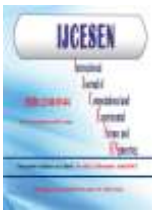
International Journal of Computational and Experimental Science and ENgineering (IJCESEN)

Vol. 11-No.1 (2025) pp. 1666-1682
http://www.ijcesen.com



ISSN: 2149-9144

Research Article

Autonomous SAP ALM Control Fabric (ASACF): A Practitioner-Grounded AI Governance Architecture for Change, Compliance, and Release Management Across Hybrid Enterprise Landscapes

Karthik Paramasivam*

PSG College of Technology, Peelamedu, Coimbatore, Tamil Nadu 641004, India
* **Corresponding Author Email:** karthik.sapalm@gmail.com - **ORCID:** 0000-0002-5247-6650

Article Info:

DOI: 10.22399/ijcesen.5504
Received : 05 February 2025
Revised : 25 March 2025
Accepted : 27 March 2025

Keywords

SAP ALM;
SAP ChaRM;
Autonomous ALM;
SAP Solution Manager;
Central ATC;
BPCA

Abstract:

Enterprise SAP landscapes have shifted from the rather centralized ERP landscape to a hybrid enterprise landscape that integrates SAP S/4HANA, SAP Solution Manager, SAP Cloud ALM, SAP Business Technology Platform extensions, custom ABAP and CDS developments, Fiori applications, integration middleware, analytics services, identity controls and third-party service management platforms. In such contexts, Application Lifecycle Management (ALM) transcends mere coordination between development and transport releasing tasks. It's a governance issue where there is a need to consider the change evidence, technical quality, business-process impact, authorization integrity, testing assurance, audit requirements and operational telemetry in tandem, before a decision can be deemed safe to release. There are various existing SAP ALM practices that create a considerable amount of evidence, such as Change Request Management (ChaRM), Central ATC, Business Process Change Analyzer (BPCA), SCMON, UPL, Test Suite, Cross System Object Lock (CSOL), Downgrade Protection, Retrofit and Focused Insights and incident management. These evidence sources can, however, be dispersed throughout workflows, repositories and governance roles. This fragmentation hampers release explainability and leaves behind long-lasting gaps in change classification, routing for approval, conflict prevention and compliance proof for retrofits. This paper suggests a literature inspired and practitioner-driven governance structure for AI-assisted SAP change, compliance and release management, called the Autonomous SAP ALM Control Fabric (ASACF). The framework is developed using design-science reasoning by synthesizing research on ERP governance, continuous auditing, process mining, DevOps, software analytics, explainable AI, MLOps, and self-adaptive systems, together with anonymized practitioner design evidence from enterprise SAP ChaRM landscapes. ASACF comprises seven layers: Change Evidence Collection, SAP Dependency Knowledge Graph, AI Risk Intelligence, Governance Policy Engine, Release Decision Layer, Audit Evidence Layer and Continuous Learning Layer. The paper also introduces four AI governance services built on ChaRM: a change type classification engine, a risk adaptive approval routing engine, a predictive retrofit conflict engine and a continuous compliance evidence assembler. The core idea is a bounded-autonomy model in which AI complements, rather than takes the place of, SAP governance authority, making recommendations, providing evidence completeness checks, routing according to the policy and enabling auditable human override. The framework provides a reference architecture and an evaluation roadmap for future empirical study of autonomous SAP ALM that can be reused.

1. Introduction

Enterprise software environments are becoming increasingly hybrid, distributed, and governance-aware. These environments now span SAP S/4HANA, SAP Solution Manager, SAP Cloud ALM, custom ABAP and CDS artifacts, Fiori applications, identity and access controls, integration middleware, process monitoring, analytics, third-party IT service management platforms, and cloud extensions. A single enterprise change can include a transport, a configuration

change, workflow rules, authorization change, interface mapping, test package, process documentation node and production monitoring object. If the only way to know if the risk is present is whether or not the code compiles or whether or not a transport was transferred from development to quality assurance, then that is not the right way of determining the risk of release. A governance question is more general: What is the rationale for the change in terms of its technical safety, process-awareness, compliance, testing, approval and operational recoverability?

Several mechanisms are already available in SAP ALM tooling to support this form of governance. ChaRM has an auditable change workflow with a controlled change state transition. Central ATC provides quality checking evidence of ABAP code and repository objects in the SAP native format. The Change-impact, test-scope and regression-evidence mechanisms are provided in BPCA and SAP Solution Manager Test Suite. Retrofit, CSOL and Downgrade Protection allow for dual-landscape synchronization and control of transport conflicts, and UPL and SCMON usage logging allow for runtime usage evidence for ABAP objects [50-55]. Typically, these mechanisms are used as fragmented evidence sources rather than as components of a unified release-governance intelligence layer.

This paper argues that the next stage of SAP ALM does not require automation alone, but governed automation. Governed autonomy: Evidence is collected by AI and analytical models, change impact analysis, risk score, policy gap identification, release actions are suggested based on evidence, and learning from evidence and accountability for decision rights and audit evidence are still well defined. This especially applies to systems that are based on SAP, since numerous changes affecting financial reporting, execution of the supply chain, regulated manufacturing, procurement, master data and authorization controls etc. have an impact. Neither fully automated deployment nor a purely manual governance would make sense with enterprise control requirements, and a purely manual governance is becoming more and more too slow and fragmented in hybrid environments.

This paper proposes a framework for Autonomous SAP ALM Control Fabric (ASACF) – an AI governance framework based on practitioners' experience in SAP change, compliance and release management. ASACF aims to integrate evidence generated by the tools of SAP ALM in a multi-layered decision-making process. It adds an SAP Dependency Knowledge Graph (SDKG) to relate changes, transports, objects, processes, roles, tests, incidents and controls. It also specifies four operational gaps with respect to ChaRM that are current concerns with AI governance services: ITSM ticket classification to change types, Routing to Approval chain that doesn't evolve over time, Reactive retrofit conflict detection and Fragmented compliance evidence assembly.

How can enterprise SAP organizations design AI-assisted ALM architectures that integrate technical, process, compliance, and operational evidence while supporting explainable, auditable, and bounded release governance in hybrid environments? The remaining of this paper is organized as follows: The relevant research streams are discussed in Section 2. A research method and design evidence that are discussed in section 3. In Section 4, the architecture of ASACF is presented. Section 5 defines the SAP Dependency Knowledge Graph and explains its role in release-risk reasoning. In this section, you will learn about the four AI governance services available on ChaRM. Bounded autonomy: decision model is presented in Section 7. Evaluation and Research Propositions are included in Section 8. Implementation issues are covered in Section 9. In Section 10 a governance maturity model is presented. The discussion, limitation, future research and conclusion are in sections 11-14 following.

2. Literature Review

2.1 Enterprise systems and SAP change governance

There has always been a strong focus on ERP as more than just a system of applications, and a focus on ERP as an infrastructure that brings together processes, data structures, decision rights and operational routines. Although integration and standardization and cross-functional data consistency are considered core features of ERP, foundational ERP studies indicate that these features are important to obtain but are not yet understood as the main reasons for long-term value, as revealed by implementation research, which points to subsequent ERP system adaptations, organizational fit and process governance as sources of long-term value [1-8]. The results of this research are relevant to SAP ALM as transport requests and configuration changes are not just technical artifacts. They change the 'process infrastructure' that the enterprise uses to run finance, procurement, manufacturing, sales, logistics and reporting processes.

The other finding from ERP customization research is that customization might be required for fit with the organization, but it makes maintenance and

upgrading more complex [5-7]. This complexity manifests itself in SAP landscapes in the form of custom ABAP objects, user exits, enhancement implementations, forms, workflow rules, interface mappings, authorization objects and process variants. Such changes have a release consequence, which a governance architecture considering only source code quality cannot account for. Multi-layer evidence is thus required to link technical changes to process execution and control exposure, which is why SAP ALM is necessary.

2.2 IT governance, role-based controls and separation of duties

The IT governance literature is centred around rights and accountability decision making, control and business-IT alignment [9-12]. SAP ChaRM implements these concepts in transaction types, partner functions, status profiles, approval roles, change cycles and task lists. However, governance research also suggests that a lack of flexibility in the organisation can be too late and too little, as the pace of change accelerates, the complexity of the landscape grows and there is increased risk of regulation. The rights to make a decision have to be related to risk, materiality and the presence of evidence.

To grant permissions formally based on roles rather than on persons [13-15] we make use of role based access control. The change requester, developer, tester, change manager, release manager and functional lead roles are similar in SAP ALM. However, the traditional RBAC and Partner function assignment doesn't guarantee a semantic safety or process compliance of a specific change. A user can approve a transport, yet the transport is still causing an impact: high risk process impact or authorization conflict. The proposed ASACF model is thus a combination of role authority and risk-aware evidence and policy evaluation.

2.3 Continuous auditing, process mining and compliance evidence

The literature on continuous auditing suggests the move from retrospective sampling to near real-time monitoring through analytics [16,17]. There are methods in process mining research for discovering process behaviour from event logs, replaying it, and checking it [18-22]. Compliance-monitoring research also demonstrates the possibility to assess business rules and process models via execution traces [23,24]. These streams put forward the idea that the evidence of ALM should not be built up after an audit request but should be built on a continuous basis as there are changes in the fact.

With SAP ALM, the compliance challenge is not just about having an approval or not. It also contains whether the appropriate change type was used, whether the change was made in accordance with separation of duties, whether there are any tests performed, whether the change was a production import within a valid change window, and whether the change is related to a material control, with the results of these questions designated Yes or No. ASACF considers compliance evidence assembly to be a first-class architectural layer as opposed to a manual audit activity.

2.4 DevOps, software analytics and SAP release intelligence

Research in DevOps and continuous delivery has led to greater understanding of the benefits of automation, feedback, deployment pipelines and release flow [25-28]. The following regression test selection, defect prediction, just in time quality assurance and automated repair techniques have been developed by software analytics research: [29-35]. Each of these fields shows that history of the repository, change metrics, dependency structure, test results and incident records can contribute to a preliminary assessment of risk. But generic DevOps is not the same as SAP ALM. The cross-client configuration and transport sequencing, process variants, authorization controls, business criticality and audit evidence are all essential features that need to be considered in SAP release governance.

The SAP-specific challenge is not just about getting it automated. It is how to determine when the transport can move forward, what evidence is adequate, what approvals are necessary, which tests are needed, whether there is any possible object conflicts, and how the outcomes of the post-release should influence future thresholds for governance. ASACF extends software analytics to the governed enterprise-change context, through policy, process and audit layer coupling.

2.5 Explainable AI, MLOps and bounded autonomy

A few concepts that are highlighted in the literature on MLOps are versioning, deployment discipline, monitoring, data lineage, and feedback control [36]. Explainable AI (XAI) research attempts to characterize interpretable models, post-hoc explanations, explanations by features, rule extraction, and local explanations, and cautions against the use of black-box models in high-stakes situations [37-39]. Responsible AI research also demonstrates that general moral guidelines are not

enough without being institutionalized in mechanisms, operational controls and accountability regimes [40,41].

These warnings are at the heart of AI supported SAP ALM. Any decision that denies a transport, elevates the approval or risk assessment, must have a sound reason for the decision and must be explained to developer(s), release manager(s), auditor(s) and owner(s) of the process. It's not just enough to have a generic confidence score. Explanations have to specify the object, process, test gap, control rule and incident pattern or dependency path that affected the recommendation. For this reason, ASACF prefers bounded autonomy; the analytical models can be used to facilitate the release governance, even though the rules of override and audits are still clearly defined.

2.6 Self-adaptive systems and closed-loop ALM

The field of self-adaptive systems research (autonomic computing, control-loop architectures) offers a valuable conceptual framework for autonomous ALM [42-45]. Runtime outcomes, monitoring, analysis, planning and execution provide a basis for what enterprise systems could learn. The work in traceability research also highlights the need to know how requirements, design, implementation, and runtime evidence are related [46,47]. Micro services and cloud architecture research reveal the growing reliance of modern enterprise applications on distributed services and external dependencies [48,49].

ASACF explores this concept further in the context of SAP ALM and suggests a closed loop: evidence gathering, dependency graphing, risk analysis, governance policy, Release Action decision, audit evidence capturing and learning from outcomes. The closed-loop design does not eliminate the role of humans in governance. It shifts humans from assembling evidence to adjudicating exceptions, refining policies, and validating models and holding accountability for control actions.

2.7 SAP official documentation as technical grounding

The peer-reviewed literature is limited and so this paper uses the peer-reviewed literature to create a theoretical understanding of the involved mechanisms of governance, compliance monitoring, process mining, explainable AI, software analytics and self-adaptive control associated with SAP ChaRM, Central ATC, BPCA, Test Suite, CSOL, Downgrade Protection, Retrofit, UPL and SCMON. Technical interpretation of SAP-specific ALM components (ABAP Test Cockpit quality checking, Business Process Change Analyzer impact analysis, SAP Solution Manager Test Suite, and dual-landscape synchronization via Retrofit, Administration Cockpit functions related to CSOL and Downgrade Protection, and usage logging via UPL/SCMON [50-55]) is only performed using SAP official documentation. This separation enables the framework to be academically grounded while at the same time being representative of the mechanisms of implementation of SAPs.

3. Research Methodology and Practitioner-Grounded Design Evidence

This research is conducted in the design science research approach. The aim is to develop a reusable governance artifact - the ASACF architecture - for a relevant class of organizational problems in enterprise SAP ALM. The artifact is structured around a synthesis of peer-reviewed research to provide a broad overview of the field of enterprise ChaRM implementation and design evidence from the practitioners that have used it to date, which has been anonymized. The results presented in this paper are not the result of a controlled experiment or a statistical field trial. Rather, it proposes an architecture, design requirements, governance services and evaluation agenda which can be empirically tested in future work.

There are 5 steps in the design process. As for the problem domain, it is determined as SAP change, compliance and release governance in hybrid landscapes. Second, research streams are aligned with ALM functions: ERP governance, access control, continuous auditing, process mining, DevOps, software analytics, explainable AI, MLOps, traceability and self-adaptive systems. Third, governance gaps that practitioners observe in the practice of enterprise ChaRM are distilled into design requirements based on their experience with enterprise ChaRM implementation. Fourth, the architecture of the ASACF and the four AI governance services are described. Fifth, evaluation is specified and research propositions are proposed for future validation.

Practitioner evidence has been anonymized. It contains repeated implementation patterns across the SAP Solution Manager and ChaRM landscapes (such as 3-to-8-system, N and N+1 production support/project development, major ChaRM transaction types, CSOL, Downgrade Protection, enhanced Retrofit, Central ATC, BPCA, ChaRM and external ITSM integration with platforms like Remedy, ServiceNow, and Salesforce, Test Suite, Focused Insights, HPALM integration). Client-identifying information and proprietary process data are omitted to keep the manuscript suitable for publication, while maintaining domain specificity.

4. Autonomous SAP ALM Control Fabric (ASACF)

The main contribution for this paper is the Autonomous SAP ALM Control Fabric (ASACF). ASACF is a multi-layered governance structure where SAP ALM evidence is incorporated into decision-making processes for releases which are explainable and auditable. The Add-On is not a substitute for ChaRM, Central ATC, BPCA, CSOL, Downgrade Protection, Retrofit or Test Suite. Rather, it allows them to build an intelligence layer that is able to correlate their evidence and use policy-aware decision logic.

4.1 Layer 1: Change Evidence Collection

The first layer is used to collect the evidence that has been created during the SAP change lifecycle. Evidence sources include ChaRM change documents, transport request, Central ATC findings, BPCA impact result, SCMON and UPL usage records, Test Suite execution result, Focused Insights dashboards, incident records, approval logs and production monitoring events. These artifact classes are based on official SAP documentation at the product-function level: ATC is used for quality checking, BPCA and Test Suite are used for impact analysis and test planning, Retrofit is used for dual-landscape synchronization, Administration Cockpit functions are used for CSOL/Downgrade Protection control, and UPL/SCMON is used for usage logging [50-55].

4.2 Layer 2: SAP Dependency Knowledge Graph

The second layer translates evidence to a graph of relationships. The nodes are SAP lifecycle artifacts like change documents, transport requests, ABAP objects, CDS views, Fiori applications, configuration items, interfaces, authorization roles, process steps, test cases, incident, controls and release packages. Edges represent relationships such as contains, calls, impacts, tests, violates, depends on, approved by, imported to and co-changes with. This graph allows impact reasoning which is an explainable risk; the release manager can now see not only that a change is high risk, but which dependency path gives rise to the risk.

4.3 Layer 3: AI Risk Intelligence

The AI Risk Intelligence layer calculates technical risk, process risk, compliance risk, operational risk and retrofit risk. May employ interpretable classifiers, rules, calibrated scores, anomaly detection, and graph-based queries. The ASACF has to provide explanations for all material recommendations due to audit-sensitive nature of SAP Release Governance. Useful explanation refers to the specific transport object, ATC finding, process criticality, CSOL conflict, test coverage gap, and/or prior incident cluster/policy rule involved.

4.4 Layer 4: Governance Policy Engine

The Governance Policy Engine offers release rules, evidence requirements, SoD restrictions, emergency-change controls, control materiality thresholds and escalation policies. This layer is between the governance policy and model prediction. The policy engine should translate model-estimated risk into governance actions by applying evidence requirements, approval rules, release-window constraints, and exception policies. For instance, if all the functional tests are successful, but the financial posting change occurs in a high-risk period near the end of the period the functional lead approval and post-deployment monitoring may be required.

4.5 Layer 5: Release Decision Layer

The Release Decision Layer takes evidence and policy evaluation and translates it into action. Some options are auto-approve for low-risk and complete-evidence change, conditional approve with extra testing, escalate to change advisory review, block release until remediation is completed, or increased post-release monitoring. The decision layer is a bounded autonomy layer, where AI-generated recommendations remain auditable and may influence, but not independently determine, material release decisions.

4.6 Layer 6: Audit Evidence Layer

Audit Evidence Layer contains decision logic, evidence snapshots, versions of the model, model approvals, overrides, evidence test records, and transport import records and exception justifications. The layer is important because decisions made for releasing a product or service in SAP governance need to be explainable afterwards. It also facilitates ongoing auditing, as evidence of controls can be created from events during the operations rather than during audit.

4.7 Layer 7: Continuous Learning Layer

The Continuous Learning Layer pushes production incidents and rollback events, audit results, failed

tests, emergency fixes, drift indicators and human override results back into the risk and policy layers. Learning is governed: Updates to the model, thresholds, and policy changes are also controlled changes with lineage, approval and rollback mechanisms. This way, the ALM intelligence layer isn't an uncontrolled second order risk.

5. SAP Dependency Knowledge Graph Model

ASACF is built on an architectural foundation called the SAP Dependency Knowledge Graph (SDKG). A workflow is used to create status transitions. A graph helps to explain the reason for a change. With SAP landscapes comes risk that often spans across artifacts: An ABAP object may be locked by another change document; the transaction may be part of a critical process that is covered by a manual test; the same object may be required for a transaction with another critical process covered by a manual test; the same transaction may be relevant to a role that is subject to SoD; and the same transaction may be near a financial close window. This chain of meaning is difficult to represent in a tabular change record. A dependency graph can represent these cross-artifact relationships more effectively than a tabular change record.

The SDKG should not be considered merely a technical dependency graph, but a governance knowledge base. It is a mixture of static relationships, historical relationships and policy relationships. Object containment and call dependencies are examples of static relationships. Common historical relationships are co-change, incident association and incident rollback patterns. SoD relationships involve the relevance of SoD and ownership of the control and the necessary approval processes. The graph can be used for automatic queries and explanation in human-readable format.

6. Four AI Governance Services Over SAP ChaRM

The ASACF architecture is made operational by four AI governance services that are stacked on top of the existing ChaRM architecture. These services are based on recurring practitioner-observed gaps and are intended to complement, not replace, SAP change workflows.

6.1 AI Change Type Classification Engine

ChaRM follow-up documents are often created or triggered by external ITSM platforms like Remedy, ServiceNow and Salesforce. The key decision in practice at this boundary is that of the type of change to choose. It may have to be converted to a Request for Change, Normal Change, Urgent Change, Admin Change, Standard Change, Defect Correction or Retrofit Change. Misclassification can create a control risk. For example, a production-impacting correction incorrectly classified as a low-risk standard change may bypass the approval rigor required for urgent or emergency changes.

The history of ITSM tickets, their text, priority, affected module, category, incident linkage and resulting ChaRM transaction type are used as training evidence by the AI Change Type Classification Engine. It classifies each incoming ticket into the most probable ChaRM change type and assigns a confidence level. Some classifications can be used to auto-populate the change document, and others are sent to a change manager to verify. The service should also provide explanation including the ticket phrases, affected process, priority indicators and historical examples that affected the recommendation.

6.2 Risk-Adaptive Approval Routing Engine

For many ChaRM workflows, approvals are made by transaction type and status, instead of based on material risk. ASACF recommends a Risk-Adaptive Approval Routing Engine with variable approval criteria that take into account transport evidence. The inputs are ATC severity, object criticality, authorization relevance, CSOL/Downgrade Protection status, affected process criticality, test coverage, incident history, emergency status and release timing [50,54]. The result is an approval path that is aligned with risk.

If the low-risk transport has total test evidence and there is no impact on the process control, then it can be processed under the standard approval route. Functional lead approval, security review, release manager sign-off and post-deployment monitoring may be necessary on a transport that impacts financial postings, tax logic, goods movement, payment processing or critical interfaces. The purpose is not to eliminate governance effort but to distribute governance effort in the places where it will make the most assurance value.

6.3 Predictive Retrofit Conflict Engine

For dual-track SAP landscapes, there is a need to synchronize the two streams of production and project support. Retrofit is described in SAP documentation as a way to synchronize both landscapes and Administration Cockpit functions for managing Cross-System Object Lock and Downgrade Protection entries [53,54] are

described. But there are many organizations that still deal with reactive conflict resolution once the transport sequencing decision is made. ASACF proposes a Predictive Retrofit Conflict Engine that can predict the probability of conflicts before release of the transport.

Engine features include object overlap between N and N+1 landscapes, historical co-modification frequency, active change documents, developer activity, package ownership, CSOL lock signals, Downgrade Protection warnings, and previous retrofit outcomes. If the risk is high, the engine issues a Retrofit Risk Alert on the change document. The alert will highlight conflicting objects, related N+1 changes, likely conflict type and possible actions (sequencing, bundling, deferral, or joint developer review). The retrofit governance is shifted from "after the fact" remediation to a proactive approach of release planning.

6.4 Continuous Compliance Evidence Assembler

In SAP environments that are regulated and/or audit sensitive, proof of changes requested, approved, tested, imported and reviewed per SAP policy is necessary. Evidence can be found in the document history of ChaRM, transport logs, Test Suite records, approval records, access reviews and incident management tools, but isn't always coupled into an audit-ready document. SAP Solution Manager Test Suite and BPCA documentation provide the technical grounding for the testing and impact-analysis evidence used in this layer [51,52]. ASACF is thus supported by a Continuous Compliance Evidence Assembler.

The assembler records requester identity, source ticket reference, approval chain, SoD checks, evidence of testing, transport numbers, import times, production confirmation, exceptions, override justifications, and post-release incidents, for each change document. The result is a Change Governance Evidence Certificate, by change document, transport, release package, control and fiscal period. The service must also create dashboards for missing evidence, SoD violations, overdue evidence approvals, evidence changes imported without evidence, and evidence changes that need to be reviewed after implementation is done.

7. Bounded Autonomy Release Decision Model

ASACF is intentionally not a model of full autonomous production deployment. The release decisions of Enterprise SAPs are material governance decisions. The right design is bounded autonomy: analytical models are free to collect evidence on their own, route cases for further investigation, raise exceptions and escalate cases – but are not permitted to make exceptions without notification, without policy review or override human decision.

Use of calibrated risk scores and policy thresholds should be used for the decision model, not just raw model confidence. If a high capacity black-box model is not accompanied by strong explanations and validation evidence, it may be better to have a transparent rule-based or interpretable model in high accountability settings. Human override is not a bug, it's a governance feature. All overrides, though, should be recorded with a reason, role, date and next outcome, to allow for the Continuous Learning Layer to differentiate between exceptions created by good reason and drift by governance.

8. Evaluation Framework and Research Propositions

As a proposed architecture, ASACF should be evaluated using a multi-dimensional scale, not to a single metric of accuracy. A model with a higher accuracy rate that generates more exceptions in the audit might not be acceptable in SAP ALM. Likewise, a test-selection algorithm that saves in terms of test effort but misses key process variants could be bad. Evaluation should take into account technical, operational, process and compliance outcomes.

9. Implementation Roadmap

ASACF can be done gradually. There is no need for all layers to be deployed at the same time. A roadmap helps decrease the risk of adoption and helps governance teams validate value before increasing autonomy.

10. ASACF Governance Maturity Model

Beyond the implementation roadmap, ASACF can also serve as a maturity model to measure the extent to which an organisation is already on the road to governed autonomous ALM from manual change control. The value of the maturity model for both research and practice is that it clears the path between the “mere presence” of tools and the “governance intelligence” they provide. Even if an organization has ChaRM, ATC, BPCA and Test Suite in place, it could still be at a low maturity level if the evidence is disorganized, approvals are linear and audit trails are manually built.

The maturity model also helps to explain the path that is expected for the adoption. ASACF is not

intended to impose immediate autonomy across all governance levels; Level 5 should be treated as an aspirational maturity state rather than a mandatory target. For a big organisation, evidence completeness may be the first step, followed by the creation of the SDKG, automation of compliance evidence and finally, the addition of AI-assisted routing or retrofit prediction. This is a staged progression that is useful in audit-sensitive environments where each autonomy capability can be evaluated prior to using it for material release decisions.

11. Discussion

ASACF shifts SAP ALM from a workflow-execution paradigm toward a governance-intelligence paradigm. The traditional ChaRM workflows are required since they provide the structure of the change states, approvals, task lists, and transport movement. The change itself and workflows alone cannot predict if the change is low risk, if evidence is complete, if there is a process-control impact, or if a future retrofit conflict is likely. ASACF does not take the place of ChaRM, it is the backbone of the governed process and provides an evidence, graph, AI and policy fabric around it.

The design evidence is practitioner-focused because many governance issues with SAP are only apparent at enterprise-level. Errors in the ITSM classification, fatigue in the approval-chain and N/N+1 rework for retrofit, manual evidence assembly may seem more of an operational than a scientific process. The problems, however, indicate other research gaps: there is no semantic change classification, there is no risk-proportionate governance routing, there is no predictive cross-landscape conflict reasoning, and there is no continuous assurance integration. ASACF uses these observed issues to generate design propositions that can be tested.

The framework also explains the role of AI in SAP ALM. AI shouldn't be promoted as a standalone release authority. It has defensible roles of interpretation, prioritization, explanation, anomaly detection, routing recommendation and learning from outcomes. This separation has implications for enterprise adoption. Release managers and auditors won't be able to take any decisions based on black boxes, but they will welcome AI that delivers transparency, uncovers missing controls, and provides explanations for risk factors and diminishes manual effort.

A second consequence is that compliance automation needs to be built into the ALM architecture and not as an audit reporting after-thought. The operational controls that are released by lifecycles can also generate audit evidence. With approvals, tests, transport imports, overrides and monitoring results recorded on an ongoing basis, audit readiness is an operating characteristic of ALM not a project done afterwards.

12. Limitations

Firstly, this paper is a proposal of framework and design model and not a controlled field experiment. Before any statements regarding incident reduction, audit-effort reduction or improvement of the release cycle, the architecture needs empirical testing in production SAP landscapes.

Second, the practitioner evidence is anonymized. This ensures confidentiality but may hinder readers from verifying the implementation details themselves. Further research should employ formal case-study protocols with anonymous, but traceable data sets if possible.

Third, the quality of the data is important for ASACF. The evidence layer will be incomplete if transport objects are not associated to incidents, if tests are not maintained, if the scope of BPCA is not complete, if CSOL is not consistently enforced and if there is approval outside of ChaRM.

Fourth, the transferability of the model is not known. Different ticket language, module ownership, change types and change governance rules across different organizations can make a change classifier trained in one organization ineffective in another. Calibration and monitoring needs to be done on an organization basis.

Fifth, AI recommendations can pose governance risks if they are adopted without interpretation or questioning. Model governance, drift detection, explanation logging and human override controls are therefore needed in ASACF.

13. Future Research

There is a need for benchmark datasets to be developed for the governance of SAP ALM in future. These data sets would ideally contain anonymised change documents, transport objects, ATC findings, BPCA impact, SCMON/UPL usage data, test evidence, approval paths, retrofit outcomes, incidents and audit exceptions [50-55]. Model comparisons and dependency-reasoning approaches would be possible without releasing proprietary business processes while preserving privacy for privacy-preserving graph datasets.

The other direction is graph-native SAP impact analysis. Initial implementation of the SDKG can be done using graph rules with interpretable rules and graph path queries. Future work can play off of

the graph reasoning techniques outlined in the rules and graph neural networks, Bayesian networks and hybrid approaches. For contexts in which auditing is required, measures of explainability should be used in conjunction with measures of predictive performance.

The third direction is an empirical testing of the AI change classification. Studies should evaluate classification accuracy and recall with respect to the type of change, the level of confidence in the classification, the burden on human confirmation and the subsequent governance consequences. Weight misclassification cost by risk (misclassification of Urgent Change will have a more significant impact than misclassification of two low-risk categories).

The fourth direction is predictive retrofit conflict validation. Future studies should evaluate whether it is possible to predict retrofit conflicts early enough to affect the sequencing of the changes, by using object-overlap, co-change history, CSOL/DGP signals, and active N/N+1 changes. Outcomes key metrics include conflict lead time, manual rework, release delay and defect leakage due to merge.

The fifth direction is for continuous compliance evidence evaluation. Research should be conducted to assess the effectiveness of evidence assemblers in terms of audit preparation time, evidence completeness and SoD exceptions. Research should also explore how automated certificates of change governance are trusted by auditors.

Last but not least, future research could expand ASACF to other SAP Cloud ALM, SAP BTP, Clean Core governance, S/4HANA transformation programs and DevSecOps pipelines. Governance in hybrid SAP landscapes must extend across on-premise systems, cloud services, integration layers, and extension platforms.

***Table 1**. Practitioner evidence to ASACF design requirements*

Observed governance pattern	Practitioner-grounded evidence	Governance risk	ASACF design response
ITSM-to-ChaRM classification	Incoming service tickets require manual interpretation before creating RFC, Normal, Urgent, Admin or Standard change documents.	Incorrect change type can bypass required approval rigor or create unnecessary governance overhead.	AI change type classification engine with confidence-based human confirmation.
Static approval chain design	Approval workflows are often fixed by transaction type rather than transport risk, object criticality or compliance exposure.	Low-risk changes are over-governed and high-risk changes may not receive proportional review.	Risk-adaptive approval routing engine that selects approval path based on evidence and policy thresholds.
Reactive N/N+1 retrofit conflict handling	Retrofit, CSOL and Downgrade Protection identify conflicts, but many conflicts are handled after transport sequencing decisions have already been made.	Manual conflict resolution delays releases and increases risk in dual-track landscapes.	Predictive retrofit conflict engine using object overlap, co-change history, CSOL/DGP signals and N/N+1 activity.
Fragmented compliance evidence	Approval history, test evidence, transport imports, SoD checks and production confirmation are distributed across records and tools.	Audit preparation becomes manual, delayed and error-prone.	Continuous compliance evidence assembler that generates audit-ready change governance records.
Large landscape observability	Hundreds of managed systems, multiple sectors or business units and parallel release calendars create evidence volume beyond manual review.	Release managers struggle to prioritize exceptions and material risks.	SDKG and governance dashboards that consolidate evidence and rank exception criticality.

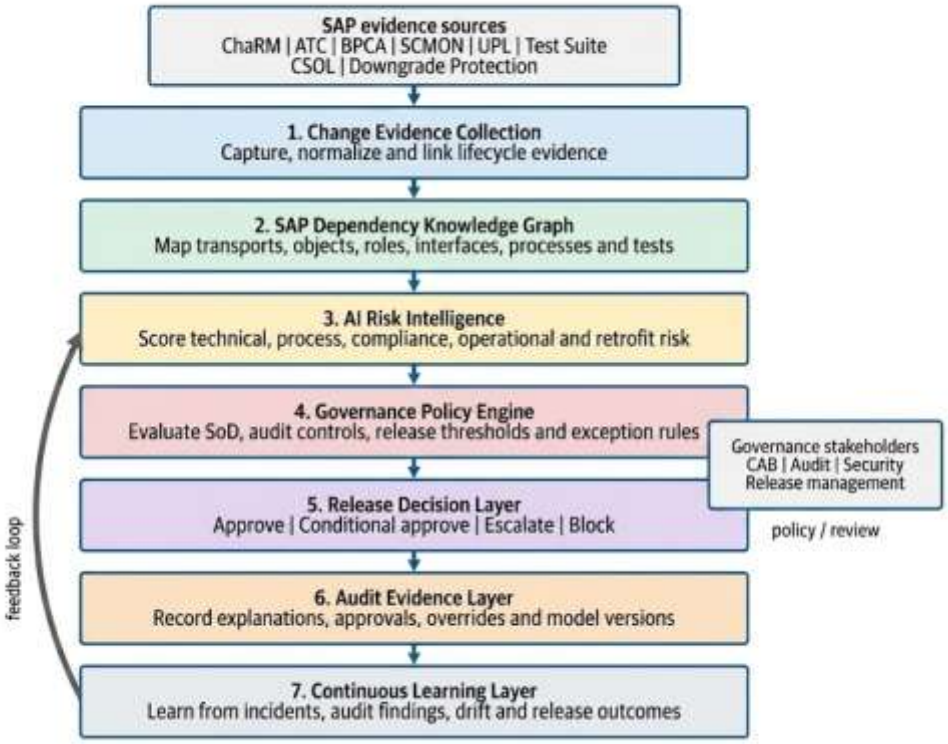


Figure 1. *ASACF architecture integrating SAP ALM evidence, risk intelligence, policy governance, release decisioning, audit evidence and learning.*

Table 2. *ASACF layer-to-SAP capability mapping*

ASACF layer	SAP evidence and capability inputs	Analytical function	Governance output
Change Evidence Collection	ChaRM, Central ATC, BPCA, SCMON, UPL, Test Suite, CSOL, DGP, Retrofit, incidents	Evidence completeness and normalization	Create a unified release evidence record
SAP Dependency Knowledge Graph	Transports, objects, roles, processes, tests, controls, incidents	Graph construction, path analysis, link confidence	Identify hidden cross-landscape and process impacts
AI Risk Intelligence	ATC findings, object history, process criticality, test coverage, incident history	Interpretable classification, calibrated risk scores, anomaly detection	Prioritize review, testing, retrofit and monitoring
Governance Policy Engine	SoD rules, release windows, audit rules, emergency-change policies	Rule evaluation, thresholds, policy exceptions	Determine required control action
Release Decision Layer	Risk scores, policy outcomes, approvals, test status, release queue	Decision rules and bounded automation	Approve, escalate, block or conditionally approve
Audit Evidence Layer	Approvals, explanations, transport imports, model versions, overrides	Evidence packaging and traceability	Support continuous assurance and audit review
Continuous Learning Layer	Incidents, defects, rollbacks, audit findings, override outcomes	Feedback learning, drift monitoring, recalibration	Improve future release governance decisions

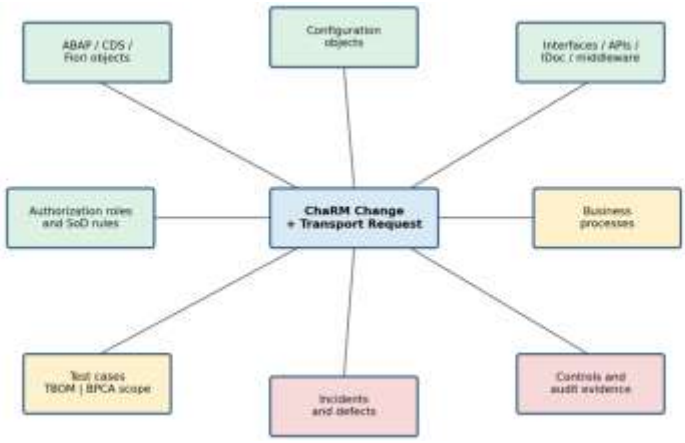


Figure 2. *SAP Dependency Knowledge Graph linking changes, objects, processes, roles, tests, incidents and controls.*

Table 3. *SDKG node categories*

Node category	Examples
Change governance nodes	ChaRM documents, change cycles, task lists, release packages, approval steps
Technical artifact nodes	ABAP programs, classes, function modules, CDS views, Fiori apps, enhancements, forms, interfaces, APIs
Configuration and process nodes	Customizing objects, business process hierarchy nodes, transactions, process steps, variants
Security and control nodes	Roles, profiles, authorization objects, SoD rules, audit controls, emergency access records
Assurance nodes	ATC findings, test cases, TBOMs, BPCA results, manual test evidence, automated test runs
Operational nodes	Incidents, defects, performance events, monitoring alerts, rollback records, post-release defects

Table 4. *SDKG edge semantics for SAP ALM governance*

Edge type	Governance meaning
contains	ChaRM change contains transport; transport contains development objects
impacts	Object or configuration change impacts process step, role, interface or control
tests	Test case or TBOM covers transaction, process step or object
violates	Change conflicts with SoD rule, evidence requirement or release policy
depends on	Object depends on another object, interface, configuration or API
co-changes with	Objects historically changed together or appear together in defect-producing transports
approved by	Approval step performed by role or user
imported to	Transport imported to development, quality, pre-production or production system
caused or associated with	Post-release incident linked to change, transport or object cluster

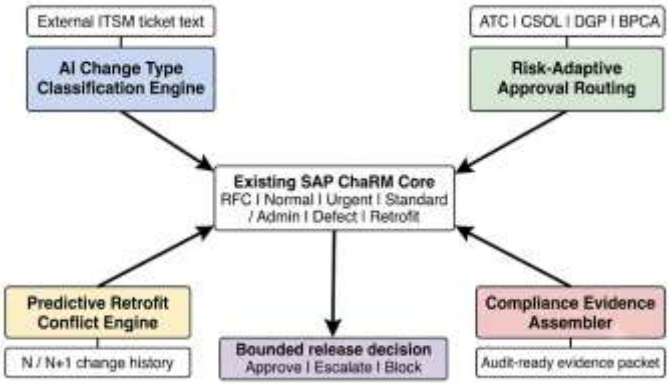


Figure 3. *Four AI governance services over existing SAP ChaRM architecture.*

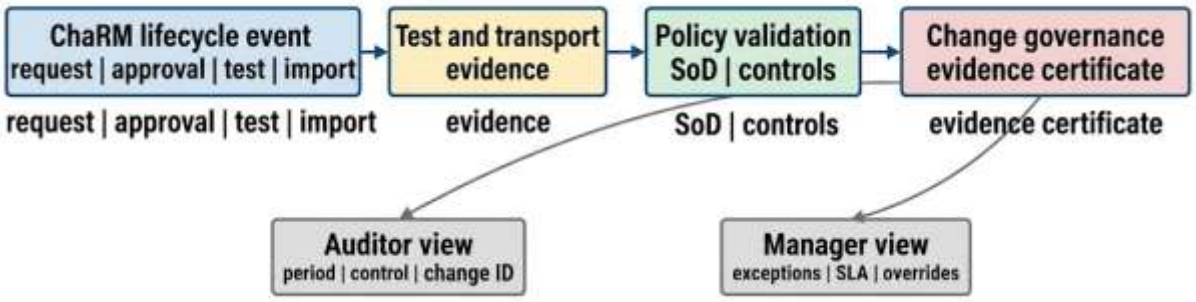


Figure 4. *Continuous compliance evidence assembly from ChaRM events, test evidence, policy validation and audit-facing records.*

Table 5. *Four AI governance services and required explanations*

Service	Inputs	Output	Required explanation	Governance risk addressed
AI change classifier	ITSM ticket text, category, priority, module, historical mapping	Recommended ChaRM transaction type and confidence score	Ticket phrases and historical examples supporting classification	Misclassification can bypass or over-apply governance
Risk-adaptive approval routing	ATC severity, process criticality, CSOL/DGP, roles, test coverage, incidents	Approval path and required reviewers	Risk factors and policy thresholds triggering the path	Static approval can under-review high-risk changes
Predictive retrofit conflict engine	N/N+1 object overlap, active changes, co-change history, CSOL/DGP, retrofit logs	Conflict probability and sequencing recommendation	Conflicting objects and related change documents	Reactive retrofit delays release and increases rework
Compliance evidence assembler	ChaRM lifecycle events, approvals, tests, transports, SoD rules, incidents	Audit-ready evidence record and exception dashboard	Evidence chain with timestamps and responsible roles	Manual assembly creates audit delay and incompleteness

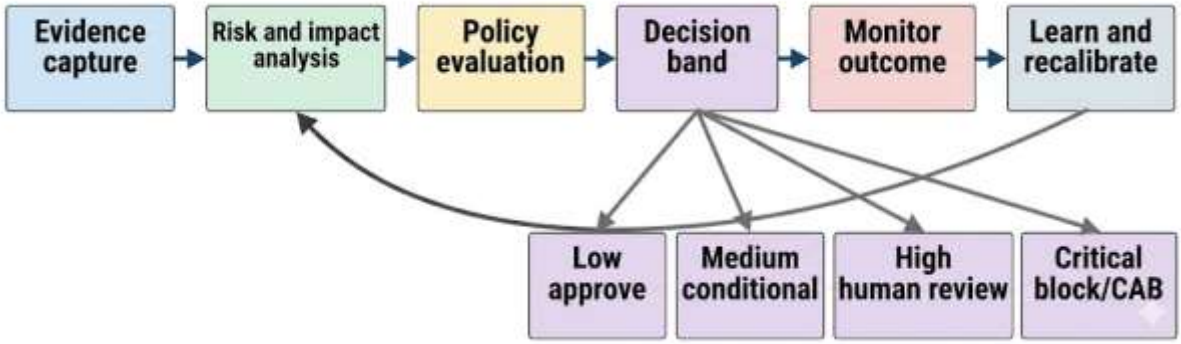


Figure 5. *Bounded autonomous release governance lifecycle.*

Table 6. *Bounded autonomy release-decision policy*

Risk tier	Typical evidence pattern	Recommended action	Audit requirement
Low	Complete evidence, no ATC high severity finding, no CSOL/DGP conflict, no control impact,	Auto-approve lower-landscape movement or standard release path	Log evidence, model score and policy match

	adequate test coverage		
Medium	Minor findings, moderate process impact, manual testing required, limited dependency uncertainty	Conditional approval with required tests or peer review	Document conditions and completion evidence
High	High-severity ATC finding, critical process impact, authorization or SoD relevance, unresolved conflict	Escalate to functional lead, security, release manager or CAB	Capture reviewer decision, override reason and extra monitoring
Critical	Material compliance risk, missing test evidence, production-down change, unresolved DGP/retrofit conflict, audit policy violation	Block or emergency path with post-implementation review	Require explicit exception approval and remediation plan

Table 7. *Evaluation metrics for ASACF validation*

Evaluation dimension	Representative metrics	Data source	Expected improvement
Release stability	Production incidents per release, rollback frequency, emergency fix rate	Incident management and release records	Lower incident leakage and fewer emergency corrections
Governance accuracy	Correct change type selection, approval path appropriateness, policy exception rate	ChaRM documents and policy engine logs	Fewer misclassifications and better risk-proportionate routing
Retrofit effectiveness	Predicted vs actual conflict rate, conflict lead time, N/N+1 synchronization delay	Retrofit, CSOL and DGP logs	Earlier conflict detection and reduced manual rework
Compliance assurance	Evidence completeness, SoD violation rate, audit findings, overdue post-implementation reviews	Audit evidence layer and access review records	More complete evidence and fewer control exceptions
Testing efficiency	Test scope reduction, critical process coverage, fault detection rate	BPCA, Test Suite and defect records	Reduced redundant testing without losing critical coverage
Model governance	Calibration error, explanation completeness, drift alerts, override outcomes	MLOps logs and audit evidence layer	Reliable, explainable and governed AI behaviour
Operational adoption	User acceptance, override frequency, approval cycle time, release manager workload	Workflow analytics and stakeholder feedback	Higher trust and lower governance friction

Table 8*. Research propositions for future empirical validation*

Proposition	Statement
P1	A SAP Dependency Knowledge Graph will improve the explainability of release-risk decisions compared with isolated workflow, code-quality or test records.
P2	AI-assisted change type classification will reduce ITSM-to-ChaRM misclassification when combined with confidence-based human confirmation.
P3	Risk-adaptive approval routing will reduce approval cycle time for low-risk changes while increasing scrutiny for high-risk changes.
P4	Predictive retrofit conflict detection will reduce N/N+1 synchronization rework compared with reactive retrofit conflict handling.
P5	Continuous compliance evidence assembly will reduce audit preparation effort and improve evidence completeness.
P6	Bounded autonomy with explanation and override logging will increase release-manager trust compared with black-box release scoring.
P7	Post-release feedback from incidents, audit findings and overrides will improve future release-risk calibration when governed through MLOps controls.

Table 9*. Incremental implementation roadmap for ASACF*

Phase	Key activities	Primary deliverable
Phase 1: Evidence baseline	Inventory ChaRM transaction types, transport routes, approval roles, ATC results, BPCA	Evidence completeness map and data-quality gaps

	usage, test repositories, CSOL/DGP logs and incident linkages.	
Phase 2: Dependency graph foundation	Create SDKG node and edge model for transports, objects, processes, roles, tests, controls and incidents.	Initial graph queries for impact and missing evidence
Phase 3: Compliance evidence assembly	Automate evidence collection for approvals, tests, imports, SoD checks and production confirmation.	Change Governance Evidence Certificate and exception dashboard
Phase 4: AI-assisted classification and routing	Train change type classifier and risk-adaptive routing model using historical records with human confirmation.	Reduced misclassification and risk-proportionate approval paths
Phase 5: Predictive retrofit governance	Model conflict risk using N/N+1 object overlap, active changes, CSOL/DGP and retrofit history.	Pre-release retrofit risk alerts and sequencing recommendations
Phase 6: Continuous learning and MLOps	Monitor model drift, override outcomes, incident feedback and audit findings; update thresholds under governance.	Closed-loop improvement with controlled model changes

Table 10. *ASACF governance maturity model*

Maturity level	Characteristic state	Primary capability	Key risk or dependency
Level 1 - Manual control	Change records and transports exist, but evidence is reviewed manually and audit records are assembled after the fact.	Individual expertise and manual checklist discipline	High variance in release decisions and audit readiness
Level 2 - Workflow control	ChaRM workflows, approval statuses, task lists and transport routes are standardized across landscapes.	Structured governance and traceable approval history	Limited risk adaptivity and fragmented evidence across tools
Level 3 - Evidence integration	ATC, BPCA, SCMON/UPL, Test Suite, CSOL/DGP and incident evidence are correlated for release review.	Better visibility into technical, process and test impacts	Evidence correlation still depends on manual interpretation
Level 4 - Predictive governance	Risk scoring, change classification, retrofit prediction and compliance evidence assembly support release decisions.	Earlier exception detection and risk-proportionate approval routing	Requires model governance, validation and explanation controls
Level 5 - Bounded autonomous ALM	Policy-aware AI services recommend, route, block, escalate and learn from outcomes under auditable human governance.	Continuous assurance, closed-loop learning and explainable release governance	Requires mature data quality, trust, operating model and MLOps discipline

14. Conclusions

This paper introduced the Autonomous SAP ALM Control Fabric (ASACF) an AI governance framework that has been developed based on a practitioner's point of view on SAP change, compliance and release management in hybrid enterprise contexts. The framework addresses a key issue of today's SAP ALM: organisations already have a lot of technical and process evidence, approval evidence, testing evidence, operational evidence and other evidence, but the evidence is not centralized and not easy to be converted to a comprehensible release decision.

The seven-layer architecture that ASACF contributes includes Change Evidence Collection, SAP Dependency Knowledge Graph, AI Risk Intelligence, Governance Policy Engine, Release Decision Layer, Audit Evidence Layer and Continuous Learning Layer. The paper went on to present four AI governance services on ChaRM: AI Change Type Classification Engine, Risk-Adaptive Approval Routing Engine, Predictive Retrofit Conflict Engine, and Continuous Compliance Evidence Assembler. They are designed to solve recurring governance issues that are seen in enterprise SAP landscapes while maintaining the governance logic of the existing SAP ChaRM workflows.

The most important theoretical contribution is bounded autonomy. In SAP ALM, do not think autonomy equals any kind of deployment automation. It should imply support for decision-making based on evidence, awareness of policies, explainability and auditability. While AI can help streamline manual work, optimize prioritization,

uncover undocumented inconsistencies, and compile compliance evidence, material release authority must be subject to a clear policy, human oversight, and audit trails.

The practical contribution is a reusable reference architecture that SAP organizations can leverage to enhance release stability, compliance readiness, testing efficiency, retrofit governance and operational learning. The scientific contribution comes in the form of an evaluation agenda for future work on autonomous SAP ALM, which includes graph-based dependency modelling, interpretable risk scoring, and predicting retrofit conflict detection and continuous compliance assurance. The future of SAP ALM is not just faster transport movements; it's intelligent, explainable and well-governed enterprise change.

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] Feldman, G., Shah, H., Chapman, C., & Amini, A. (2016). Enterprise systems: The upgrade process model. Journal of Enterprise Information Management, 29(6), 822–840. doi:10.1108/JEIM-12-2014-0122

[2] 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

[3] 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

[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] 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

[7] 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

[8] Őri, D., & Szabó, Z. (2024). A systematic literature review on business-IT misalignment research. Information Systems and e-Business Management, 22(1), 139–169. doi:10.1007/s10257-023-00664-w

[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] 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

[11] 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

[12] 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

[13] 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

[14] 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

[15] van Beest, N., Groefsema, H., Cryer, A., Governatori, G., Tosatto, S. C., & Burke, H. (2023). Cross-instance regulatory compliance checking of business process event logs. IEEE

Transactions on Software Engineering, 49(11), 4917–4931. doi:10.1109/TSE.2023.3319086

[16] 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

[17] Waseem, M., Liang, P., & Shahin, M. (2020). A systematic mapping study on microservices architecture in DevOps. Journal of Systems and Software, 170, Article 110798. doi:10.1016/j.jss.2020.110798

[18] 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

[19] 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

[20] 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

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

[22] 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

[23] Batool, A., Zowghi, D., & Bano, M. (2025). AI governance: A systematic literature review. AI and Ethics. doi:10.1007/s43681-024-00653-w

[24] 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

[25] 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

[26] 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

[27] 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

[28] 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), 1–37. doi:10.1145/3469440

[29] 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

[30] 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

[31] 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

[32] Butarbutar, Z. T., Handayani, P. W., Suryono, R. R., & Wibowo, F. W. (2023). Systematic literature review of critical success factors on enterprise resource planning post implementation. Cogent Business & Management, 10(3), Article 2264001. doi:10.1080/23311975.2023.2264001

[33] Mahmood, F., Khan, A. Z., Shah, S. A., & Adil, M. (2024). Post ERP implementation issues and challenges: Exploratory case studies in the context of Saudi Arabia. Kybernetes, 53(12), 5749–5774. doi:10.1108/K-06-2022-0914

[34] 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

[35] Kirmizi, M., & Kocaoglu, B. (2022). The influencing factors of enterprise resource planning readiness stage on enterprise resource planning project success: A project manager's perspective. Kybernetes, 51(3), 1089–1113. doi:10.1108/K-11-2020-0812

[36] Jans, M., Soffer, P., & Jouck, T. (2019). Building a valuable event log for process mining: An experimental exploration of a guided process. Enterprise Information Systems, 13(5), 601–630. doi:10.1080/17517575.2019.1587788

[37] 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

[38] El-Gharib, N. M., & Amyot, D. (2023). Robotic process automation using process mining: A systematic literature review. Data & Knowledge Engineering, 148, Article 102229. doi:10.1016/j.datak.2023.102229

[39] Azad, N., & Hyrynsalmi, S. (2023). DevOps critical success factors and organizational practices: A systematic literature review. Information and Software Technology, 157, Article 107150. doi:10.1016/j.infsof.2023.107150

[40] Kumar, R., Nadeem, M., & Shameem, M. (2025). DevOps metrics: A systematic literature review. Journal of Software: Evolution and Process, 37(1), Article e2733. doi:10.1002/smr.2733

[41] Lwakatare, L. E., Kuvaja, P., & Oivo, M. (2019). Dimensions of DevOps. Information and Software Technology, 114, 217–230. doi:10.1016/j.infsof.2019.06.010

[42] Laukkanen, E., Itkonen, J., & Lassenius, C. (2017). Problems, causes and solutions when adopting continuous delivery: A systematic literature review. Information and Software Technology, 82, 55–79. doi:10.1016/j.infsof.2016.10.001

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

[44] Stradowski, D., & Madeyski, L. (2023). The state of the art in software defect prediction. Information and Software Technology, 159, Article 107192. doi:10.1016/j.infsof.2023.107192

[45] 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

[46] 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

[47] 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

[48] 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

[49] 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

[50] 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

[51] Paulheim, H. (2017). Knowledge graph refinement: A survey of approaches and evaluation methods. Semantic Web, 8(3), 489–508. doi:10.3233/SW-160218

[52] 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

[53] 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

[54] 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

[55] 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