed from manufacturing quality control and applicable here with minimal adaptation.

Figure 4. Illustrative Statistical Process Control Chart: Weekly Gate First-Pass Rate

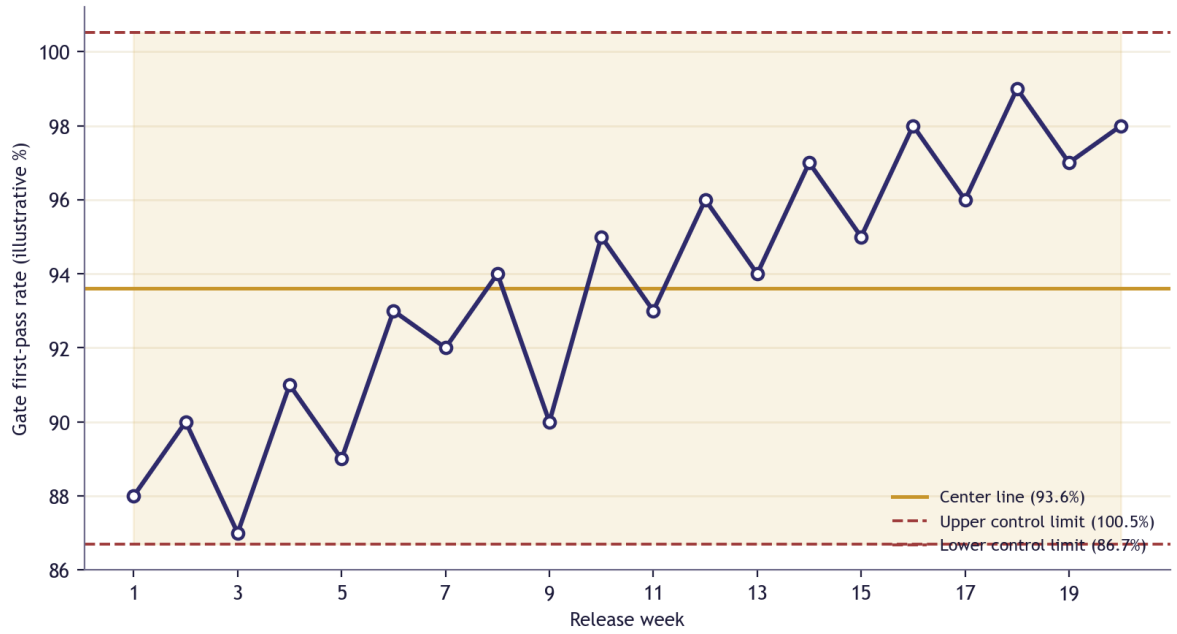


Figure.4. Illustrative statistical process control chart tracking weekly gate first-pass rate against a center line and control limits over a twenty-week period.

7.1 Why Control Limits Matter More Than the Trend Line Alone

A simple upward trend line, of the kind shown in prior treatments of gate adoption, can mask week-to-week volatility that matters operationally: a single week's sharp dip below the lower control limit is a signal worth investigating (a bad release, a tooling outage, a process change) even if the multi-month trend is positive. Control limits, calculated from the observed mean and standard deviation, give a statistically grounded threshold for distinguishing normal variation from an out-of-control signal, rather than relying on subjective judgment about whether a given week's dip is "bad enough" to investigate.

Figure 6 presents an illustrative bump chart tracking how the relative ranking of blocked-transport root causes shifted across four quarters - showing, in this illustration, code-level SAST findings overtaking dependency vulnerabilities as the leading root cause by 2024 Q4, while configuration/IaC drift, insufficient test coverage, and secrets in repository history held comparatively stable ranks.

© 2025 IJIRCCE | Volume 13, Issue 5, May 2025| DOI:10.15680/IJIRCCE.2025.1305305

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 8.771| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

Figure 6. Illustrative Rank Trend of Blocked-Transport Root Causes by Quarter

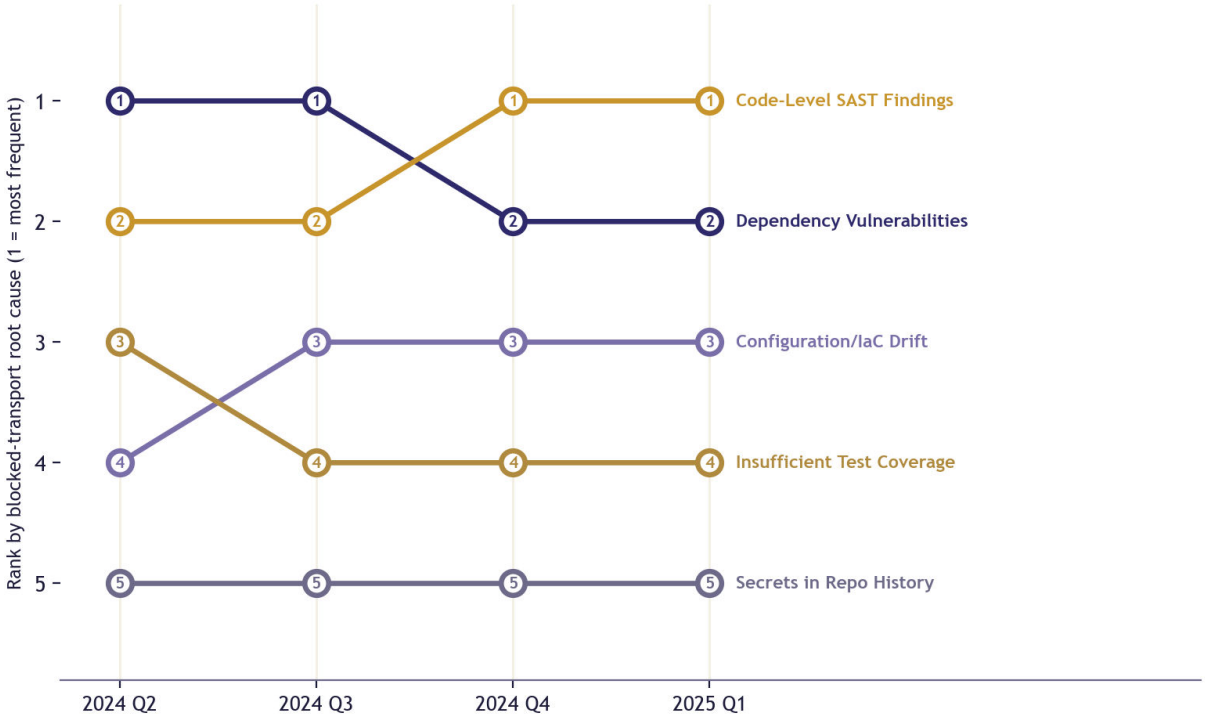


Figure.6. Illustrative rank trend of blocked-transport root causes across four quarters, showing code-level SAST findings overtaking dependency vulnerabilities.

A rank-trend view like this is more actionable than a static root-cause breakdown for one period, because it directs attention to root causes that are worsening in relative terms even if their absolute count is not yet the largest - an early-warning signal that a static snapshot would miss.

VIII. ROOT-CAUSE TRENDS AND CALENDAR-LEVEL PATTERNS

8.1 Calendar-Level Failure Patterns

Figure 10 presents an illustrative calendar heatmap of gate failures by day of week and release week across one quarter, surfacing patterns that a simple weekly or monthly aggregate would obscure - for example, an elevated Thursday failure rate potentially linked to a pre-release rush to complete transports before a Friday cutoff, and an elevated failure count in specific weeks potentially corresponding to release crunch periods.

© 2025 IJIRCCE | Volume 13, Issue 5, May 2025| DOI:10.15680/IJIRCCE.2025.1305305

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 8.771| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

Figure 10. Illustrative Calendar Heatmap of Gate Failures by Day and Release Week (One Quarter)

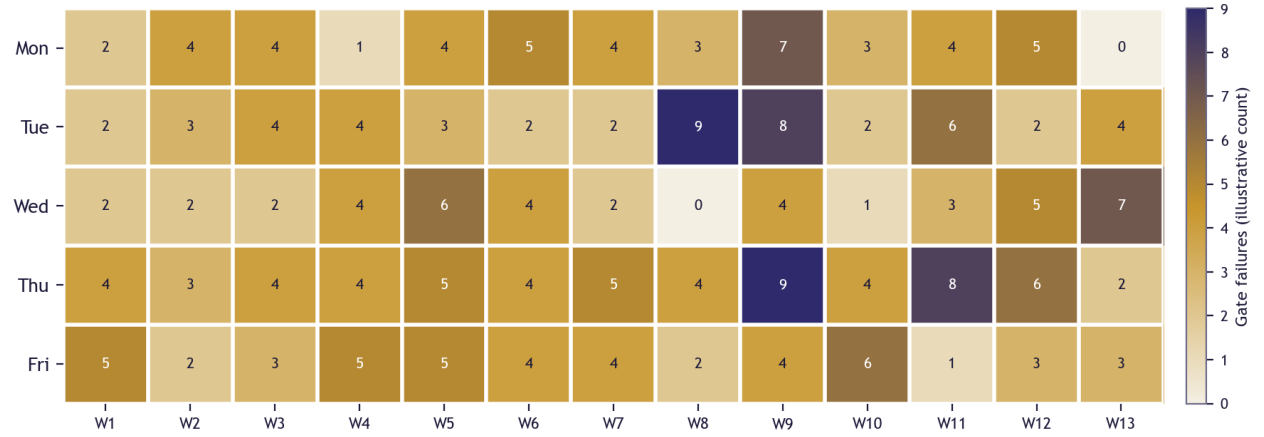


Figure 10. Illustrative calendar heatmap of gate failures by day of week and release week, surfacing day-of-week and crunch-period patterns.

Calendar-level views like this are a useful complement to the SPC chart in Section 7: where the SPC chart answers "is this week's overall performance in control," the calendar heatmap answers "is there a structural pattern by day or by release-cycle timing that a single control chart would average away."

IX. IMPLEMENTATION CONSIDERATIONS

Organizations building the measurement and governance capabilities described in Sections 5 through 8 should sequence the underlying data collection deliberately rather than attempting to stand up every view simultaneously.
Establish gate result logging first. Every gate pass/fail decision, its timestamp, and its associated finding details must be captured in a queryable format before any of the analytical views in this article (SPC, bump chart, calendar heatmap) can be constructed.

Backfill at least one full quarter before publishing trend views. Rank-trend (Figure 6) and calendar-pattern (Figure 10) views require enough historical data to distinguish genuine trends from noise; publishing them prematurely on a few weeks of data risks drawing false conclusions.

Pair cost modeling with a named data owner. The illustrative cost waterfall and iceberg models in Section 6 depend on inputs (developer hours, licensing costs, incident cost estimates) that should have a designated owner responsible for keeping them current, not a one-time estimation exercise.

Review control limits and thresholds quarterly. As referenced in Section 7, SPC control limits calculated from an early, immature period of gate operation may not remain appropriate as the process matures - plan a recurring recalibration rather than treating the initial limits as permanent.

X. RISKS, LIMITATIONS, AND OPEN QUESTIONS

10.1 Risks

Gate fatigue and exception creep. If exception requests for blocked transports become routine, the gate's effective strictness erodes even though its configuration has not changed.

Over-reliance on illustrative cost models. The cost waterfall, iceberg, and tornado analyses in Section 6 are presented explicitly as illustrative; a real business case must substitute an organization's own measured figures before those models can support an actual budgeting or investment decision.

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 8.771| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

Statistical misinterpretation of control charts. SPC control limits assume a reasonably stable underlying process; a control chart applied across a period of major pipeline architecture change may generate misleading out-of-control signals that reflect the change itself rather than a genuine quality issue.

10.2 Limitations of This Article

This article synthesizes patterns from publicly available vendor documentation, DevSecOps practice literature, and industry commentary current through April 2025. All quantitative figures in Section 6, and the illustrative datasets underlying Figures 4, 5, 6, and 10, are explicitly constructed for demonstration purposes and are not measured from a specific organization's pipeline.

10.3 Open Questions

How should the hidden-cost iceberg model (Section 6.2) be operationalized with real developer time-tracking data without introducing an invasive or resented measurement burden on engineering teams?

What is the appropriate frequency for recalibrating SPC control limits (Section 7.1) as a DevSecOps program matures, and should recalibration be triggered by a calendar schedule or by a detected structural shift in the underlying data?
As ERP vendors ship more first-party AI-assisted code and configuration generation, how should the root-cause taxonomy underlying the bump chart (Section 8) evolve to capture AI-specific failure modes distinctly from traditional human-authored defect classes?

XI. CONCLUSION

A DevSecOps transport-gating program is easiest to justify at the moment of initial adoption, when the case rests on qualitative risk reduction, and hardest to sustain in year two and beyond, when stakeholders reasonably ask what the program has cost and what it has delivered. The economic and measurement views presented in this article - the cost waterfall, the hidden-cost iceberg, the sensitivity analysis, the control chart, and the root-cause trend views - are offered as a starting toolkit for answering that second, harder question with the same rigor that architectural design already receives. Enterprises that build these measurement disciplines alongside the pipeline itself, rather than retrofitting them once a steering committee asks for numbers, will be far better positioned to sustain their DevSecOps investment through the inevitable budget scrutiny that follows any multi-year infrastructure program.

REFERENCES

[1] Amazon Web Services. "AWS DevOps Guru and CodeGuru: Security Findings Integration." AWS Documentation, 2024.
[2] Atlassian. "State of DevOps and Security: 2024 Report." Atlassian Research, 2024.
[3] Cloud Security Alliance. "DevSecOps: A Practitioner's Guide." CSA Publications, 2023.
[4] Forrester Research, Inc. "The State of Application Security, 2024." Forrester Research, 2024.
[5] Gartner, Inc. "Market Guide for DevSecOps Tools." Gartner Research, 2024.
[6] Gartner, Inc. "Predicts 2025: Application Security and Software Supply Chain." Gartner Research, 2024.
[7] GitLab, Inc. "2024 Global DevSecOps Survey." GitLab Research, 2024.
[8] GitHub, Inc. "The State of the Octoverse: Security." GitHub Publications, 2024.
[9] IBM Corporation. "Cost of a Data Breach Report 2024." IBM Security, 2024.
[10] Linux Foundation / OpenSSF. "Software Supply Chain Security Best Practices." OpenSSF Documentation, 2024.
[11] MITRE Corporation. "Common Weakness Enumeration (CWE) Top 25 Most Dangerous Software Weaknesses." MITRE, 2024.
[12] National Institute of Standards and Technology (NIST). "Secure Software Development Framework (SSDF), SP 800-218." U.S. Department of Commerce, 2022.
[13] National Institute of Standards and Technology (NIST). "AI Risk Management Framework (AI RMF 1.0)." U.S. Department of Commerce, 2023.
[14] OWASP Foundation. "OWASP DevSecOps Guideline." OWASP Documentation, 2024.
[15] OWASP Foundation. "OWASP Top 10 for CI/CD Security Risks." OWASP Documentation, 2023.
[16] Project Management Institute. "Practice Standard for Project Risk Management: Sensitivity Analysis Techniques." PMI Publications, 2019.
[17] SAP SE. "SAP Code Vulnerability Analyzer: Overview and Best Practices." SAP Help Portal, 2024.

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 8.771| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

[18] SAP SE. "SAP Cloud ALM for Transport and Release Management." SAP Documentation, 2024.
[19] SANS Institute. "DevSecOps Survey 2024: Practices and Maturity." SANS Institute Publications, 2024.
[20] Snyk, Ltd. "State of Open Source Security 2024." Snyk Research, 2024.
[21] Sonatype, Inc. "State of the Software Supply Chain, 10th Edition." Sonatype Research, 2024.
[22] Synopsys, Inc. "2024 Open Source Security and Risk Analysis (OSSRA) Report." Synopsys Software Integrity Group, 2024.
[23] U.S. Cybersecurity and Infrastructure Security Agency (CISA). "Secure by Design Principles." CISA Publications, 2023.
[24] Veracode, Inc. "State of Software Security 2024." Veracode Research, 2024.
[25] World Economic Forum. "Global Cybersecurity Outlook 2025." WEF Publications, 2025.
[26] American Society for Quality (ASQ). "Statistical Process Control (SPC) Fundamentals." ASQ Quality Press, 2021.
[27] European Union Agency for Cybersecurity (ENISA). "Threat Landscape for Supply Chain Attacks." ENISA Publications, 2023.

Appendix A: Glossary of Terms

Bump Chart - A line chart tracking the relative rank of multiple categories over time, useful for showing how a ranking shifts even when absolute values are not directly comparable.

CI/CD (Continuous Integration / Continuous Delivery) - A software development practice in which code and configuration changes are automatically built, tested, and prepared for release on a frequent, often per-commit basis.

DevSecOps - A software delivery practice that embeds security testing and review directly into the CI/CD pipeline rather than treating it as a separate, later-stage activity.

ERP (Enterprise Resource Planning) - Integrated software managing core business processes such as finance, procurement, supply chain, and human resources within a single system of record.

Iceberg Cost Model - A cost visualization separating visible, directly invoiced costs from hidden or indirect costs that are harder to itemize but often larger in aggregate.

RACI - A responsibility assignment framework categorizing stakeholders as Responsible, Accountable, Consulted, or Informed for a given decision or activity.

Release Train - A scheduled, recurring release cycle to which a defined set of changes is assigned, allowing multiple release trains to be in different stages of development, testing, and deployment concurrently.

Sankey Diagram - A flow visualization in which the width of a band is proportional to the quantity it represents, commonly used to show how a starting quantity splits across successive stages or outcomes.

SPC (Statistical Process Control) - A method, originating in manufacturing quality control, for monitoring a process over time using a center line and calculated control limits to distinguish normal variation from an out-of-control signal.

Tornado Chart - A horizontal bar chart used in sensitivity analysis, ranking input variables by the width of their impact on an output metric, typically sorted widest at the top.

Transport - In ERP change management, a packaged set of configuration or code objects moved between systems in a landscape, typically from development toward production.

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

INNO SPACE SJIF Scientific Journal Impact Factor

doi crossref

निस्केयर NISCAIR

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN COMPUTER & COMMUNICATION ENGINEERING

9940 572 462 6381 907 438 ijircce@gmail.com

www.ijircce.com

Scan to save the contact details