International Journal of Computational and Experimental Science and ENgineering (IJCESEN)

Vol. 11-No.2 (2025) pp. 3790-3803
http://www.ijcesen.com

ISSN: 2149-9144

Copyright © IJCESEN

Research Article

Predictive Change Governance in Enterprise SAP Landscapes: A Risk-Aware AI Model for Transport Release Optimization

Karthik Paramasivam*

PSG College of Technology, Peelamedu, Coimbatore, Tamil Nadu 641004, India
* **Corresponding Author Email:** karthik.sapalm@gmail.com- **ORCID:** 0000-0002-5247-7330

Article Info:

DOI: 10.22399/ijcesen.5521
Received : 15 January 2025
Revised : 20 March 2025
Accepted : 27 March 2025

Keywords

SAP transport governance;
predictive change governance;
enterprise application lifecycle management;
transport risk scoring;
release optimization;
regression test prioritization

Abstract:

When dealing with enterprise SAP landscapes, the release of business change must happen rapidly and be regression stable, transport stable, audit traceable and compliant with control. There is a wealth of existing SAP governance evidence (which is not necessarily coordinated) which includes Central ATC code quality findings, BPCA business-process impact estimates, UPL and SCMON production-usage signals, Test Suite and HPALM regression evidence, ChaRM transport history, and CSOL and Downgrade Protection collision risk. However, these evidence streams are only reported in typical governance workflows in isolation and are not natively integrated into an explainable transport-level risk score that can be leveraged to drive release sequencing and test prioritization. This paper outlines a model for predictive SAP transport governance, which we call the Transport Risk Score (TRS), proposed by the practitioners. The model combines the ATC delta severity, BPCA process criticality, test coverage gaps, object dependency depth, historical incident rate, CSOL/Downgrade collision risk, Retrofit conflict risk, emergency change factors and calendar / freeze period risk to a normalized 0-100 score. The paper also introduces a risk aware release queue optimizer, a UPL-enriched BPCA test-scope optimizer, a TRS Evidence Package Generator and a validation protocol with chronological splits, calibration, precision-recall analysis, false-negative cost and decision utility. The study draws on academic literature on ERP governance, static analysis, regression test selection, defect prediction, process mining, release planning and explainable AI, as well as anonymised design evidence from SAP solution managers in enterprise ERP implementations across the food service, beverage, pharmaceutical, semiconductor, healthcare, retail and oil and gas sectors, together with anonymized practitioner design evidence from enterprise SAP implementation contexts. The study draws on academic literature on ERP governance, static analysis, regression test selection, defect prediction, process mining, release planning, and explainable AI, together with anonymized practitioner design evidence from enterprise SAP implementation contexts across the food service, beverage, pharmaceutical, semiconductor, healthcare, retail and oil and gas sectors provided by enterprise SAP solution managers at the official SAP forums. TRS is not intended to provide autonomous release authority; rather, it offers a reusable, auditable decision-support mechanism for transport-risk visibility, targeted regression, release optimization, and post-release learning.

1. Introduction

Unlike conventional software repositories, enterprise SAP environments may combine ABAP objects, customizing elements, authorization changes, interface definitions, workflow changes, forms, and test artifacts within a single transport. The number of changed objects is only part of the equation; when such a transport is released, it is also dependent on whether the objects are executed in production, whether they are part of critical business processes, whether Central ATC findings are newly introduced or inherited, whether BPCA indicates impacted process steps, whether test coverage exists, whether related objects are locked through CSOL or flagged by Downgrade Protection, and whether the change occurs near a period close, production freeze, or major upgrade window.Many of the evidence streams required for risk-aware governance are provided by SAP

Solution Manager and SAP ALM capabilities. SAP Help mentions central quality checking of ABAP programs and repository objects via the ABAP Test Cockpit (ATC) as a means to check the quality of programs for the purpose of consolidation or quality system [31]. Test Suite provides an option to analyse and plan for change [32, 33] using BPCA. Additionally, SAP Help provides documentation for Retrofit for dual-landscape synchronization [34], Administration Cockpit functions for Cross-System Object Lock and Downgrade Protection [35] and Solution Manager usage logging with UPL and SCMON data [36]. From the transport-level predictive decision support perspective these capabilities are disjointed but provide a solid technical basis for governed change.

The proposed Transport Risk Score model is based on anonymized design experience from enterprise SAP Solution Manager implementations in diverse industry scenarios, such as the food service and beverage industry, the pharmaceutical industry, the semiconductor industry, the healthcare industry, the retail industry and the oil and gas industry. These implementations covered Central ABAP Test Cockpit Hub configuration, Business Process Change Analyzer dynamic and semi-dynamic test bill-of-materials generation with production usage information, Custom Development Management Cockpit analysis in large managed-system landscapes and release control in compliance-sensitive environments. The common theme in all these implementations was that there was lots of evidence to be found in separate SAP transactions, but none of them was brought together into a risk-based decision-support output that release managers, change managers, test leads, or auditors could act on. This paper fills that gap by defining a named, normalized, explainable model which fills the integration absence [A1, A2].

How can the Central ATC findings, BPCA impact analysis, UPL/SCMON production usage, Test Suite evidence, ChaRM history, CSOL/Downgrade signals, Retrofit context and release-calendar constraints be combined into an explainable SAP transport-risk model that enables risk-aware testing and release optimization? The paper is intentionally designed as a model-driven SAP transport governance paper. A related enterprise-level architecture addresses autonomous ALM governance, whereas the present paper focuses specifically on transport-level feature extraction, risk scoring, regression prioritization, release-queue optimization, validation, and evidence packaging. This separation minimizes conceptual overlap and places TRS as a unique and citable contribution.

2. Research Gap and Contribution

The literature on static analysis, regression test selection, defect prediction, release planning, and process mining shows that there are strong theoretical mechanisms, but these do not directly relate to the problem of transport in the SAP. The literature reviewed assumes that it is at the level of a source-code repository, test suite, change set, or feature release. SAP transport brings in some extra dimensions: customizing dependencies, business-process mappings, Solution Documentation structures, ChaRM transaction types, CSOL/Downgrade state, Retrofit synchronization, and audit evidence requirements.

Four common governance gaps were identified in the anonymized practitioner design evidence from the author's experience implementing SAP Solution Manager: Object-level quality findings aren't always mapped to transport-level risk; business-process impact scope isn't always weighed by production usage; production telemetry isn't integrated as a predictive feature for transport-level scoring; and compliance evidence is manually consolidated across multiple sources. This manuscript takes those observations and turns them into a contribution in a journal-safe, anonymous, model-driven way.

The paper's contribution is five-fold:

1. It establishes the Transport Risk Score (TRS) of a SAP transport, which is a risk model with a normalized range of 0 to 100.
2. It adds a set of feature normalization rules to map SAP-native evidence streams to comparable risk factors.
3. It suggests a new UPL-based BPCA test-scope optimizer which prioritizes tests with both technical impact and production usage.
1. It specifies a risk-aware release queue optimizer which takes in TRS as one of the dependencies, test capacity, calendar constraints, and rollback difficulty.
4. Presents a validation and evidence-governance process appropriate for compliance-sensitive SAP landscapes.

3. Literature and SAP Technical Grounding

3.1 Enterprise systems, customization, and transport risk

Numerous ERP studies have confirmed that ERP systems are integral parts of organizational systems and not applications [1–6]. The greater the level of customization and adaptation in post-implementation, the more complex the maintenance becomes as all vendor code, customer extensions, configuration, authorizations and process-specific

routines need to be kept in sync across the enterprise system landscape [6, 7]. In SAP landscapes, this maintenance complexity is manifested in the form of transport requests, release cycles, quality gates, and after-release incident handling. Practitioners experience this as an operational challenge when implementing ChaRM N/N+1 landscape governance in multi-sector SAP estates [A1, A2]. The model of a transport should therefore be something more than just a code change; it should be a multi-level risk object. The risk it poses is related to code quality, process criticality, how often it is used, the level of dependency, test coverage, incident history and the context of the release. This encourages a composite risk score, and not one static-analysis gate.

3.2 Static analysis and Central ATC-like gates

Static analysis research establishes that code can be assessed for defects, data-flow anomalies, control-flow issues, and maintainability problems without execution [8, 9]. Developer-adoption studies also show that static-analysis value depends on actionable findings, false-positive management, workflow integration, and fix prioritization [8]. In SAP, Central ATC provides a practical mechanism for code-quality evidence [30, 31], but a raw count of findings is not the same as transport risk. A high-severity finding in a frequently executed financial posting enhancement is more consequential than the same finding in a rarely used report. TRS therefore treats ATC findings as delta-weighted, object-contextual evidence rather than as a flat release veto.

3.3 BPCA, change impact analysis, and regression test selection

Previous work in change impact analysis and program slicing demonstrate estimation of dependency relationships to determine affected artifacts [10, 11]. The research for regression test selection also points out that safety, precision, effort and inclusiveness are important factors to consider for test selection [12-14]. The operational analogue in SAP is BPCA, which connects changes to process steps and test scope [32, 33]. But business criticality, production execution frequency and compliance sensitivity should also be taken into account when using SAP test prioritization. This paper thus builds upon the BPCA reasoning for UPL/SCMON production usage [36] and gaps in test coverage.

3.4 Defect prediction, validation, and explainability

The software fault prediction literature suggests that several measures, such as software metrics, prior faults, change history, and process indicators, can aid in the ranking of software risks [15, 16]. Benchmarking studies however caution that it is more important to have data quality, evaluation design, leakage control and validation protocol than the choice of algorithm [17–20]. In SAP transport governance, random train-test splits would not be appropriate as transports are time-ordered and are frequently shared over release cycles with objects, packages and teams. Chronological or rolling-window validation must be used for TRS evaluation. The above mentioned release decisions are high accountability decisions, therefore interpretable baselines or explainable model outputs are desired [28, 29].

3.5 Release planning and optimization

Research models for the software release selection problem with value, cost, priority and dependency constraints are released by release planning [21, 22]. Search-based software engineering shows that multi-objective optimization can support human decision-making when objectives conflict [23]. SAP transport governance takes this issue of feature selection further with transport bundling, sequencing, deferral, dependency integrity, release calendar management, test-capacity allocation and rollback readiness. Therefore, a release optimizer should try to reduce the expected exposure to a release rather than just increase the throughput.

3.6 SAP official documentation as technical grounding

Because peer-reviewed literature directly evaluating SAP Central ATC, BPCA, UPL/SCMON, Retrofit, CSOL, Downgrade Protection, and CDMC remains limited, this paper uses SAP Help Portal documentation for SAP-specific technical grounding. In addition, ATC is presented as a quality checking functionality for ABAP code and repository objects [30], central ATC quality checking [31], BPCA and Test Suite functions for change impact and tailored test planning [32, 33] are covered, as well as Retrofit for dual landscape synchronization [34] and Administration Cockpit functions for CSOL and Downgrade Protection [35]. Custom Code Management including CDMC is covered [37, 38] as well as usage data collection for UPL and SCMON [36]. Table 1 summarizes the evidence-to-TRS grounding across academic and SAP technical sources:

4. Practitioner-Grounded Design Requirements

The Transport Risk Score model results are based on anonymized practitioner evidence gathered during enterprise SAP Solution Manager implementations that include Change Request Management, central code-quality checking, business-process change analysis, dynamic and semi-dynamic test bill-of-materials, production usage logging, custom-code lifecycle analysis, Test Suite, HP ALM, component-based test automation, cross-system object lock, downgrade protection, retrofit, and compliance-sensitive release control. This design evidence is not in the form of a statistical field study and does not contain any of the client's confidential design records. It is used as a design-science input to identify the need for recurring operational requirements that are not well captured in generic software-engineering literature. The design requirements are based on documented practitioner experience with N/N+1 landscape governance, cross-system object lock, downgrade protection, enhanced retrofit, Test Suite integration, business-process change analysis, and test-scope optimization, dynamic test-bill-of-material creation, and production usage data for custom-code analysis. The anonymized practitioner contexts are correlated with TRS design requirements in Table 2.

5. Transport Risk Score Model Design

5.1 Model overview and formula

TRS is a transport-level risk score. All factors are first scaled to a 0 to 100 range and then summed together with a weighting system formulated by experts in the field. The default weights are suggested as a starting point for research and initial implementation, and should be recalibrated based on incident history, the control environment, transport volume, business criticality, and audit tolerance. The weights add up to 1.00.

TRS = 0.22(ATC Delta Severity) + 0.18(BPCA Process Criticality) + 0.14(Test Coverage Gap) + 0.12(Object Dependency Depth) + 0.10(Historical Incident Rate) + 0.09(CSOL/Downgrade Collision Risk) + 0.08(Retrofit/Release Conflict Risk) + 0.04(Emergency Change Factor) + 0.03(Calendar/Freeze Risk)

Normalized 0–100 · Weights sum to 1.00 · Expert-derived · Assign risk band and recommended incident history

5.2 Risk bands and governance actions

There are four bands of TRS risk defined in Table 3, which has the recommended governance actions. The proposed band boundaries and actions are considered a baseline threshold and should be customised by organisations in relation to their risk appetite and operating environment.

5.3 Feature normalization

Using raw SAP evidence directly in a weighted equation is not useful unless the evidence is normalized into comparable numerical features. One possible normalization scheme is defined in Table 4. The rules for normalisation are deliberately written in such a way as to enable the release manager and auditor to see why a transport is awarded a specific score.

5.4 Worked example

Table 5 shows an example of computing the TRS for a hypothetical transport. The numbers given are for illustrative purposes and the method is shown only.

5.5 Algorithmic workflow

The TRS computation and governance recommendation procedure is described in Algorithm 1:

Algorithm 1: TRS Computation and Governance Recommendation

Input: Transport request T, ChaRM document C, release window R

1. Extract objects, packages, change type, requester, status, and target landscape from C and T.
2. Query Central ATC for baseline, introduced, worsened, remediated, and exempted findings.
3. Query BPCA for impacted process steps, transactions, tests, and test evidence.
4. Query UPL/SCMON for production execution frequency and coverage for affected objects and transactions.
5. Query ChaRM/ITSM history for incidents, rework, emergency changes, and rollback links.
6. Query CSOL/Downgrade and Retrofit collision and sequence signals.
7. Normalize all features to the 0–100 scale defined in Table 4.
8. Compute TRS using the weighted formula in Section 5.1.
9. Assign risk band and recommended governance action using Table 3.
10. Generate TRS Evidence Package with features, score, explanation, action, approvals, and post-release outcome.

6. UPL-Enriched BPCA Test Scope Optimization

The utility of using standard BPCA-style impact analysis is that it can identify potentially impacted process steps and tests. But a process path that is theoretically impacted does not necessarily imply a high risk production path. A rarely used variation can eat up test time that should be spent on a high volume transaction or a compliance type process. The use of UPL/SCMON can be used to discriminate between theoretical and production-relevant impact. Practitioner observations suggest that dynamic TBOMs may over-scope rarely executed process variants and under-prioritize high-frequency production paths that UPL/SCMON data could help identify.

The proposed optimization algorithm goes through 4 steps. First, BPCA determines impacted transactions, process steps and test cases. Second, UPL/SCMON adds to each affected element the information of execution frequency and coverage. Third, test cases are sorted according to their production use, process criticality and coverage gap, to be mandatory, priority, optional or exploratory test cases. Fourth, the final test manifest accompanies the TRS Evidence Package so as to allow for auditability: it is not just cut down; it is justified.

7. Risk-Aware Release Queue Optimizer

The release queue optimizer is not the sole decision criterion, it is an input. High-risk transports may still be released when they mitigate a significant production incident, whereas medium-risk transports may be postponed if dependencies remain unresolved or the release window falls within a freeze period. Optimizer takes into account the risk, dependency, capacity, priority, rollback difficulty and release calendar constraints at once.

Minimize: Expected Release Exposure = Σ TRS(t) + DependencyPenalty(t) + TestCapacityPenalty(t) + CalendarPenalty(t) + RollbackDifficulty(t)

Under the following restrictions:

1. Overall, prerequisite transports have to be released first and then dependent transports.
2. Before the import of PRD, regression evidence needs to be completed for critical business-process transports.
3. Transports with conflicts in CSOL and Downgrade Protection will not be released without exception approval.
4. Critical-risk transports (TRS > 80) must be approved by the release manager or CAB.
5. Emergency transports can be made outside of normal timings only if there is a need for post-deployment review and monitoring evidence.
6. Release sequencing must preserve Retrofit integrity across N and N+1 landscapes

8. TRS Evidence Package Generator

The TRS Evidence Package Generator is more specific than a general ALM compliance structure. It is an evidence output that occurs at the transport level and contains information on the features on the transport, the normalized scores, weights, overall TRS, the recommended governance action, the approval record, the test evidence, exception justifications, the model version, and the post-release outcome of each transport evaluated by the TRS model. This transport-level focus differentiates this research stream from Paper 1 of this research stream which addresses compliance from an enterprise-level point of view.

9. Data Model for an SAP Transport-Risk Dataset

There is a significant lack of SAP transport-risk datasets that are benchmarkable. A reproducible dataset should be a linked data set, not a flat row. Names of objects, users, packages and process labels should be anonymised or hashed, if required, but with dependency relationships and time order retained.

10. Validation Framework

TRS needs to be considered a governance decision support model, rather than simply a classification. Accuracy alone is insufficient because incident-causing transports are likely to form a minority class. Most of the time, a model that is accurate will suggest low risk for most transports. Precision-recall, calibration, false-negative cost and decision utility are more useful. For validation, chronological splits are preferred: train on transports before date D, test on transports after date D and do rolling windows training and testing. This design minimizes temporal leakage and more accurately represents deployment in operations.

11. Data Privacy and Model Governance

A TRS implementation should not reveal personally identifiable information for the developer, requester, tester, or approver in model outputs except for the purpose of accountability in governance. If possible, training sets should hash names of objects, user IDs, package names, and identifiers of business processes and retain the

dependency structure. Be careful of attributing blame towards people when explaining risk in model outputs.

The following should be stored as governed metadata: TRS model versions, training windows, feature definitions, threshold changes, calibration outcomes, override decisions, and release outcomes. Any change to the TRS model may be a governed ALM change as it can impact the approval, delay, block, or escalation of transports. Model updates should be managed accordingly, following version control, approval, rollback, monitoring and audit-review processes according to the ChaRM change governance framework.

12. Discussion

The primary benefit of TRS is integration. Central ATC provides technical-quality evidence, but it cannot, by itself, establish business consequence. However, BPCA alone is not sufficient to prioritize test effort without any production usage and test coverage. UPL/SCMON is a runtime evidence but does not give the release risk. Chronological events are recorded in ChaRM, but predictive scoring is not a feature of ChaRM. CSOL and Downgrade Protection reveal conflicts but are not sufficient to quantify the total exposure of release. TRS transforms these signals into a coherent, understandable model.

The model isn't meant to be a stand-alone release authority. It is a tool for decision support that makes release risk visible, explainable and auditable. Human release managers, CABs, business process owners and compliance stakeholders continue to have a financial, regulatory, operational or customer responsibility for decisions.

The practical implementation of a TRS could start with a rule-based weighting and progress through the development of calibrated machine-learning models. An initial weighted model may provide a transparent baseline, but it should be calibrated and validated before operational use. As the labelled transport outcomes continue to be accumulated, the organizations are able to train classifiers or gradient models that are interpretable and keep the weighted score as a governance baseline and explainability anchor.

13. Limitations

The results are not a controlled field test, and this paper presents a model and validation framework. The main limitations are as follows

1. The default weights are derived from experts and need to be recalibrated prior to production use. Target weights are only valid in the context of the specific industry and may not be directly applicable to other industries without local incident history.
2. The outcome labels for transports may be noisy since incidents may not always be associated with the transport in ChaRM or ITSM records.
3. UPL/SCMON data captures frequency of execution, but not necessarily business consequence or reversibility. Labels for business processes must be provided by domain experts.
4. The completeness of the BPCA mappings can be impacted if Solution Documentation, TBOMs or test-case links are stale or have not been updated as a result of SOLDOC migrations.
5. CSOL and Downgrade Protection signals assume that the landscape is consistently configured and maintained across managed systems
6. Release calendars, compliance requirements, variations of the ATC rules, transport volume, and incident taxonomies vary across industries making it difficult to apply models trained in one industry to another.

14. Future Research

Create anonymised ERP transport benchmark datasets for transports, objects, ATC findings, BPCA links, UPL/SCMON usage, test evidence, incident labels, based on the schema from Table 8.

Under rolling window validation, compare the results of the various methods: weighted TRS, logistic regression, calibrated gradient models, Bayesian networks, and graph-based models.

Compare the UPL-enriched BPCA test selection with the standard BPCA test selection, History-based test selection and full Regression baselines in various industries.

Model release optimization with uncertainty – instead of point estimates for TRS, dependency confidence and test capacity, treat them as distributions.

Research on human adoption, explanation quality, governance acceptance of release managers, change managers, developers, testers, and auditors.

Enhance TRS governance to SAP S/4HANA Clean Core, SAP BTP extension transport management, cloud transport management, API-based integration governance, and hybrid SAP/non-SAP release orchestration.

Table 1: *Evidence-to-TRS grounding across academic and SAP technical sources.*

Evidence domain	Academic mechanism	SAP technical artifact	TRS use
Static analysis	Program analysis, developer adoption	Central ATC findings [30, 31]	ATC delta severity and exemption evidence
Impact analysis	Slicing, dependency analysis	BPCA, TBOM, Solution Documentation [32, 33]	BPCA process criticality and test scope
Runtime usage	Process mining, execution traces	UPL, SCMON, CDMC [36, 37, 38]	Production-usage weighting
Defect prediction	Risk ranking, temporal validation	ChaRM/ITSM incidents, rollback history	Historical incident rate
Release planning	Multi-objective optimization	Transport queue, release calendar	Sequencing, deferral, test-capacity allocation
Governance evidence	Continuous auditing, traceability [A3, A4]	ChaRM, Test Suite, ATC exemptions	TRS evidence package

Table 2: *Anonymized practitioner evidence mapped to TRS design requirements. Client names are withheld; industry and landscape characteristics identify the operational context.*

Implementation context	SAP capabilities observed	Governance gap	Design requirement derived
Large retail landscape	Central ATC, CSOL, Downgrade Protection, release checklist, monitoring dashboards	ATC findings were useful but needed transport-level and business-context weighting	Add ATC Delta Severity and collision-risk factors to TRS
Global multi-sector beverage landscape	ChaRM N/N+1, BPCA, dynamic/semi-dynamic TBOMs, SCMON/UPL, Test Suite	Static impact scope over-scoped low-use paths and under-prioritized high-use paths	Add UPL/SCMON-weighted BPCA and test-priority logic
Regulated pharmaceutical context	BPCA, Test Management, Defect Management, audit-sensitive change control	Audit evidence required cross-transaction manual assembly	Add TRS Evidence Package Generator and evidence-completeness scoring
Food service multi-system landscape (140 SAP + 75 non-SAP systems)	CDMC city model, UPL, custom-code lifecycle analysis, BPMon	Unused or low-use custom code received equal governance attention as high-use objects	Add production-usage weighting and CDMC utilization signals
Semiconductor upgrade context	BPCA upgrade analysis, Test Suite, SOLDOC migration	BPCA outputs and usage evidence were parallel rather than unified	Add unified risk-ranking layer for impacted process steps

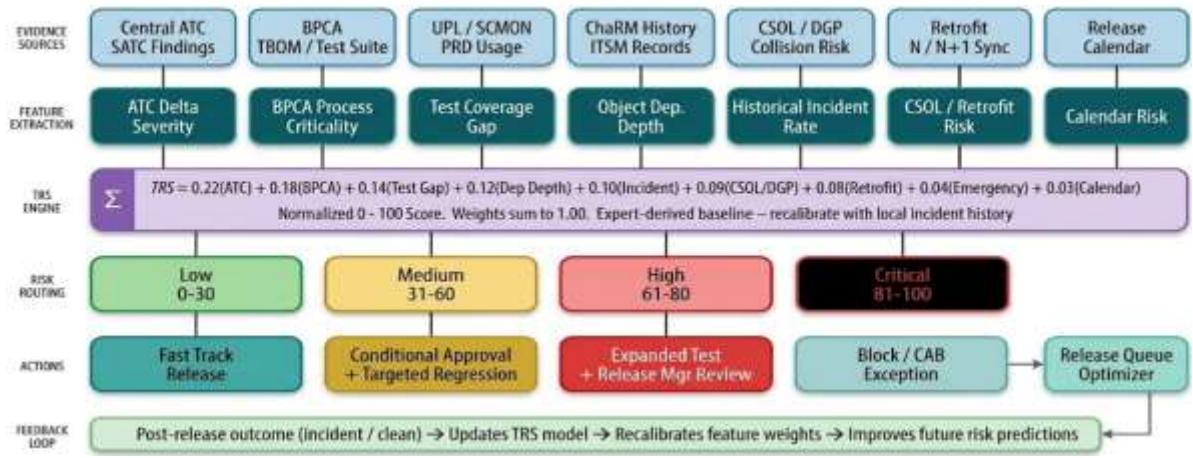


Figure 1: *TRS Predictive SAP Transport Governance Architecture. Evidence sources feed feature extraction, the TRS formula produces a normalized risk score, risk bands route transports to governance actions, and post-release outcomes feedback to recalibrate the model.*

Table 3: *TRS risk bands and recommended governance actions.*

TRS range	Risk band	Recommended governance action
0–30	**Low**	Fast-track release with standard evidence. Standard ChaRM approval chain. BPMon monitoring only. No expanded regression required.
31–60	**Medium**	Conditional approval with targeted BPCA-based regression. Review high-scoring feature contributors. UPL-prioritized test scope. L1 + L2 approval chain.
61–80	**High**	Expanded regression + Release Manager review + dependency check + post-release monitoring period. L1 + L2 + L3 approval. Full BPCA test scope execution required.
81–100	**Critical**	Block transport. Defer, remediate, or obtain explicit CAB or Release Exception approval. Mandatory remediation before resubmission. Post-deployment review required.

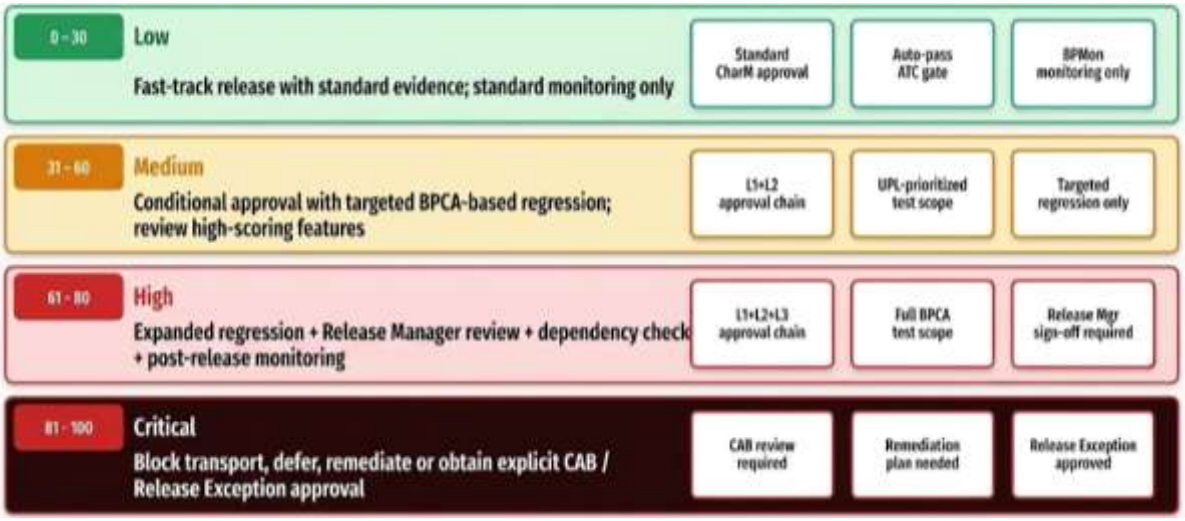


Figure 2: *TRS Risk Bands and Governance Actions. Each band maps to a specific approval chain, test scope requirement, and monitoring obligation within the ChaRM change control framework.*

Table 4*: Feature normalization table for TRS. The nine features are intended to be derived from existing SAP Solution Manager and related ALM evidence sources, minimizing the need for new data-collection infrastructure.*

Feature	Raw SAP source	Normalization method	Output
ATC Delta Severity	Central ATC / SATC [30, 31]	Weighted score for newly introduced high, medium, low findings; inherited findings lower weight; remediated reduce score	0–100
BPCA Process Criticality	BPCA / SOLDOC / Test Suite [32, 33]	Weighted by impacted process category, centrality, compliance sensitivity, and affected transaction importance	0–100
Test Coverage Gap	BPCA + Test Suite / HPALM / CBTA	Impacted tests minus executed and passed tests; automated evidence receives higher confidence than manual placeholder	0–100
Object Dependency Depth	ABAP where-used, TBOM, dep. graph	Number of impacted objects, call depth, package spread, interface touchpoints, indirect dependencies	0–100
Historical Incident Rate	ChaRM, ITSM, defect records	Incident count and severity linked to package/object/release train over a rolling time window	0–100
CSOL/Downgrade Collision Risk	ChaRM Administration Cockpit [35]	Active lock, downgrade flag, overlapping transport, or warning state translated into risk tier	0–100
Retrofit/Release Conflict Risk	N/N+1 transport history [34]	Object overlap, prior conflict frequency, sequence conflict, unresolved Retrofit status	0–100
Emergency Change Factor	ChaRM transaction type	Standard or planned change = 0; urgent/emergency change = 100; optionally scaled by severity	0 or 100
Calendar/Freeze Risk	Release and business calendar	Proximity to period close, freeze, blackout, fiscal reporting, support package, or major cutover window	0–100

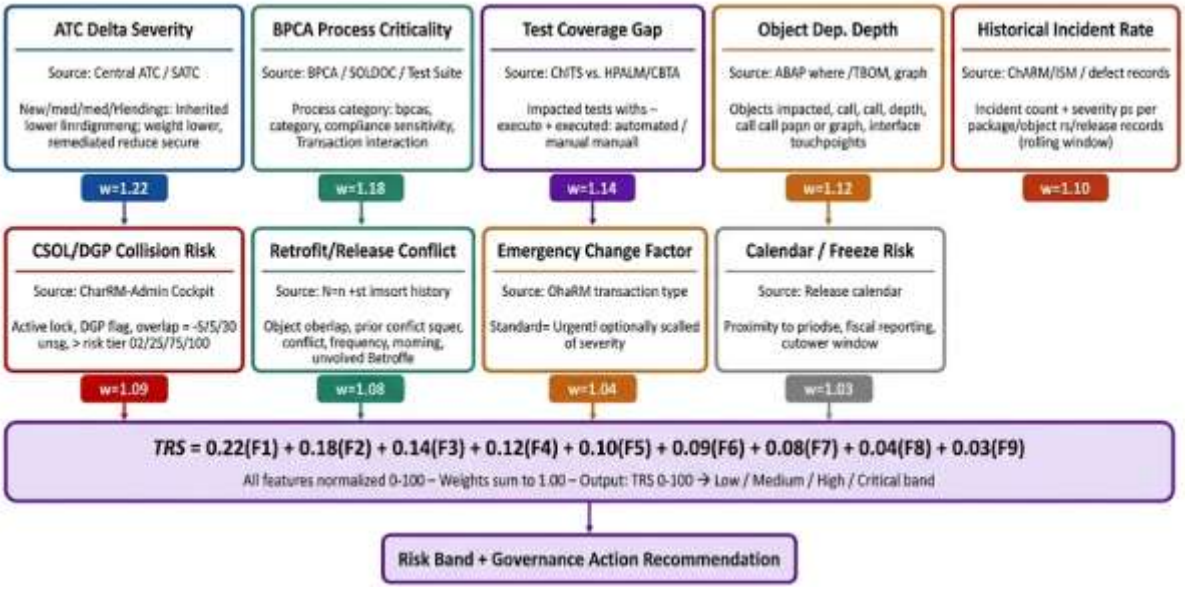


Figure 3: *TRS Feature Pipeline and Normalization Flow. Each of the 9 SAP evidence streams is normalized to a 0–100 scale using the rules in Table 4, then combined using the weighted TRS formula to produce a risk band and governance action recommendation.*

Table 5: *Worked example of TRS computation. TRS = 54.70 → Medium risk band. Recommended action: conditional approval with targeted BPCA-based regression, review of high BPCA process criticality score, and confirmation that calendar/freeze risk is acceptable for the release window.*

Feature	Normalized score	Weight	Weighted contribution
ATC Delta Severity	70	0.22	15.40
BPCA Process Criticality	85	0.18	15.30
Test Coverage Gap	40	0.14	5.60
Object Dependency Depth	60	0.12	7.20
Historical Incident Rate	30	0.10	3.00
CSOL/Downgrade Collision Risk	50	0.09	4.50
Retrofit/Release Conflict Risk	20	0.08	1.60
Emergency Change Factor	0	0.04	0.00
Calendar/Freeze Risk	70	0.03	2.10
Total TRS	**54.70 → Medium Band**		

Table 6: *UPL-enriched BPCA test-scope prioritization logic.*

Test category	Selection rule	Governance purpose
Mandatory	High BPCA impact + high UPL/SCMON usage + critical process	Protect business-critical and frequently used paths — non-negotiable test coverage
Priority	Moderate impact or usage with historical incident linkage	Allocate test effort to historically volatile objects and processes
Optional	Low usage, low criticality, no incident history	Reduce unnecessary regression load when test capacity is constrained without compromising risk coverage
Exploratory	SCMON coverage gap or unknown dynamic path	Address uncertainty where static and runtime evidence diverge — high-unknown-risk paths

Table 7: *Minimum contents of a TRS Evidence Package. The outcome evidence section also serves as the labelled training record for future model iterations.*

Evidence package section	Contents
Transport identity	Transport number, ChaRM document, release cycle, target landscape, change type
Technical evidence	ATC findings, new/inherited/remediated classification, exemption decisions with approver
Process evidence	BPCA impacted process steps, test cases, process criticality classification

Runtime evidence	UPL/SCMON execution frequency, coverage gaps, production caller transaction list
Collision evidence	CSOL locks, Downgrade flags, Retrofit overlap, dependency conflict status
Decision evidence	TRS score, risk band, recommended action, human decision, override reason if applicable
Outcome evidence	Import result, post-release incidents, rollback actions, emergency correction record, monitoring result

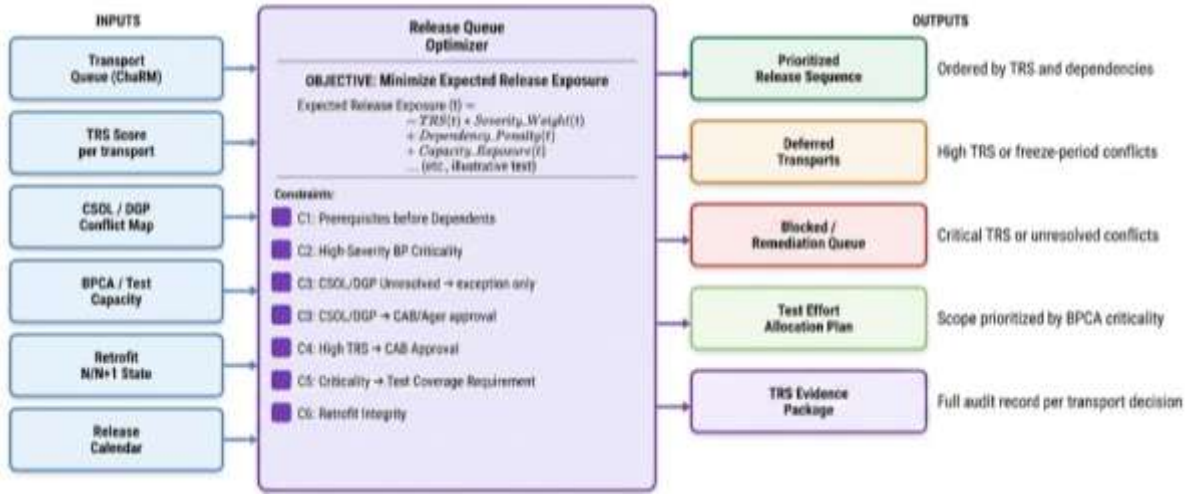


Figure 4: *Risk-Aware Release Queue Optimizer. TRS is one input among dependency constraints, test capacity, release calendar, and rollback difficulty. The optimizer minimizes expected release exposure while satisfying 6 SAP-specific constraints including Retrofit integrity and CSOL/Downgrade resolution requirements.*

Table 8: *Proposed data model for SAP transport-risk datasets.*

Entity	Key fields
Transport	TR number hash, ChaRM document hash, change type, release cycle, status, target landscape, timestamps
Object	Object name hash, object type, package hash, component, owner role, dependency links
ATC Finding	Rule ID, severity, new/inherited/worsened/remediated, exemption status, timestamp
BPCA Impact	Impacted process step, transaction hash, test case hash, criticality, test recommendation
UPL/SCMON	Execution count, caller transaction hash, time window, coverage percentage, production system hash
Test Evidence	Test case ID hash, automation flag, result, defect link, execution timestamp
Incident Label	Incident count, severity, rollback flag, emergency correction, time-to-detect
Release Context	Freeze period, fiscal close proximity, dependency conflicts, release window, rollback complexity
Model Metadata	Model version, training window, feature definitions, thresholds, calibration method

Table 9: *Validation protocol for TRS across 9 dimensions*

Validation area	Metric or method
Risk prediction	Precision, recall, F1, PR-AUC, high-risk recall — not accuracy alone due to class imbalance
Calibration	Brier score, calibration curve, expected calibration error — verify score reflects true probability
Release usefulness	Decision-curve utility, net benefit, release-manager acceptance in controlled field study
False negative risk	High-risk transport leakage rate; incident-causing low-risk releases — the highest-cost error
False positive burden	Unnecessary release blocks, excess regression hours, approval delay — operational cost of over-flagging
Testing benefit	Regression effort saved versus defect coverage retained — UPL-enriched vs. standard BPCA scope
Compliance readiness	Evidence completeness score; missing approval/test/ATC/BPCA/UPL fields per transport

Operations impact	Emergency correction rate, rollback frequency, post-release incident rate — before/after deployment
Fairness and governance	Override frequency by team/package, explanation completeness, model drift over time

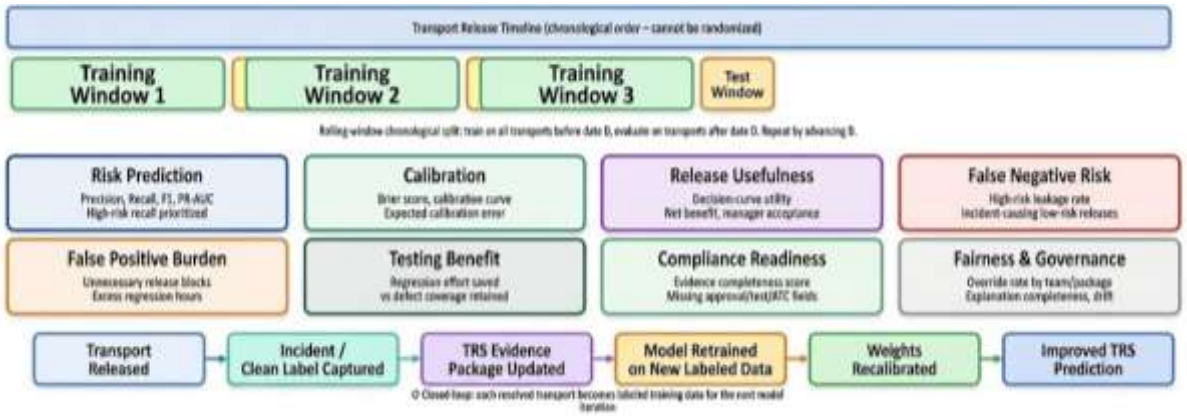


Figure 5: *Temporal Validation Protocol and Closed-Loop Learning. Rolling-window chronological splits prevent temporal leakage. Nine validation dimensions cover prediction quality, release usefulness, and compliance readiness. Post-release outcomes feedback to recalibrate TRS weights in the closed-loop learning cycle.*

15. Conclusions

This paper has introduced TRS, a predictive SAP transport governance model that combines Central ATC, BPCA, UPL/SCMON, Test Suite evidence, ChaRM history, CSOL/Downgrade signals, Retrofit context and release-calendar risk into a normalized transport-level risk score. The model is designed to tackle a real-world problem in SAP release governance where there is rich evidence across SAP tools, but it cannot be natively unified to a predictive, explainable, transport-level score that can enable targeted regression, release sequencing, and audit-ready evidence generation.

The most valuable additions are the feature normalization (Table 4), the weighted risk scoring (Section 5.1), the risk bands and the governance actions (Table 3), the worked example (Table 5), the algorithm (Algorithm 1), the release optimization constraints (Section 7), the dataset structure (Table 8), the validation metrics (Table 9) and the TRS Evidence Package (Table 7). These elements position the paper as a model-driven contribution rather than a literature review alone. It offers a modifiable model that practitioners and researchers in the SAP ALM domain can implement, calibrate, evaluate, challenge and extend.

The goal for the future of SAP transport governance shouldn't be to automate completely. It should be predictive, evidence-based and human accountable. A model that can help enterprises go faster when risk is low, smarter when impact is in question, escalate when risk is high, and retain traceable evidence when compliance is a concern is offered by TRS.

Author Statements:

- **Ethical approval:** The conducted research is not related to either human or animal use.
- **Conflict of interest:** The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper
- **Acknowledgement:** The authors declare that they have nobody or no-company to acknowledge.
- **Author contributions:** The authors declare that they have equal right on this paper.
- **Funding information:** The authors declare that there is no funding to be acknowledged.
- **Data availability statement:** The data that support the findings of this study are available on request from the corresponding author. The data are not publicly available due to privacy or ethical restrictions.
- **Use of AI Tools:** The author(s) declare that no generative AI or AI-assisted technologies were used in the writing process of this manuscript.

References

[1] Oseni, T., Foster, S., Rahim, M., & Smith, S. P. (2017). A framework for ERP post-implementation amendments: A literature analysis. Australasian Journal of Information Systems, 21, 1–21. doi:10.3127/ajis.v21i0.1268

[2] Barth, C., & Koch, S. (2019). Critical success factors in ERP upgrade projects. Industrial Management & Data Systems, 119(3), 656–675. doi:10.1108/IMDS-01-2018-0016

[3] Hasan, N., Miah, S. J., Bao, Y., & Hoque, M. R. (2019). Factors affecting post-implementation success of enterprise resource planning systems: A perspective of business process performance. Enterprise Information Systems, 13(9), 1217–1244. doi:10.1080/17517575.2019.1612099

[4] Lee, N. C. A., & Chang, J. Y. T. (2020). Adapting ERP systems in the post-implementation stage: Dynamic IT capabilities for ERP. Pacific Asia Journal of the Association for Information Systems, 12(1), 28–59. doi:10.17705/1pais.12102

[5] Domagała, A., Grobler-Dębska, K., Wąs, J., & Kucharska, E. (2021). Post-implementation ERP software development: Upgrade or reimplementation. Applied Sciences, 11(11), Article 4937. doi:10.3390/app11114937

[6] Salih, A. A., Zeebaree, S. R. M., Abdulraheem, A. S., Zebari, R. R., Sadeeq, M. A. M., & Ahmed, O. M. (2022). Evolution of enterprise resource planning. IEEE Access, 10, 108004–108020. doi:10.1109/ACCESS.2022.3202954

[7] Lee, C., Kim, H. F., & Lee, B. G. (2024). A systematic literature review on the strategic shift to cloud ERP: Leveraging microservice architecture and MSPs for resilience and agility. Electronics, 13(14), Article 2885. doi:10.3390/electronics13142885

[8] Priyadarsini, A., & Kumar, A. (2022). A literature review on IT governance using systematicity and transparency framework. Digital Policy, Regulation and Governance, 24(3), 309–328. doi:10.1108/DPRG-09-2021-0114

[9] Dutta, A., Roy, R., & Seetharaman, P. (2022). An assimilation maturity model for IT governance and auditing. Information & Management, 59(1), Article 103569. doi:10.1016/j.im.2021.103569

[10] Nguyen Quang Do, L., Wright, J. R., & Ali, K. (2022). Why do software developers use static analysis tools? A user-centered study of developer needs and motivations. IEEE Transactions on Software Engineering, 48(3), 835–847. doi:10.1109/TSE.2020.3004525

[11] Vassallo, C., Panichella, S., Palomba, F., Proksch, S., Gall, H. C., & Zaidman, A. (2020). How developers engage with static analysis tools in different contexts. Empirical Software Engineering, 25(2), 1419–1457. doi:10.1007/s10664-019-09750-5

[12] Trautsch, A., Herbold, S., & Grabowski, J. (2020). A longitudinal study of static analysis warning evolution and the effects of PMD on software quality in Apache open source projects. Empirical Software Engineering, 25, 5137–5192. doi:10.1007/s10664-020-09880-1

[13] Kazmi, R., Jawawi, D. N. A., Mohamad, R., & Ghani, I. (2017). Effective regression test case selection: A systematic literature review. ACM Computing Surveys, 50(2), Article 29, 1–32. doi:10.1145/3057269

[14] Pan, R., Bagherzadeh, M., Ghaleb, T. A., & Briand, L. (2022). Test case selection and prioritization using machine learning: A systematic literature review. Empirical Software Engineering, 27, Article 29. doi:10.1007/s10664-021-10066-6

[15] Singhal, S., Jatana, N., Suri, B., Misra, S., & Fernandez-Sanz, L. (2021). Systematic literature review on test case selection and prioritization: A tertiary study. Applied Sciences, 11(24), Article 12121. doi:10.3390/app112412121

[16] Lima, J. P., & Vergilio, S. R. (2020). Test case prioritization in continuous integration environments: A systematic mapping study. Information and Software Technology, 121, Article 106268. doi:10.1016/j.infsof.2020.106268

[17] Karvonen, T., Behutiye, W., Oivo, M., & Kuvaja, P. (2017). Systematic literature review on the impacts of agile release engineering practices. Information and Software Technology, 86, 87–100. doi:10.1016/j.infsof.2017.01.009

[18] Shahin, M., Babar, M. A., & Zhu, L. (2017). Continuous integration, delivery and deployment: A systematic review on approaches, tools, challenges and practices. IEEE Access, 5, 3909–3943. doi:10.1109/ACCESS.2017.2685629

[19] Lwakatare, L. E., Kilamo, T., Karvonen, T., Sauvola, T., Heikkilä, V., Itkonen, J., Kuvaja, P., Mikkonen, T., & Oivo, M. (2019). DevOps in practice: A multiple case study of five companies. Information and Software Technology, 114, 217–230. doi:10.1016/j.infsof.2019.06.010

[20] Sangwan, R. S., Negahban, A., Nord, R. L., & Ozkaya, I. (2022). Optimization of software release planning considering architectural dependencies, cost, and value. IEEE Transactions on Software Engineering, 48(4), 1369–1384. doi:10.1109/TSE.2020.3020013

[21] Li, N., Shepperd, M., & Guo, Y. (2020). A systematic review of unsupervised learning techniques for software defect prediction. Information and Software Technology, 122, Article 106287. doi:10.1016/j.infsof.2020.106287

[22] Stradowski, S., & Madeyski, L. (2023). Machine learning in software defect prediction: A business-driven systematic mapping study. Information and Software Technology, 155, Article 107128. doi:10.1016/j.infsof.2022.107128

[23] Zhao, Y., Damevski, K., & Chen, H. (2023). A systematic survey of just-in-time software defect prediction. ACM Computing Surveys, 55(10), Article 201, 1–35. doi:10.1145/3567550

[24] Grattan, N., da Costa, D. A., & Stanger, N. (2024). The need for more informative defect prediction: A systematic literature review. Information and Software Technology, 171, Article 107456. doi:10.1016/j.infsof.2024.107456

[25] Verdecchia, R., Engström, E., Lago, P., Runeson, P., & Song, Q. (2023). Threats to validity in software engineering research: A critical reflection. Information and Software Technology, 164, Article 107329. doi:10.1016/j.infsof.2023.107329

[26] Jans, M., & Hosseinpour, M. (2019). How active learning and process mining can act as continuous auditing catalyst. International Journal of Accounting Information Systems, 32, 44–58. doi:10.1016/j.accinf.2018.11.002

[27] Augusto, A., Conforti, R., Dumas, M., La Rosa, M., Maggi, F. M., Marrella, A., Mecella, M., & Soo, A. (2019). Automated discovery of process models from event logs: Review and benchmark. IEEE Transactions on Knowledge and Data Engineering, 31(4), 686–705. doi:10.1109/TKDE.2018.2841877

[28] Martin, N., Fischer, D. A., Kerpedzhiev, G. D., Goel, K., Leemans, S. J. J., Röglinger, M., van der Aalst, W. M. P., Dumas, M., La Rosa, M., & Wynn, M. T. (2021). Opportunities and challenges for process mining in organizations: Results of a Delphi study. Business & Information Systems Engineering, 63(5), 511–527. doi:10.1007/s12599-021-00720-0

[29] Rinderle-Ma, S., Winter, K., & Benzin, J. V. (2023). Predictive compliance monitoring in process-aware information systems: State of the art, functionalities, research directions. Information Systems, 115, Article 102210. doi:10.1016/j.is.2023.102210

[30] Barredo Arrieta, A., Díaz-Rodríguez, N., Del Ser, J., Bennetot, A., Tabik, S., Barbado, A., García, S., Gil-Lopez, S., Molina, D., Benjamins, R., Chatila, R., & Herrera, F. (2020). Explainable artificial intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. Information Fusion, 58, 82–115. doi:10.1016/j.inffus.2019.12.012

[31] Ali, S., Abuhmed, T., El-Sappagh, S., Muhammad, K., Alonso-Moral, J. M., Confalonieri, R., Guidotti, R., Del Ser, J., Díaz-Rodríguez, N., & Herrera, F. (2023). Explainable artificial intelligence (XAI): What we know and what is left to attain trustworthy artificial intelligence. Information Fusion, 99, Article 101805. doi:10.1016/j.inffus.2023.101805

[32] Hassija, V., Chamola, V., Mahapatra, A., Singal, A., Goel, D., Huang, K., Scardapane, S., Spinelli, I., Mahmud, M., & Hussain, A. (2024). Interpreting black-box models: A review on explainable artificial intelligence. Cognitive Computation, 16(1), 45–74. doi:10.1007/s12559-023-10179-8

[33] Kreuzberger, D., Kühl, N., & Hirschl, S. (2023). Machine learning operations (MLOps): Overview, definition, and architecture. IEEE Access, 11, 31866–31879. doi:10.1109/ACCESS.2023.3262138

[34] Birkstedt, T., Minkkinen, M., Tandon, A., & Mäntymäki, M. (2023). AI governance: Themes, knowledge gaps and future agendas. Internet Research, 33(7), 133–167. doi:10.1108/INTR-01-2022-0042

[35] Mäntymäki, M., Minkkinen, M., Birkstedt, T., & Viljanen, M. (2022). Defining organizational AI governance. AI and Ethics, 2, 603–609. doi:10.1007/s43681-022-00143-x

[36] Werder, K., Ramesh, B., & Zhang, S. (2022). Establishing data provenance for responsible artificial intelligence systems. ACM Transactions on Management Information Systems, 13(2), Article 22, 1–23. doi:10.1145/3503488

[37] Mehrabi, N., Morstatter, F., Saxena, N., Lerman, K., & Galstyan, A. (2021). A survey on bias and fairness in machine learning. ACM Computing Surveys, 54(6), Article 115, 1–35. doi:10.1145/3457607

[38] Mittelstadt, B. (2019). Principles alone cannot guarantee ethical AI. Nature Machine Intelligence, 1(11), 501–507. doi:10.1038/s42256-019-0114-4

[39] Hogan, A., Blomqvist, E., Cochez, M., d'Amato, C., de Melo, G., Gutierrez, C., Kirrane, S., Gayo, J. E. L., Navigli, R., Neumaier, S., Ngonga Ngomo, A.-C., Polleres, A., Rashid, S. M., Rula, A., Schmelzeisen, L., Sequeda, J., Staab, S., & Zimmermann, A. (2021). Knowledge graphs. ACM Computing Surveys, 54(4), Article 71. doi:10.1145/3447772

[40] Ji, S., Pan, S., Cambria, E., Marttinen, P., & Yu, P. S. (2022). A survey on knowledge graphs: Representation, acquisition, and applications. IEEE Transactions on Neural Networks and Learning Systems, 33(2), 494–514. doi:10.1109/TNNLS.2021.3070843

[41] Tamašauskaitė, G., & Groth, P. (2023). Defining a knowledge graph development process through a systematic review. ACM Transactions on Software Engineering and Methodology, 32(1), Article 27, 1–40. doi:10.1145/3522586

[42] Wang, L., Sun, Z., Zhang, Y., Nie, L., & Huang, Y. (2023). Application of knowledge graph in software engineering field: A systematic literature review. Information and Software Technology, 164, Article 107327. doi:10.1016/j.infsof.2023.107327

[43] Charalampidou, S., Ampatzoglou, A., Karountzos, E., & Avgeriou, P. (2020). Empirical studies on software traceability: A mapping study. Journal of Software: Evolution and Process, 32(12), Article e2294. doi:10.1002/smr.2294

[44] Mucha, J., Kaufmann, A., & Riehle, D. (2024). A systematic literature review of pre-requirements specification traceability. Requirements Engineering, 29, 119–141. doi:10.1007/s00766-023-00412-z

[45] Lyu, Y., Cho, H., Jung, P., & Lee, S. (2023). A systematic literature review of issue-based requirement traceability. IEEE Access, 11, 13334–13348. doi:10.1109/ACCESS.2023.3242294

[46] Aftab, M. U., Qin, Z., Hundera, N. W., Ariyo, O., Zakria, Son, N. T., & Dinh, T. V. (2019). Permission-based separation of duty in dynamic role-based access control model. Symmetry, 11(5), Article 669. doi:10.3390/sym11050669

[47] Atlam, H. F., Azad, M. A., Alassafi, M. O., Alshdadi, A. A., & Alenezi, A. (2020). Risk-based access control model: A systematic literature review. Future Internet, 12(6), Article 103. doi:10.3390/fi12060103

[48] Huang, Y., Li, W., Macheret, F., Gabriel, R. A., & Ohno-Machado, L. (2020). A tutorial on calibration measurements and calibration models for clinical prediction models. Journal of the American Medical Informatics Association, 27(4), 621–633. doi:10.1093/jamia/ocz228

[49] Singh, K., Shah, N. H., & Vickers, A. J. (2023). Assessing the net benefit of machine learning models in the presence of resource constraints. Journal of the American Medical Informatics

Association, 30(4), 668–673. doi:10.1093/jamia/ocad006

[50] Mishra, L. N., Rangari, A., Nagrare, S., & Nayak, S. K. (2026). A taxonomy for detecting and preventing temporal data leakage in machine learning-based build prediction: A dual-platform empirical validation. PLOS ONE, 21(5), Article e0340167. doi:10.1371/journal.pone.0340167

[51] Baskerville, R., Baiyere, A., Gregor, S., Hevner, A., & Rossi, M. (2018). Design science research contributions: Finding a balance between artifact and theory. Journal of the Association for Information Systems, 19(5), 358–376. doi:10.17705/1jais.00495

[52] Gheibi, O., Weyns, D., & Quin, F. (2021). Applying machine learning in self-adaptive systems: A systematic literature review. ACM Transactions on Autonomous and Adaptive Systems, 15(3), Article 9, 1–37. doi:10.1145/3469440

[53] Weyns, D., Gerostathopoulos, I., Abbas, N., Andersson, J., Biffl, S., Brada, P., Bures, T., Di Salle, A., Galster, M., Lago, P., Lewis, G., Litoiu, M., Musil, A., Musil, J., Patros, P., & Pelliccione, P. (2023). Self-adaptation in industry: A survey. ACM Transactions on Autonomous and Adaptive Systems, 18(2), Article 5, 1–44. doi:10.1145/3589227

[54] Wong, T., Wagner, M., & Treude, C. (2022). Self-adaptive systems: A systematic literature review across categories and domains. Information and Software Technology, 148, Article 106934. doi:10.1016/j.infsof.2022.106934

[55] Durelli, V. H. S., Durelli, R. S., Borges, S. S., Endo, A. T., Eler, M. M., Dias, D. R. C., & Guimarães, M. P. (2019). Machine learning applied to software testing: A systematic mapping study. IEEE Transactions on Reliability, 68(3), 1189–1212. doi:10.1109/TR.2019.2892517