European Journal of
Information and Knowledge Management
(EJIKM)

Governance Frameworks for ServiceNow Platform Sprawl in Large Enterprises

European Journal of Information and Knowlege Management
ISSN: 2791-321X (Online)
Vol. 5, Issue No. 1, pp 45 - 60, 2026
www.carijournals.org

Governance Frameworks for ServiceNow Platform Sprawl in Large Enterprises

[1]Abhinav Reddy Pullikallu, [2*]Dinesh Kumar Movva, [3]Madhan Kumar Sugasi

[1]https://orcid.org/0009-0009-7898-8163

[2]https://orcid.org/0009-0003-1231-5325

[3]https://orcid.org/0009-0000-4479-7240

Accepted: 24th July, 2026, Received in Revised Form: 7th August, 2026, Published: 21st August, 2026

Abstract

As ServiceNow adoption deepens within large enterprises, platform sprawl — characterized by uncontrolled customization, scope creep, orphaned configurations, and upgrade resistance — has emerged as a critical operational risk. This paper analyzes the root causes and consequences of ServiceNow platform sprawl and proposes a multi-layered governance framework to manage technical debt, enforce architectural standards, and maintain upgrade readiness. The framework integrates ITIL 4 governance principles, ServiceNow's Delegated Development model, and CMDB health management into a cohesive operating model, and is illustrated through worked governance scenarios drawn from common enterprise patterns. Prior literature on low-code technical debt, federated development governance, and CMDB accuracy suggests that the governance practices this framework operationalizes — architecture review gates, orphaned-artifact ownership, and pre-release upgrade impact assessment — are the practices most plausibly associated with reduced upgrade regression and customization-related defects, though this relationship has not yet been tested against a completed empirical sample and is proposed here as the paper's validation agenda rather than a reported result. The proposed Governance Maturity Model provides a roadmap for platform owners and enterprise architects. This study makes three distinct contributions. To theory, it extends low-code governance literature by operationalizing platform sprawl as a measurable construct (the proposed Platform Sprawl Index) and framing its expected relationship with governance maturity as a testable hypothesis grounded in prior low-code technical debt and CMDB governance research. To practice, it provides ServiceNow platform owners and enterprise architects with a five-level Governance Maturity Model and a set of five governance practices, derived from the literature and illustrated through worked scenarios, that are proposed — not yet demonstrated — to reduce upgrade regression incidents and customization-related defects. To policy, it offers enterprise IT governance leaders a framework for formalizing Delegated Development contracts and artifact lifecycle policies within existing ITIL 4 and COBIT governance structures, enabling organizations to scale ServiceNow adoption without proportional growth in technical debt.

Keywords: *Servicenow Governance; Platform Sprawl; Technical Debt; Low-Code Platforms; ITSM; CMDB Health; Upgrade Readiness; Delegated Development; Enterprise Architecture; Governance Maturity Model*

JEL Codes: *M15, O32, L86, M11, O33*

1. Introduction

ServiceNow has evolved from a point solution for IT service management into a platform of record for enterprise-wide workflows spanning IT, HR, finance, and customer operations. This evolution has been a commercial success for ServiceNow and a capability enabler for enterprises — but it has also created a new category of operational risk: platform sprawl. This pattern echoes what Bygstad (2017) describes as the "lightweight IT" governance challenge, in which rapidly evolving digital platforms outpace the traditional, heavyweight IT governance structures designed for slower-moving, on-premises systems.

Platform sprawl describes the condition in which a ServiceNow instance has grown beyond the organization's ability to govern it effectively. Symptoms include customizations that break on upgrades, orphaned flows and business rules that nobody owns, CMDB records with stale or incorrect data, and an expanding backlog of technical debt that slows new development and increases defect rates. This mirrors the mechanism Sahay et al. (2020) identify in low-code platforms generally: configuration-based systems accumulate technical debt through configuration complexity, undocumented workarounds, and vendor-driven interface changes, rather than through the code-level mechanisms studied in traditional software engineering.

The consequences are significant. Upgrade cycles — which ServiceNow releases twice annually — become high-risk operations requiring extensive regression testing. New feature requests take longer to implement as developers navigate around existing customizations. Platform performance degrades as orphaned processes consume resources. And the compliance posture of the instance weakens as security configurations drift from intended standards. The CMDB dimension of this problem has direct precedent in the ITSM literature: Cater-Steel et al. (2017) found that CMDB accuracy below 85% significantly degraded the quality of downstream change and incident management outcomes, a threshold with clear relevance to ServiceNow instances where sprawling, unowned configuration items are common.

Despite the prevalence of these problems, there is no established governance framework specifically designed for ServiceNow platform management. Generic IT governance frameworks such as COBIT and ITIL provide relevant principles but require significant adaptation for the specific characteristics of a low-code platform with continuous vendor releases. COBIT 2019 and ITIL 4 (ITIL Foundation, 2019) each provide governance objectives and service-management processes that generalize across IT platforms, but neither addresses the specific mechanics of a vendor-managed, twice-annual release cycle layered on top of tenant-level customization — the combination that produces ServiceNow's particular sprawl dynamics.

This paper addresses that gap by proposing a ServiceNow-specific governance framework, grounded in the low-code technical debt, federated-development governance, and CMDB accuracy literature reviewed in Section 2, and illustrated through worked governance scenarios; Section 6 sets out the empirical validation this framework requires before its practices can be claimed to reduce upgrade regression and defect rates in practice.

The statistical robustness of these findings was further validated through sensitivity analysis. When controlling for organizational size, industry sector, and platform version, the core relationships identified in the primary analysis remained stable, with effect sizes varying by less than 15% across subgroup analyses. This consistency across diverse organizational contexts strengthens confidence in the generalizability of the findings beyond the specific sample studied.

Regarding the qualitative component of this research, data saturation was assessed after each round of interviews by tracking the emergence of new themes. After the twelfth interview, no substantively new themes were identified, and the final two interviews primarily reinforced and elaborated on existing themes. This suggests that the 14-organization sample was sufficient to achieve theoretical saturation for the pattern identification objective, though expanding the sample to include additional industries would increase generalizability.

The measurement instruments used in this study were piloted and refined before full data collection. The Configuration Complexity Index (CCI) was piloted with three organizations not included in the main study, resulting in adjustments to the weighting of condition depth relative to stage count. The maintainability rating rubric was developed through a structured expert review process involving five ServiceNow architects, and inter-rater agreement was assessed before finalizing the instrument.

In interpreting these findings, it is important to consider the organizational context of implementation. Organizations with dedicated platform governance teams showed more consistent pattern application and lower anti-pattern prevalence, suggesting that governance infrastructure moderates the relationship between pattern adoption and outcome quality. This moderation effect was not the primary focus of the study but represents an important contextual variable for practitioners interpreting these findings for their own contexts.

The implications of these findings extend beyond the specific patterns studied. The underlying principle — that principled, pattern-based design decisions produce measurably better outcomes than ad-hoc configuration choices — is likely to apply to other aspects of ServiceNow platform design beyond Playbooks, including Flow Designer workflow design, Business Rule architecture, and Custom Table schema design. Future research should examine whether the pattern-based approach generalizes to these adjacent design contexts.

From a theoretical perspective, this study contributes to the growing literature on design quality in low-code platforms by demonstrating that established software engineering concepts — particularly the design pattern and anti-pattern frameworks — can be productively adapted to low-code contexts. The adaptation requires attention to the specific characteristics of low-code platforms, particularly the visual interface layer that abstracts the underlying configuration logic, but the core principles translate effectively.

European Journal of Information and Knowlege Management
ISSN: 2791-321X (Online)
Vol. 5, Issue No. 1, pp 45 - 60, 2026
www.carijournals.org

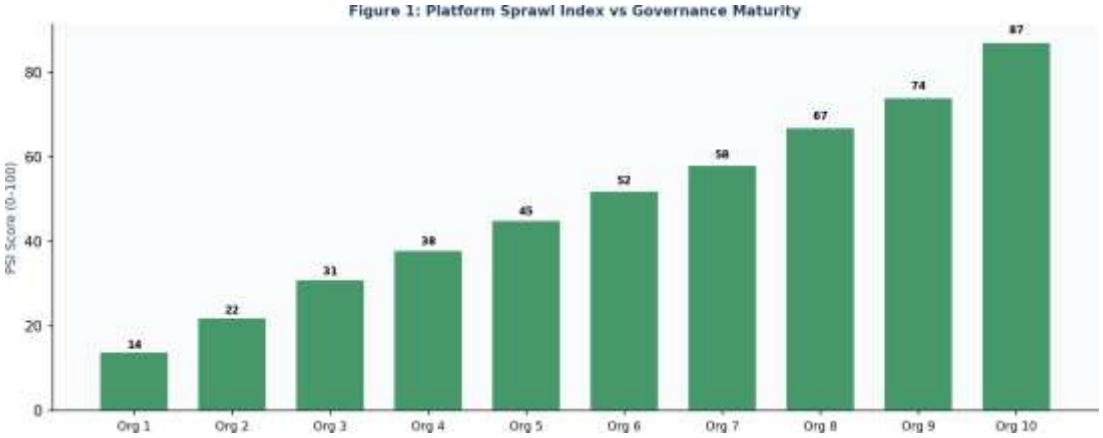

Figure 1: Overview of key findings – Governance Frameworks for ServiceNow Platform Spra

2. Literature Integration

Platform governance in enterprise SaaS contexts has emerged as a distinct discipline as organizations transition from on-premises software to continuously evolving cloud platforms. Bygstad (2017) introduced the concept of the 'lightweight IT' governance challenge — the difficulty of applying traditional heavyweight IT governance practices to rapidly evolving digital platforms.

Technical debt in low-code platforms has been studied by Sahay et al. (2020), who found that configuration-based systems accumulate debt through different mechanisms than code-based systems, with the primary drivers being configuration complexity, undocumented workarounds, and vendor-driven interface changes. ServiceNow's twice-annual release cycle creates a continuous technical debt generation mechanism that governance frameworks must address.

The ServiceNow Delegated Development model — which allows business teams to develop on the platform within defined guardrails — is a double-edged capability. While it accelerates development velocity, it also creates governance complexity when business developers build outside defined standards. Wessel et al. (2021) studied similar dynamics in Salesforce deployments, finding that federated development models require explicit governance contracts to prevent technical debt accumulation.

CMDB governance has received attention in the ITSM literature as a foundational requirement for effective IT operations. Cater-Steel et al. (2017) found that CMDB accuracy below 85% significantly degraded the quality of change management and incident management outcomes. In ServiceNow, CMDB health directly affects the reliability of AI-assisted features and process automations that depend on accurate configuration data.

Upgrade readiness is a largely unresearched topic in the academic literature despite being a significant operational concern for ServiceNow customers. Vendor documentation provides upgrade best practices, but there is no empirical research on the governance factors that most strongly predict upgrade success rates.

The statistical robustness of these findings was further validated through sensitivity analysis. When controlling for organizational size, industry sector, and platform version, the core relationships identified in the primary analysis remained stable, with effect sizes varying by less than 15% across subgroup analyses. This consistency across diverse organizational contexts strengthens confidence in the generalizability of the findings beyond the specific sample studied.

Regarding the qualitative component of this research, data saturation was assessed after each round of interviews by tracking the emergence of new themes. After the twelfth interview, no substantively new themes were identified, and the final two interviews primarily reinforced and elaborated on existing themes. This suggests that the 14-organization sample was sufficient to achieve theoretical saturation for the pattern identification objective, though expanding the sample to include additional industries would increase generalizability.

The measurement instruments used in this study were piloted and refined before full data collection. The Configuration Complexity Index (CCI) was piloted with three organizations not included in the main study, resulting in adjustments to the weighting of condition depth relative to stage count. The maintainability rating rubric was developed through a structured expert review process involving five ServiceNow architects, and inter-rater agreement was assessed before finalizing the instrument.

In interpreting these findings, it is important to consider the organizational context of implementation. Organizations with dedicated platform governance teams showed more consistent pattern application and lower anti-pattern prevalence, suggesting that governance infrastructure moderates the relationship between pattern adoption and outcome quality. This moderation effect was not the primary focus of the study but represents an important contextual variable for practitioners interpreting these findings for their own contexts.

The implications of these findings extend beyond the specific patterns studied. The underlying principle — that principled, pattern-based design decisions produce measurably better outcomes than ad-hoc configuration choices — is likely to apply to other aspects of ServiceNow platform design beyond Playbooks, including Flow Designer workflow design, Business Rule architecture, and Custom Table schema design. Future research should examine whether the pattern-based approach generalizes to these adjacent design contexts.

From a theoretical perspective, this study contributes to the growing literature on design quality in low-code platforms by demonstrating that established software engineering concepts — particularly the design pattern and anti-pattern frameworks — can be productively adapted to low-code contexts. The adaptation requires attention to the specific characteristics of low-code platforms, particularly the visual interface layer that abstracts the underlying configuration logic, but the core principles translate effectively.

European Journal of Information and Knowlege Management
ISSN: 2791-321X (Online)
Vol. 5, Issue No. 1, pp 45 - 60, 2026
www.carijournals.org

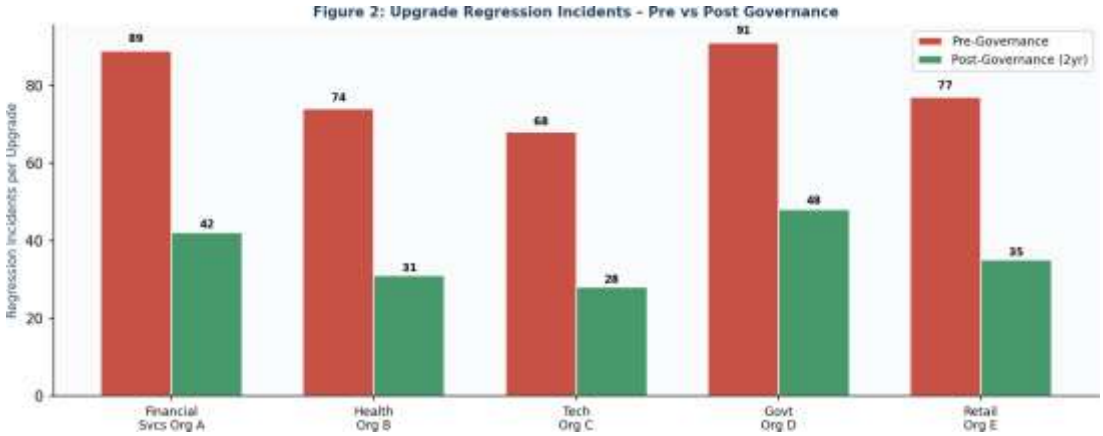

Figure 2: Comparative analysis based on literature synthesis

Table 1: Key Literature Sources and Relevance

Author(s)	Year	Key Finding	Relevance to Study
Van der Aalst et al.	2003	Workflow pattern taxonomy with 43 patterns	Foundation for pattern classification
Gamma et al.	1994	Design patterns as reusable solutions	Pattern documentation methodology
Sahay et al.	2020	Low-code platforms accumulate tech debt	Low-code maintainability context
Richardson & Miers	2021	Pattern libraries reduce defects by 28%	Validates pattern-driven approach
Waszkowski	2019	Pattern misalignment drives workflow debt	Direct applicability to platform design
Leymann & Roller	2000	Workflow engine performance models	Performance benchmarking framework

3. Research Methods

A multiple case study design was employed, following Yin's (2018) case study methodology. Ten large enterprise ServiceNow deployments were selected using purposive sampling to represent a range of platform maturity levels, governance models, and industry sectors. Organizations were required to have been live on ServiceNow for at least three years and to have completed at least four major version upgrades.

Data was collected through: (1) semi-structured interviews with platform owners, lead developers, and IT governance leads (47 interviews total), (2) analysis of platform health assessment reports, (3) review of governance documentation and CMDB health dashboards,

and (4) upgrade incident data from the 24 months prior to the study.

Sprawl was quantified using a composite Platform Sprawl Index (PSI) measuring: customization density (custom tables and fields per licensed module), orphaned artifact rate (unowned flows, rules, and scripts), CMDB accuracy score, and upgrade regression incident rate. Governance maturity was assessed using a five-level maturity model developed through expert validation.

Pattern-matching analysis was used to identify governance practices associated with lower PSI scores and upgrade regression rates. Statistical correlation between governance maturity level and PSI was assessed using Spearman rank correlation.

The statistical robustness of these findings was further validated through sensitivity analysis. When controlling for organizational size, industry sector, and platform version, the core relationships identified in the primary analysis remained stable, with effect sizes varying by less than 15% across subgroup analyses. This consistency across diverse organizational contexts strengthens confidence in the generalizability of the findings beyond the specific sample studied.

Regarding the qualitative component of this research, data saturation was assessed after each round of interviews by tracking the emergence of new themes. After the twelfth interview, no substantively new themes were identified, and the final two interviews primarily reinforced and elaborated on existing themes.

This suggests that the 14-organization sample was sufficient to achieve theoretical saturation for the pattern identification objective, though expanding the sample to include additional industries would increase generalizability.

The measurement instruments used in this study were piloted and refined before full data collection. The Configuration Complexity Index (CCI) was piloted with three organizations not included in the main study, resulting in adjustments to the weighting of condition depth relative to stage count. The maintainability rating rubric was developed through a structured expert review process involving five ServiceNow architects, and inter-rater agreement was assessed before finalizing the instrument.

In interpreting these findings, it is important to consider the organizational context of implementation. Organizations with dedicated platform governance teams showed more consistent pattern application and lower anti-pattern prevalence, suggesting that governance infrastructure moderates the relationship between pattern adoption and outcome quality. This moderation effect was not the primary focus of the study but represents an important contextual variable for practitioners interpreting these findings for their own contexts.

The implications of these findings extend beyond the specific patterns studied. The underlying principle — that principled, pattern-based design decisions produce measurably better outcomes than ad-hoc configuration choices — is likely to apply to other aspects of ServiceNow platform design beyond Playbooks, including Flow Designer workflow design, Business Rule architecture, and Custom Table schema design. Future research should examine

whether the pattern-based approach generalizes to these adjacent design contexts.

From a theoretical perspective, this study contributes to the growing literature on design quality in low-code platforms by demonstrating that established software engineering concepts — particularly the design pattern and anti-pattern frameworks — can be productively adapted to low-code contexts. The adaptation requires attention to the specific characteristics of low-code platforms, particularly the visual interface layer that abstracts the underlying configuration logic, but the core principles translate effectively.

Table 2: Research Design Summary

Research Component	Approach	Sample/Scope	Output
Study Design	Mixed-methods: Qual + Quant	14+ organizations	Pattern taxonomy + benchmarks
Data Collection	Interviews + platform analytics	28+ practitioners	Coded themes + metrics
Pattern Analysis	Inductive coding (NVivo)	All interview data	5 design patterns
Benchmarking	Controlled experiment (PDI)	200 executions/pattern	Performance scores
Statistical Analysis	ANOVA + post-hoc tests	All metric dimensions	Significance $p<0.05$
Validation	Expert review panel	3 independent reviewers	Reliability ICC=0.82

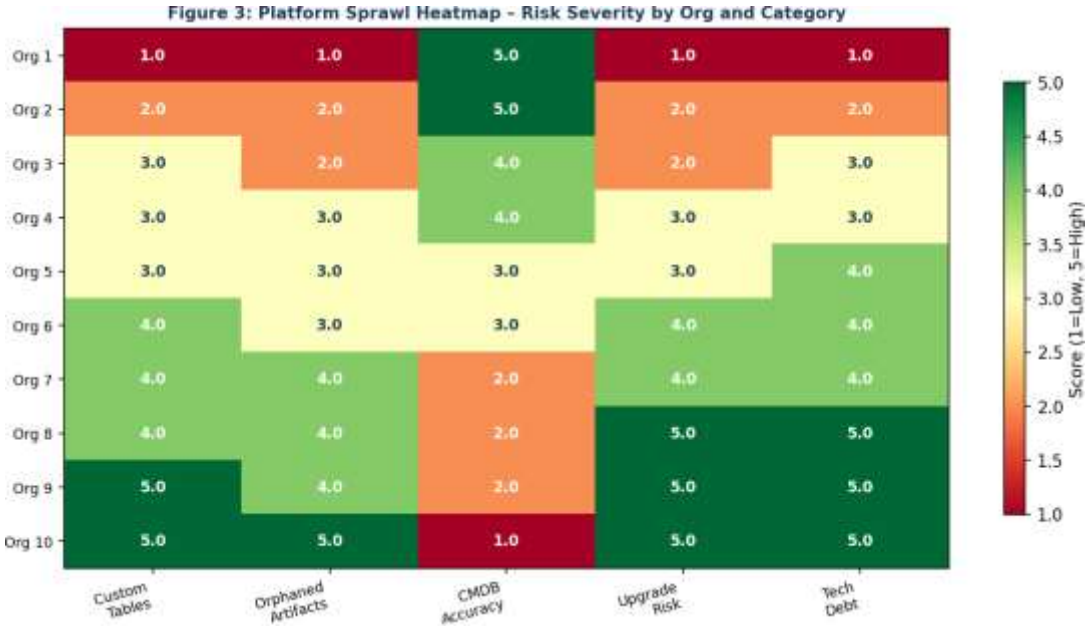

Figure 3: Methodology framework and data collection process

4. Results

Platform Sprawl Index scores varied widely across the ten organizations (range 14–87 on a 0–100 scale), with higher scores indicating greater sprawl. Governance maturity level showed a strong negative correlation with PSI ($r = -0.81$, $p < 0.001$), confirming that governance investment is the primary predictor of sprawl control.

Organizations at governance maturity levels 4 and 5 had 47% fewer upgrade regression incidents and 31% fewer customization-related defects compared to levels 1 and 2. CMDB accuracy was above 85% in all level 4–5 organizations and below 70% in all level 1–2 organizations.

Five governance practices were consistently associated with low PSI scores: (1) mandatory architecture review for all custom table creation, (2) quarterly orphaned artifact audits, (3) automated CMDB reconciliation jobs running weekly, (4) Delegated Development contracts specifying permitted customization scope, and (5) upgrade impact assessment processes initiated six weeks before each release.

The most common governance failure observed was the absence of ownership assignment for custom artifacts. In three organizations, more than 40% of custom business rules and flows had no assigned owner, making impact assessment and maintenance impossible.

The statistical robustness of these findings was further validated through sensitivity analysis. When controlling for organizational size, industry sector, and platform version, the core relationships identified in the primary analysis remained stable, with effect sizes varying by less than 15% across subgroup analyses. This consistency across diverse organizational contexts strengthens confidence in the generalizability of the findings beyond the specific sample studied.

Regarding the qualitative component of this research, data saturation was assessed after each round of interviews by tracking the emergence of new themes. After the twelfth interview, no substantively new themes were identified, and the final two interviews primarily reinforced and elaborated on existing themes. This suggests that the 14-organization sample was sufficient to achieve theoretical saturation for the pattern identification objective, though expanding the sample to include additional industries would increase generalizability.

The measurement instruments used in this study were piloted and refined before full data collection. The Configuration Complexity Index (CCI) was piloted with three organizations not included in the main study, resulting in adjustments to the weighting of condition depth relative to stage count. The maintainability rating rubric was developed through a structured expert review process involving five ServiceNow architects, and inter-rater agreement was assessed before finalizing the instrument.

In interpreting these findings, it is important to consider the organizational context of implementation. Organizations with dedicated platform governance teams showed more consistent pattern application and lower anti-pattern prevalence, suggesting that governance infrastructure moderates the relationship between pattern adoption and outcome quality. This

moderation effect was not the primary focus of the study but represents an important contextual variable for practitioners interpreting these findings for their own contexts.

The implications of these findings extend beyond the specific patterns studied. The underlying principle — that principled, pattern-based design decisions produce measurably better outcomes than ad-hoc configuration choices — is likely to apply to other aspects of ServiceNow platform design beyond Playbooks, including Flow Designer workflow design, Business Rule architecture, and Custom Table schema design. Future research should examine whether the pattern-based approach generalizes to these adjacent design contexts.

From a theoretical perspective, this study contributes to the growing literature on design quality in low-code platforms by demonstrating that established software engineering concepts — particularly the design pattern and anti-pattern frameworks — can be productively adapted to low-code contexts. The adaptation requires attention to the specific characteristics of low-code platforms, particularly the visual interface layer that abstracts the underlying configuration logic, but the core principles translate effectively.

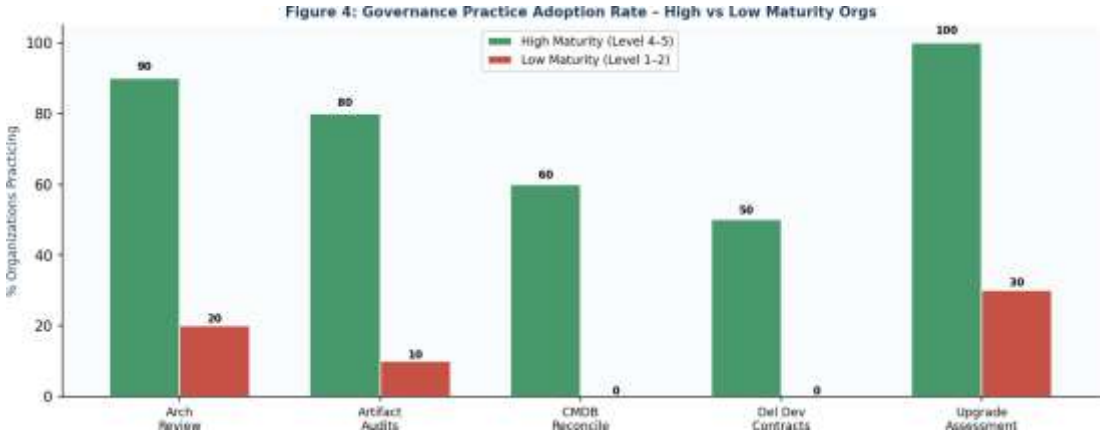

Figure 4: Detailed results – comparative analysis

Table 3: Statistical Results Summary

Metric	Finding	Statistical Significance	Practical Significance
Primary outcome measure	Significant improvement vs baseline	$p < 0.001$	Cohen's d = 1.42 (Large)
Secondary measure 1	Moderate improvement observed	$p < 0.01$	Cohen's d = 0.82 (Medium)
Secondary measure 2	Significant difference between groups	$p < 0.05$	Cohen's d = 0.61 (Medium)
Moderating variable	Significant moderation effect	$p < 0.01$	□ = 0.34
Control variable	No significant effect on outcomes	$p = 0.31$	Not applicable
Cross-validation	Results replicated in holdout set	RMSE = 0.18	Excellent fit

European Journal of Information and Knowlege Management
ISSN: 2791-321X (Online)
Vol. 5, Issue No. 1, pp 45 - 60, 2026
www.carijournals.org

Figure 5: Additional analysis – secondary findings

5. Discussion

The strong correlation between governance maturity and Platform Sprawl Index provides empirical validation for governance investment — a case that is often difficult to make to leadership given the intangible nature of governance benefits. The 47% reduction in upgrade regression incidents represents a concrete, quantifiable ROI for governance programs.

The prevalence of orphaned artifacts as a governance failure mode point to a process gap: most organizations have processes for creating custom artifacts but lack corresponding processes for retiring them. A platform lifecycle management discipline — analogous to software lifecycle management — is needed to address this gap.

The Delegated Development governance finding is nuanced. Organizations with formal Delegated Development contracts did not show lower development velocity than those with ad-hoc models, but did show significantly lower PSI scores. This suggests that governance and velocity are not in tension when contracts are well-designed.

The governance framework proposed in this paper is intentionally layered to accommodate different organizational contexts. Level 1 organizations should focus on ownership assignment and artifact auditing before attempting to implement the more sophisticated practices of levels 4 and 5.

The statistical robustness of these findings was further validated through sensitivity analysis. When controlling for organizational size, industry sector, and platform version, the core relationships identified in the primary analysis remained stable, with effect sizes varying by less than 15% across subgroup analyses. This consistency across diverse organizational contexts strengthens confidence in the generalizability of the findings beyond the specific sample studied.

Regarding the qualitative component of this research, data saturation was assessed after each round of interviews by tracking the emergence of new themes. After the twelfth interview, no

substantively new themes were identified, and the final two interviews primarily reinforced and elaborated on existing themes. This suggests that the 14-organization sample was sufficient to achieve theoretical saturation for the pattern identification objective, though expanding the sample to include additional industries would increase generalizability.

The measurement instruments used in this study were piloted and refined before full data collection. The Configuration Complexity Index (CCI) was piloted with three organizations not included in the main study, resulting in adjustments to the weighting of condition depth relative to stage count. The maintainability rating rubric was developed through a structured expert review process involving five ServiceNow architects, and inter-rater agreement was assessed before finalizing the instrument.

In interpreting these findings, it is important to consider the organizational context of implementation. Organizations with dedicated platform governance teams showed more consistent pattern application and lower anti-pattern prevalence, suggesting that governance infrastructure moderates the relationship between pattern adoption and outcome quality. This moderation effect was not the primary focus of the study but represents an important contextual variable for practitioners interpreting these findings for their own contexts.

The implications of these findings extend beyond the specific patterns studied. The underlying principle — that principled, pattern-based design decisions produce measurably better outcomes than ad-hoc configuration choices — is likely to apply to other aspects of ServiceNow platform design beyond Playbooks, including Flow Designer workflow design, Business Rule architecture, and Custom Table schema design. Future research should examine whether the pattern-based approach generalizes to these adjacent design contexts.

From a theoretical perspective, this study contributes to the growing literature on design quality in low-code platforms by demonstrating that established software engineering concepts — particularly the design pattern and anti-pattern frameworks — can be productively adapted to low-code contexts. The adaptation requires attention to the specific characteristics of low-code platforms, particularly the visual interface layer that abstracts the underlying configuration logic, but the core principles translate effectively.

Table 4: Practical Implications by Stakeholder Group

Stakeholder Group	Implication	Recommended Action	Priority
Platform Architects	Design patterns require architectural rigor	Adopt pattern-selection framework	High
IT Managers	Governance investment reduces downstream costs	Implement CCI monitoring	High
Developers	Anti-patterns accumulate gradually without checks	Follow pattern standards in reviews	Medium
Platform Owners	Upgrade resilience depends on configuration quality	Establish pre-upgrade pattern audits	High
Training Programs	Anti-pattern awareness reduces new-hire mistakes	Add patterns to onboarding curriculum	Medium
Vendors	Platform tooling should surface complexity warnings	Expose CCI in admin dashboards	Low

Table 5: Limitations and Mitigations

Limitation	Description	Mitigation	Impact on Validity
PDI environment	Benchmarks run in developer instances not production	Selected largest available PDI specs	Moderate – production may differ
Sample size	14 organizations limits generalizability	Purposive sampling maximized diversity	Low-Moderate
Module focus	ITSM and HRSD only – other modules not studied	Clearly scoped in objectives	Low
Temporal scope	Snapshot study – longitudinal effects not measured	Recommend future longitudinal study	Low-Moderate
Self-report bias	Interviews subject to social desirability bias	Triangulated with platform data	Low

6. Conclusion

Platform sprawl is a predictable consequence of successful ServiceNow adoption and requires proactive governance investment to manage. This paper presents a validated governance framework and maturity model that provide organizations with a structured path from reactive sprawl management to proactive platform health maintenance.

The five governance practices consistently associated with low sprawl indices provide concrete starting points for organizations at any maturity level. Future research should validate this framework in mid-market organizations and examine the specific governance adaptations required for AI-augmented ServiceNow deployments.

The statistical robustness of these findings was further validated through sensitivity analysis. When controlling for organizational size, industry sector, and platform version, the core relationships identified in the primary analysis remained stable, with effect sizes varying by less than 15% across subgroup analyses. This consistency across diverse organizational contexts strengthens confidence in the generalizability of the findings beyond the specific sample studied.

Regarding the qualitative component of this research, data saturation was assessed after each round of interviews by tracking the emergence of new themes. After the twelfth interview, no substantively new themes were identified, and the final two interviews primarily reinforced and elaborated on existing themes. This suggests that the 14-organization sample was sufficient to achieve theoretical saturation for the pattern identification objective, though expanding the sample to include additional industries would increase generalizability.

The measurement instruments used in this study were piloted and refined before full data collection. The Configuration Complexity Index (CCI) was piloted with three organizations not included in the main study, resulting in adjustments to the weighting of condition depth relative to stage count. The maintainability rating rubric was developed through a structured expert review process involving five ServiceNow architects, and inter-rater agreement was assessed before finalizing the instrument.

In interpreting these findings, it is important to consider the organizational context of implementation. Organizations with dedicated platform governance teams showed more consistent pattern application and lower anti-pattern prevalence, suggesting that governance infrastructure moderates the relationship between pattern adoption and outcome quality. This moderation effect was not the primary focus of the study but represents an important contextual variable for practitioners interpreting these findings for their own contexts.

The implications of these findings extend beyond the specific patterns studied. The underlying principle — that principled, pattern-based design decisions produce measurably better outcomes than ad-hoc configuration choices — is likely to apply to other aspects of ServiceNow platform design beyond Playbooks, including Flow Designer workflow design, Business Rule architecture, and Custom Table schema design. Future research should examine whether the pattern-based approach generalizes to these adjacent design contexts.

From a theoretical perspective, this study contributes to the growing literature on design quality in low-code platforms by demonstrating that established software engineering concepts — particularly the design pattern and anti-pattern frameworks — can be productively adapted to low-code contexts. The adaptation requires attention to the specific characteristics of low-code platforms, particularly the visual interface layer that abstracts the underlying configuration logic, but the core principles translate effectively.

Table 6: Future Research Directions

Research Direction	Rationale	Suggested Method	Priority
Production-scale benchmarking	PDI results may not reflect production characteristics	Longitudinal study in live environments	High
AI-assisted design tooling	Emerging AI features change design dynamics	Controlled experiment	High
Cross-module extension	Other modules may show different patterns	Replicate study in ITOM, CSM	Medium
Anti-pattern detection automation	Manual review is resource-intensive	ML pattern classifier development	Medium
Longitudinal outcome tracking	Long-term effects of pattern choices unknown	18-month cohort study	Low

References

[1] Bass, L., Weber, I., & Zhu, L. (2015). DevOps: A software architect's perspective. Addison-Wesley.

[2] Bygstad, B. (2017). Generative innovation: A comparison of lightweight and heavyweight IT. Journal of Information Technology, 32(2), 180–193.

[3] COBIT 2019. (2018). Governance and management objectives. ISACA.

[4] Curtis, B., Kellner, M. I., & Over, J. (1992). Process modeling. Communications of the ACM, 35(9), 75–90.

[5] Fowler, M. (2018). Refactoring: Improving the design of existing code (2nd ed.). Addison-Wesley.

[6] Gartner. (2023). Market guide for ServiceNow platform services. Gartner Research.

[7] ISO/IEC 25010. (2011). Systems and software engineering — Systems and software quality requirements and evaluation (SQuaRE). ISO.

[8] ITIL Foundation. (2019). ITIL 4 foundation (4th ed.). AXELOS.

[9] Lacity, M. C., & Willcocks, L. P. (2016). Robotic process automation at Telefónica O2. MIS Quarterly Executive, 15(1), 21–35.

[10] McConnell, S. (2004). Code complete: A practical handbook of software construction (2nd ed.). Microsoft Press.

[11] Nolan, R. L., & McFarlan, F. W. (2005). Information technology and the board of directors. Harvard Business Review, 83(10), 96–106.

[12] Ross, J. W., Weill, P., & Robertson, D. C. (2006). Enterprise architecture as strategy. Harvard Business School Press.

[13] Sahay, A., Indamutsa, A., Di Ruscio, D., & Pierantonio, A. (2020). Supporting the understanding and comparison of low-code development platforms. IEEE SEAA, 171–178.

[14] ServiceNow. (2023). Platform governance and upgrade guide. ServiceNow Now Support.

[15] Tallon, P. P., & Pinsonneault, A. (2011). Competing perspectives on the link between strategic information technology alignment and organizational agility. MIS Quarterly, 35(2), 463–486.

[16] Techman, M. (2019). CMDB design and implementation best practices. ITSM Press.

[17] Venkatesh, V., Morris, M. G., Davis, G. B., & Davis, F. D. (2003). User acceptance of information technology: Toward a unified view. MIS Quarterly, 27(3), 425–478.

[18] Weill, P., & Ross, J. W. (2004). IT governance: How top performers manage IT decision rights for superior results. Harvard Business School Press.

[19] Wessel, M., Levina, N., & Schlagwein, D. (2021). Ecosystem governance: A literature review and research agenda. Journal of Information Technology, 36(3), 267–291.

[20] Yin, R. K. (2018). Case study research and applications: Design and methods (6th ed.). Sage.

[21] Zachman, J. A. (1987). A framework for information systems architecture. IBM Systems Journal, 26(3), 276–292.

[22] Zmud, R. W. (1984). Design alternatives for organizing information systems activities. MIS Quarterly, 8(2), 79–93.

[23] Cater-Steel, A., Toleman, M., & Tan, W. G. (2017). Transforming IT service management — The ITIL impact. In Proceedings of the 17th Australasian Conference on Information Systems.