

Platformization as Governance Strategy: Evaluating Cross-Team Licensing Consolidation in Cloud-Native Security Platforms

Santosh Swarna Sathya Chamorthy¹

Abstract: Consolidating fragmented cloud security tooling into one platform is easy to announce as a licensing decision. Fewer SKUs, one codebase, lower total cost of ownership - the pitch writes itself. This article contests that framing. Drawing on a documented enterprise engagement that unified licensing and product roadmap across more than twenty teams contributing to a single cloud-native security platform, and synthesizing literature on organizational coupling, product-line engineering, and platform governance, it argues that the adoption gains credited to platform consolidation are gated by cross-team release governance - the coordination mechanism through which many teams contribute to a shared roadmap without fragmenting it again - rather than by licensing unification on its own. Licensing consolidation enables a unified platform to exist. It does not, by itself, produce the coordination discipline that determines whether that platform gets adopted at scale. The contribution here is a coordination-centric account of platformization that sits alongside the cost- and architecture-centric framing dominant in current platform and product-line scholarship, with direct implications for security-product organizations pursuing similar consolidation.

Keywords: *Platform, consolidation, organizational coupling, cross-team governance, software product lines, release governance, cloud-native security*

1. Introduction

Licensing consolidation is easy to announce but difficult to implement. Unifying SKUs on a spreadsheet takes an afternoon. Getting twenty teams to build against one roadmap instead of twenty is an entirely different problem - and it is the one platformization initiatives most often fail to solve, even when the licensing consolidation itself goes off without a hitch.

Two literatures speak to adjacent pieces of this problem without resolving it together. Product-line engineering research treats platform consolidation as a variability-management problem: structuring a shared codebase so distinct product variants can be derived from it efficiently [1], [2]. Organizational-coupling research, building on Conway's foundational observation that system architecture mirrors communication structure [3], asks how team structure and software architecture shape each other [4], [5]. Neither literature alone explains why an initiative that solves the variability-management problem correctly can still fail to achieve adoption - because neither treats cross-team release governance as the mechanism linking the two.

This article addresses that gap directly, reporting on a

program governance engagement that unified licensing and roadmap contribution across more than twenty teams building on a single cloud-native security platform. Its position: platformization's realized adoption benefit is gated by the release governance mechanism coordinating those teams' contributions - not by the SKU or licensing consolidation that makes a unified platform technically possible in the first place. Licensing consolidation clears a structural obstacle to coordination. It does not create the coordination itself.

The contribution follows from there. Existing platform-consolidation and product-line literature treats architectural unification and organizational coordination as related but separately studied concerns [1], [5]. This article contributes a case-grounded account of the specific mechanism - cross-team release governance - through which architectural consolidation becomes adoption, in a cloud-native security platform context where the stakes of uncoordinated contribution run higher than in a typical internal platform: conflicting security policy changes shipped by different teams carry consequences a typical internal tooling platform does not.

¹Independent Researcher

What follows: Section 2 synthesizes the architecture/cost-centric and coordination-centric literatures around this gap. Section 3 develops the analytical framework and case method. Section 4 reports findings. Section 5 discusses implications for platform-consolidation scholarship. Section 6 closes.

2. Literature Review

2.1. Platform consolidation as an architectural and financial decision

Product-line engineering scholarship formally separates two processes: domain engineering, in which reusable requirements, architecture, and components are built with explicit variation points, and application engineering, in which individual products are derived by binding those variation points to a specific customer's needs, with variability management running as the connective thread across both [1]. The framework generalizes further than its origins in conventional software suggest: one application to synthetic biology mined a repository of more than 45,000 reusable DNA parts and found that feature models captured commonality and variability in up to 93.5 percent of examined products, with derivable product spaces as large as 7.5×10^{20} - evidence that product-line reasoning about variation points is not tied to any particular engineering domain [2]. Platform-strategy research in business and information systems frames the same problem differently: platform firms such as Uber and Airbnb, on this account, win not by owning a value chain the way pipeline businesses do but by orchestrating an external network of producers and consumers, where network effects make scale, not differentiation, the decisive competitive lever [6].

This account holds together on its own terms. It is also silent on what happens once the architectural consolidation is finished - whether the teams expected to build on the unified platform actually do, and at what coordination cost.

2.2. Organizational coupling and team coordination

A separate literature asks how team structure and software architecture shape one another. Conway's original observation - that systems mirror the communication structures of the organizations building them [3] - has since been empirically tested and largely holds up: a natural-experiment comparison found that commercial firms with tightly coupled organizational structures produced measurably more tightly coupled software architectures than loosely coupled open-

source communities building comparable systems, evidence for what the authors term a mirroring hypothesis [4]. More recent empirical work on microservice ownership across 38 open-source projects finds team boundaries and service boundaries align only partially in practice, mediated by specific developer roles - the review's own terms are Connectors, Jacks, and Mavens - that bridge or reinforce organizational coupling depending on how they operate [5]. Team topology research catalogues four organizational patterns: stream-aligned teams that own a feature end-to-end, enabling teams that temporarily uplift another team's capability rather than doing the work themselves, complicated-subsystem teams handling components too specialized to embed elsewhere, and platform teams providing self-service capability to the rest - though the review advancing this taxonomy draws on a notably narrow evidence base of three primary studies, a limitation worth stating plainly rather than treating the typology as settled fact [7]. Empirical work spanning interviews across 31 multinational companies finds DevOps teams cluster into emerging, stable, and consolidated structural patterns along six dimensions including collaboration frequency and ownership sharing, and links these patterns statistically to differences in delivery performance [8]. A longitudinal case study of a large-scale agile transition found that the number of distinct coordination mechanisms teams relied on fell from 27 to 14 after moving to a more structured, second-generation coordination model [9] - a concrete, numeric instance of the pattern this article's case extends into the platformization context specifically: coordination discipline, not architecture or licensing, is what determines whether many-team contribution actually coheres.

This literature runs rich on organizational structure and coordination mechanisms. It has comparatively little to say about licensing or SKU consolidation specifically - the financial, product-packaging dimension that triggers a platformization initiative in the first place.

2.3. The coordination gap

Put together, these two literatures could fully explain platformization outcomes. In practice, they rarely cite each other toward that combination. Product-line and platform-strategy research explains why a unified architecture is desirable and how to build one [1], [6]. Organizational coupling and team topology research explains how coordination mechanisms shape

whether teams actually build together coherently [3], operationalizes the connection. [7], [8], [5]. This article sits at that intersection, using a documented case to trace how release governance

Dimension	Architecture/Cost-Centric Framing	Coordination-Centric Framing
Core question	Is the architecture unified and cost-efficient?	Do teams actually coordinate contributions to a shared roadmap?
Typical success measures	Reduced duplication, lower TCO, unified SKU adoption	Coordination-mechanism count, conflict resolution before production, cross-team roadmap coherence
Blind spot	Assumes unified architecture yields unified contribution	Does not independently address licensing/packaging economics
Representative sources	[2], [6], [8]	[1], [4], [7], [5]

Table 1. Architecture/cost-centric vs. coordination-centric accounts of platformization (author's own synthesis).

3. Methodology and Analytical Framework

The approach is case-based analysis, not a controlled experiment. The case: a documented enterprise engagement pairing licensing and SKU consolidation for a cloud-native security platform with a formal release governance structure coordinating roadmap contributions from more than twenty teams. As with any single-organization case, no control group isolates release governance's causal contribution from that of licensing consolidation itself, or from other concurrent organizational change. What the case does permit is a structured account of adoption outcomes under a documented governance structure, read against the literature synthesized above.

The framework distinguishes two mechanisms:

Unified licensing/SKU consolidation. The packaging and pricing unification removed the structural obstacle of fragmented purchasing paths, making a single platform commercially coherent.

Cross-team release governance. The coordination structure through which teams contributing features to the unified platform align on roadmap sequencing, avoid conflicting changes, and resolve contribution

disputes - distinct from the licensing model itself.

The evaluation examines adoption outcomes following the paired rollout of licensing consolidation and release governance, and separately considers evidence bearing on which mechanism the adoption gain more plausibly traces to. This attribution question cannot be fully resolved by a single case. Section 5 addresses that limitation directly rather than glossing over it.

4. Findings

4.1. Adoption outcomes following licensing consolidation

After previously fragmented SKUs were consolidated into a unified licensing model, adoption of the platform's comprehensive capabilities rose substantially (author's primary program data), consistent with the licensing-driven adoption gains this article's companion analysis of the same program documents from a cost-simplification angle. This tracks with the architecture/cost-centric account: a simplified purchasing path plausibly reduces friction for customers and internal teams alike.

4.2. Release governance as the coordinating mechanism

Licensing consolidation did not operate alone, though. A formal cross-team release governance structure accompanied it, coordinating roadmap contributions from more than twenty teams (Author's primary program data). This structure addressed something licensing consolidation cannot touch: the risk that many teams building independently toward a nominally unified platform would quietly reintroduce the same fragmentation the licensing consolidation was meant to

end - this time at the roadmap level rather than the SKU level. One team's initially proposed feature sequencing conflicted directly with another team's dependency assumptions. The release governance structure surfaced these issues during roadmap review, not during a production incident. That detail illustrates the mechanism's function; it is not a generalizable claim about how often such conflicts occur.

Mechanism	Description	Associated Outcome
Unified licensing/SKU consolidation	Packaging and pricing unification removing fragmented purchasing paths	Substantial increase in comprehensive platform capability adoption
Cross-team release governance	Coordination structure aligning roadmap sequencing across 20+ teams	Feature-sequencing conflicts surfaced during roadmap review rather than production

Table 2. Governance mechanism vs. adoption outcome mapping (Author's primary program data).

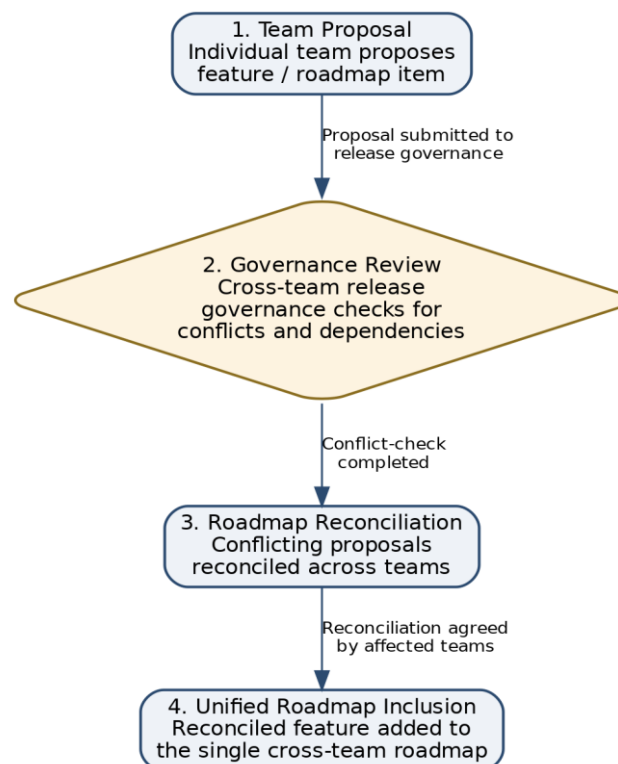


Fig. 2. Cross-team roadmap contribution and gating sequence under platformization (author's own process diagram).

4.3. Attribution considerations

Disentangling the adoption gain's attribution between licensing consolidation and release governance is not fully possible here, since both arrived together. What the case does permit: licensing consolidation created the commercial precondition for a unified platform to exist. Release governance is what determined whether more than twenty teams could contribute to that platform without recreating fragmentation at the roadmap level. Read the adoption figure above as the joint output of both mechanisms. This article's contribution is the argument that governance, not licensing alone, is the harder-to-replicate and more decisive of the two.

5. Discussion

These findings suggest a revision to the typical explanation of platform-consolidation success. The architecture/cost-centric account [1], [6] predicts that licensing and SKU unification, alone, drives adoption by reducing purchasing friction. That prediction is not wrong. It is incomplete in a specific way - it does not explain why some platform-consolidation initiatives with clean licensing unification still fail to achieve coherent team contribution, producing what might fairly be called consolidation in name only: a single SKU wrapped around a roadmap that stays, in practice, fragmented across teams pursuing incompatible priorities. Integration middleware shows the architectural half of that picture: Mittana and Raghupatrin describe a SWIFT MT103 middleware framework that connects a procurement system to the SWIFT payment network, with transaction-integrity and reconciliation controls [10], but such a framework says nothing about how the teams around it coordinate.

The organizational-coupling literature partially explains why this happens, without addressing platformization directly. Conway's Law and its empirical extensions [3], [4] predict that if the organizational structure coordinating a platform's contributors stays fragmented, the resulting system will mirror that fragmentation regardless of whatever licensing model sits on top. Team topology and DevOps structural research [7], [8] catalogues the coordination patterns that determine whether cross-team contribution coheres, but stops short of connecting this to platform-consolidation outcomes directly. This case suggests the missing link: release governance is the mechanism through which an organization avoids the Conway's Law failure mode specifically in the platformization context, by imposing

coordination discipline on roadmap contribution that the licensing model alone cannot supply.

A reasonable counter-position holds that licensing and architectural consolidation are the primary drivers, and release governance is a secondary implementation detail - necessary bureaucracy, not a genuine causal mechanism. This article's position: that counter-position understates how much of platformization's practical difficulty sits specifically in coordination, not architecture or pricing. The architectural and licensing problems in this case were, relatively speaking, tractable. The coordination problem - twenty-plus teams building toward one roadmap instead of twenty - was not, and it required a dedicated governance structure to resolve. That is not a claim that architecture and licensing don't matter. It is a claim about where the marginal difficulty actually sits once those problems get addressed.

This case cannot establish how far the coordination-centric account generalizes beyond cloud-native security platforms specifically - where the cost of uncoordinated contribution runs unusually high, since conflicting security policy changes shipped by different teams carry consequences most internal platforms don't - or beyond organizations with the program management capacity to run formal release governance at this scale.

6. Conclusion

Platform consolidation's adoption benefit is not a direct output of licensing or SKU unification. It is what cross-team release governance produces when it determines whether many teams can build toward a single roadmap without quietly reproducing the fragmentation that the consolidation was meant to end. Licensing unification is a necessary precondition for platformization. Release governance is what makes platformization real instead of nominal. For organizations pursuing similar consolidation in cloud-native security platforms, governance investment deserves at least as much attention as the licensing and architectural work that typically dominates platformization planning.

References

- [1] K. Pohl, G. Böckle, and F. J. van der Linden, *Software Product Line Engineering: Foundations, Principles, and Techniques*. Springer, 2005.

- https://www.researchgate.net/publication/220693462_Software_Product_Line_Engineering_Foundations_Principles_and_Techniques
- [2] M. C. McDevitt, J. Firestone, M. B. Cohen, T. Thianniwet, and W. Niu, "An empirical investigation of organic software product lines," *Empirical Software Engineering*, vol. 26, no. 3, Article 46, 2021. https://www.researchgate.net/publication/350392506_An_empirical_investigation_of_organic_software_product_lines
 - [3] M. E. Conway, "How Do Committees Invent?," *Datamation*, vol. 14, no. 4, pp. 28-31, 1968. <https://www.melconway.com/Home/pdf/committees.pdf>
 - [4] A. MacCormack, J. Rusnak, and C. Y. Baldwin, "Exploring the Duality between Product and Organizational Architectures: A Test of the 'Mirroring' Hypothesis," *Research Policy*, vol. 41, no. 8, pp. 1309-1324, 2012. <https://www.sciencedirect.com/science/article/abs/pii/S0048733312001205>
 - [5] X. Li, D. Amoroso d'Aragona, T. Cerny, V. Lenarduzzi, D. Taibi, and A. Janes, "Exploring microservice ownership and organizational coupling in open-source projects: an empirical study," *Computing*, vol. 107, no. 4, Article 91, 2025. <https://link.springer.com/article/10.1007/s00607-025-01454-7>
 - [6] G. G. Parker, M. W. Van Alstyne, and S. P. Choudary, "Pipelines, Platforms, and the New Rules of Strategy," *Harvard Business Review*, vol. 94, no. 4, pp. 54-62, 2016. <https://hbr.org/2016/04/pipelines-platforms-and-the-new-rules-of-strategy>
 - [7] R. Ahmed and R. Colomo-Palacios, "Team Topologies in Software Teams: A Multivocal Literature Review," in *Computational Science and Its Applications - ICCSA 2021, LNCS* vol. 12951, Springer, 2021, pp. 331-344. https://link.springer.com/chapter/10.1007/978-3-030-87013-3_21
 - [8] D. López-Fernández, J. Díaz, J. García, J. E. Pérez, and Á. González-Prieto, "DevOps Team Structures: Characterization and Implications," *IEEE Transactions on Software Engineering*, vol. 48, no. 9, pp. 3716-3736, 2022. <https://arxiv.org/abs/2101.02361>
 - [9] T. Dingsøyr, F. O. Bjørnson, J. Schrof, and T. Sporse, "A longitudinal explanatory case study of coordination in a very large development programme: the impact of transitioning from a first- to a second-generation large-scale agile development method," *Empirical Software Engineering*, vol. 28, no. 1, Article 1, 2023. https://www.researchgate.net/publication/365209243_A_longitudinal_explanatory_case_study_of_coordination_in_a_very_large_development_programme_the_impact_of_transitioning_from_a_first_to_a_second-generation_large-scale_agile_development_method
 - [10] R. R. Mittana and S. Raghupatrin, "Framework for Integrating Java-Based Procurement Systems with SWIFT Banking Payments for Generating MT103 Format Using Middleware and Message Driven Beans," *International Journal of Intelligent Systems and Applications in Engineering*, vol. 6, no. 2, pp. 170-177, 2018. <https://doi.org/10.17762/ijisae.v6i2.7929>