Engineering Company Culture as a Strategic Capability: Integrating Spotify Engineering Culture Evidence with MIT Culture 500, MIT Systems Thinking, and Harvard Tough-Love Culture Findings

Nikola Petrović[1], Pavle Dakić[2,3,4,5,6,*], Tamara Papić[2,3,4,6] and Ján Lang[4]

[1]Faculty of Technical Sciences, University of Novi Sad, Trg Dositeja Obradovića 6, 21000 Novi Sad, Serbia
[2]Faculty of Technical Sciences, Singidunum University, Danijelova 32, 11000 Belgrade, Serbia
[3]Faculty of Informatics and Computing, Singidunum University, Danijelova 32, 11000 Belgrade, Serbia
[4]Institute of Informatics, Information Systems and Software Engineering, Faculty of Informatics and Information Technologies, Slovak University of Technology in Bratislava, Ilkovičova 2, 842 16 Bratislava, Slovakia
[5]Faculty of Natural Sciences and Mathematics, University of Montenegro, Cetinjski Put bb, 81 000 Podgorica, Montenegro
[6]BabyFM, Veljka Dugoševića 54, 11 050, Belgrade, Serbia

Abstract

This paper presents a conceptual and methodological framework for examining engineering culture as a socio-technical capability linked to delivery performance, reliability, and organizational adaptability. It integrates four lenses: Spotify engineering evidence as an anchor case, MIT Sloan Culture 500 for employee/developer-perspective measurement, MIT systems thinking on culture-process alignment, and Harvard evidence on balanced hard-soft ("tough-love") governance. The framework triangulates three data streams (engineering traces, employee/developer perspectives, and public artifacts) across five mechanisms: autonomy, coordination, psychological safety, platform standardization, and people development. We specify operational indicators, a text-to-mechanism coding scheme, and interpretation boundaries for a future empirical application. All figures are reproduced from public sources or are conceptual visualizations of cited prior evidence, not measurements from data collected by the authors. We derive four results: employee/developer perspectives complement trace metrics; platform standardization supports safer autonomy by reducing coordination cost; culture-process alignment conditions systems-thinking effectiveness; and balanced accountability and support may outperform either extreme. These associative results require validation with longitudinal, multi-organization data. We further propose a new conceptual contribution—the Culture-Process Alignment Loop with Synergistic and Inhibiting Modes (CPAL-SAIM)—that distinguishes reinforcing from inhibiting culture-process dynamics in software-engineering contexts. The contribution is a reusable, transparent protocol for comparative engineering-culture research and a diagnostic guide for practitioners.

Keywords

Engineering Culture, DevOps, Team Autonomy, Spotify Model, Software Delivery Metrics, DORA Framework, Organizational Design

1. Introduction

Engineering company culture has emerged as a strategic capability that can amplify or constrain delivery performance, reliability engineering, and organizational execution. Prior studies in software engineering show consistent links between organizational practices and outcomes such as deployment frequency, lead time, change failure rate, and mean time to restore service, while also emphasizing

SQAMIA 2026: Workshop on Software Quality, Analysis, Monitoring, Improvement, and Applications, September 10–12, 2026, Niš, Serbia
*Corresponding author.
†These authors contributed equally.
nikola.petrovic.frankfurt@gmail.com (N. Petrović); pdakic@singidunum.ac.rs (P. Dakić); tpapic@singidunum.ac.rs (T. Papić); jan.lang@stuba.sk (J. Lang)
0009-0001-7750-4712 (N. Petrović); 0000-0003-3538-6284 (P. Dakić); 0000-0002-9666-7824 (T. Papić); 0000-0002-3271-7271 (J. Lang)
© 2026 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR Workshop Proceedings ceur-ws.org ISSN 1613-0073
published 2026-11-19

CEUR-WS.org/Vol-4336/paper1.pdf

multidimensional productivity constructs beyond output volume [1, 2, 3]. Despite this growing evidence base, engineering culture remains an elusive construct: organizations struggle to measure it systematically, and practitioners often resort to copying structural models (e.g., Spotify's squads and tribes) without understanding the underlying mechanisms that make them effective.

Culture 500 advanced culture measurement by operationalizing employee perspectives at scale through value-based coding of textual signals [4], providing a complementary lens to trace-based metrics. However, it was designed for general organizational contexts and has not been applied to engineering-specific mechanisms such as psychological safety in postmortems, autonomy in deployment decisions, or platform standardization.

Spotify remains a high-visibility anchor case: its organizational model motivated broad adoption attempts, yet evidence repeatedly shows that copying structural labels without underlying mechanisms often fails [5, 6, 7, 8]. Publicly reported Spotify evidence also suggests that internal platform capabilities (e.g., Backstage) function as operational mechanisms that reduce coordination costs and enable safer autonomy, but only when cultural alignment is present [9, 10].

This paper connects four complementary lenses: Spotify engineering evidence as a public anchor case, MIT Culture 500 for employee/developer-perspective measurement, MIT systems thinking on culture-process alignment, and Harvard tough-love findings on governance balance [11, 12]. We model engineering culture as a sociotechnical capability arising from lived employee/developer perspectives, process-platform alignment, and balanced hard-soft governance. A new conceptual contribution is the Culture-Process Alignment Loop with Synergistic and Inhibiting Modes (CPAL-SAIM), presented in Section 3 (Figure 2). The contribution is conceptual and methodological: a comparative framework with explicit mechanisms, measurable indicators, a text-to-mechanism coding scheme, and a transparent protocol for a future public-data study.

The remainder is organized as follows: Section 2 presents the methodology, Section 3 reviews related literature, Section 4 reports results, Section 5 discusses implications and limitations, and Section 6 concludes.

2. Materials and Methods

This conceptual and methodological study specifies a mixed-methods triangulation protocol; it does not execute a new cross-company dataset. The future unit of analysis is an engineering organization (or engineering sub-organization) within a technology company. Culture is operationalized through visible mechanisms and quantifiable indicators, while interpretive context is retained through qualitative artifact analysis. The protocol combines three streams (Figure 1): (1) engineering digital traces (Git commits and pull requests, review latency and participation, CI/CD telemetry, deployment events, and incident records); (2) employee/developer perspectives (review texts and developer-experience surveys); and (3) public engineering artifacts (blogs, handbooks, architecture write-ups, postmortems, and conference talks).

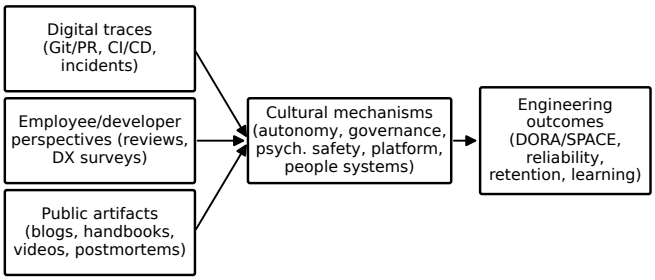


Figure 1: Conceptual triangulation protocol. Engineering traces, employee/developer perspectives (review and survey texts), and public artifacts are coded against the five mechanisms in Table 1; convergent and contradictory signals then bound, rather than prove, propositions about engineering outcomes. The diagram specifies a future analysis workflow and is not derived from new data.

For an empirical execution, the systematic protocol requires a dated source register; inclusion of publicly accessible items that identify an organization, period, and relevant mechanism or outcome; exclusion of duplicate, undated, inaccessible, or purely promotional items; and preservation of provenance after deduplication and entity/time normalization [13]. Culture 500 is used as an operational template for mapping text to Big 9 values and tracking incidence and sentiment, not as a dataset newly re-coded here [4]. The framework maps accountability to hard features and participation/safety to soft features [12], and assesses whether standardized operational assets align with local team culture using the MIT synergistic/inhibiting-loop distinction [11].

To guide this research, we formulate three research questions aligned with our comparative framework:

RQ1: What associations between observable cultural mechanisms (autonomy, coordination, psychological safety, platform standardization, and people development) and software delivery outcomes are supported by the integrated literature and public evidence?

RQ2: Which trace, employee/developer-perspective, and public-artifact indicators comprise a reproducible protocol for comparative engineering-culture analysis without proprietary access?

RQ3: Under what conditions does prior evidence suggest that "Spotify-like" mechanism bundles require governance adaptations during organizational scaling?

2.1. Operationalization: Mechanisms, indicators, and expected signals

Table 1 summarizes how we operationalize engineering culture mechanisms using a combination of trace-based indicators, employee/developer-perspective coding, and artifact evidence.

Table 1
Engineering culture mechanisms and example operationalization.

Mechanism	Operational definition	Quantitative indicators (examples)	Qualitative / employee-perspective evidence
Autonomy & decision rights	Teams can decide locally and ship safely	Deployment frequency; PR lead time; approval gates; on-call rotation ownership	Review language indicating empowerment; descriptions of squads/ownership
Governance & coordination	Routines that align teams without over-centralization	Cross-team PR participation; dependency churn; meeting load proxy	Artifacts on tribes/chapters; decision records; tough-love balance
Psychological safety & learning	Blamelessness + rapid learning loops	MTTR; postmortem completion rate; recurrence rate	Postmortem tone; employee reviews on respect/integrity
Platform standardization	Shared golden paths reduce cognitive load	Build time trends; pipeline success rate; service catalog coverage	Backstage adoption narratives; "how we work" docs
People development & mobility	Skill growth supports scaling	Internal mobility rate; tenure distribution; onboarding time	Reviews on growth; mentoring programs; hiring narratives

2.2. Text-to-Mechanism Coding Scheme

Table 2 specifies the proposed coding scheme for mapping textual signals from employee/developer perspectives and public artifacts to operational mechanisms. In a future empirical application, two researchers should independently code a shared sample, discuss disagreements against the Culture 500 value definitions and mechanism criteria, and report the resulting inter-rater reliability (for example, Cohen's kappa) before adjudication. No new corpus was coded in this paper; therefore, no kappa value is reported.

Table 2
Text-to-mechanism coding scheme with example signals.

Mechanism	Text signal examples (from reviews/artifacts)	Culture 500 value mapping
Autonomy & decision rights	"We own our services end-to-end," "No approval bottlenecks," "Squads decide what to build"	Agility, Innovation
Governance & coordination	"Clear chapter boundaries," "Tribes align dependencies," "Decision records are public"	Collaboration, Results
Psychological safety & learning	"Blameless postmortems," "Safe to fail," "Learning from incidents"	Integrity, Respect
Platform standardization	"Golden paths for deployment," "Backstage catalog," "Standardized CI/CD"	Innovation, Order
People development & mobility	"Internal mobility encouraged," "Mentoring programs," "Growth-focused reviews"	Development, Respect

2.3. Data Sources and Ethics

The public sources named in this paper are candidate inputs for a future empirical execution, not datasets newly collected or processed for the present study. Any execution must document collection dates, query terms, inclusion/exclusion decisions, preprocessing outcomes, and the exact analytic software or model configuration. Public traces and artifacts can be processed with text-mining and quantitative methods; review data additionally require anonymization and compliance with platform terms. Because Culture 500 is based on Glassdoor reviews, review texts are treated as employee/developer perspectives rather than ground truth, with self-selection and temporal-validity limitations made explicit [4].

3. Literature Review

This section reviews the literature on engineering culture, organizational models, software delivery metrics, and the interplay between culture and process, focusing on studies that have demonstrated, empirically or theoretically, how culture shapes software engineering outcomes.

3.1. Measuring and diagnosing culture: from anecdotes to employee-perspective analytics

Earlier culture measurement relied predominantly on surveys and managerial perspectives, which limited comparability and introduced bias. The MIT Sloan Culture 500 project [4] has advanced this agenda by demonstrating that 1.4 million Glassdoor reviews can be analyzed with an AI-assisted human-in-the-loop approach, mapping cultural descriptors to the Big 9 values and enabling incidence- and sentiment-based benchmarking. Their work has shown that employee reviews carry lived-experience dimensions such as autonomy, respect, and integrity, which rarely appear in trace data. Rao and Weintraub [14] have identified six interrelated building blocks (resources, processes, values, behavior, climate, and success) and have shown that organizations tend to overemphasize readily measurable constructs (resources, processes, success metrics) while neglecting people-oriented determinants (values, behavior, climate), a pattern also discussed by Russell [15]. In engineering organizations, this insight has motivated the exploration of platform and process instrumentation alongside social conditions that support experimentation and learning. Aleksić and Dakić [16] have further demonstrated that leadership is a key driver of these people-oriented determinants: in open-source software teams, transformational, mentorship-oriented leadership is associated with higher organizational commitment, lower turnover, and greater job satisfaction.

3.2. Culture-process alignment and collaborative systems thinking (MIT systems engineering lens)

In systems engineering contexts, standardized processes are frequently introduced to manage complexity and codify best practices. Lamb and Rhodes [11] have highlighted the "paradox of standards": they have shown that processes can provide context and specify information flows that enable shared mental models, but that over-specified or culturally misaligned processes can inhibit adaptability and obscure the "big picture". They have also proposed the reinforcing versus inhibiting interaction loops between culture and standardized processes (Figure 2). In the synergistic loop (left panel), culture and standard process reinforce each other: shared values guide process updates, and well-aligned processes strengthen cultural assumptions. In the inhibiting loop (right panel), imposed processes that conflict with local culture create friction and workarounds rather than learning. Translating their lens to software engineering, we treat internal developer platforms, service catalogs, and incident response playbooks as standardized process artifacts whose effectiveness depends on whether engineers perceive them as empowering (synergistic loop) or as control for its own sake (inhibiting loop). This distinction grounds our rationale for studying platform standardization and governance as cultural mechanisms rather than purely technical choices. We label this adapted two-panel visualization the Culture-Process Alignment Loop with Synergistic and Inhibiting Modes (CPAL-SAIM); it is one of the conceptual contributions of this paper.

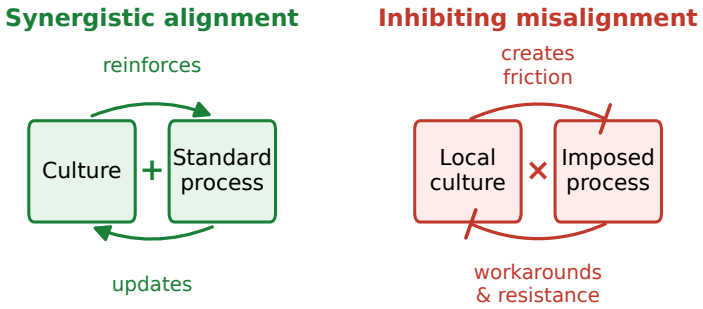


Figure 2: Culture-process alignment loops. Left: synergistic alignment—culture and standard process reinforce each other (open arrowheads, "+" marker). Right: inhibiting misalignment—local culture and imposed process create friction (inhibition bars, "×" marker). Adapted from Lamb and Rhodes [11]; conceptual diagram, not derived from new data.

3.3. Engineering cultures, inclusion, and the long-hours norm

Bagilhole et al. [17] have theorized engineering and SET (science, engineering, technology) cultures as underpinned by individualized, sexualized, and single-gendered norms, and have emphasized the role of long-hours expectations, variable policy-to-practice links, and networking obstacles in inhibiting inclusion and retention. ASCE [18] has documented parallel learning challenges in engineering education. Together, these studies have connected HR-sensitive values to HR-centric attitudes and actions, encouraging the view of diversity and respect as measurable cultural elements that contribute to learning and delivery outcomes. Kunda [19] has argued that, in a practitioner-oriented framing, culture becomes toxic when transparency is low, psychological safety is low, and rules are used to constrain rather than empower people—conditions that can degrade engineering performance through burnout and turnover.

3.4. Engineering Culture and Organizational Models

Skelton and Pais [20] have demonstrated with the team topologies framework how organizational structures should align with cognitive load constraints and Conway's law. Building on this line of work, Forsgren et al. [3] have identified cultural predictors of high-performing technology organizations through multi-year empirical studies. Šmite et al. [5] have documented guild-style knowledge sharing

mechanisms in large-scale agile settings, while Stray et al. [6] have examined the challenges that autonomous agile teams face in maintaining architectural coherence. Dikert et al. [7] have extensively reviewed large-scale agile transformations and have shown that context-blind adoption of models like Spotify's frequently fails. Bass et al. [8] have shown that governance and technical controls in release pipelines must be jointly designed.

3.5. Software Delivery and Reliability Metrics

Forsgren et al. [1] have challenged narrow productivity metrics with the SPACE framework, incorporating satisfaction, performance, activity, communication, and efficiency dimensions across individual, team, and organizational levels. In longitudinal work, Forsgren et al. [3, 2] have established DORA metrics (deployment frequency, lead time, change failure rate, and MTTR) as validated predictors of organizational performance and profitability. Noda et al. [21] have operationalized SPACE in practice, demonstrating measurement strategies applicable across diverse technology organizations. Other studies have examined code review dynamics as leading indicators of collaboration quality: Greiler et al. [22] have revealed links to defect density in industry settings, and Sadowski et al. [23] have documented how review thoroughness at Google correlates with long-term maintainability. On the human side, Storey et al. [24] have established psychological safety as a key mediating variable for developer job satisfaction and perceived productivity, while Meyer et al. [25] have linked flow-state disruptions to delivery performance degradation.

3.6. Empirical Case Studies and Data Sources

Runeson et al. [26] have established methodological foundations for case study research in software engineering, as have Easterbrook et al. [27], both emphasizing triangulation and construct validity. Several high-profile case studies have illustrated how specific companies institutionalized cultural mechanisms: Beyer et al. [28] have shown how Google introduced error budgets as a quantifiable balance between innovation velocity and reliability; Kim et al. [29] have described how Amazon enforced team autonomy architecturally through API-mandated service boundaries; and Basiri et al. [30] have demonstrated that Netflix's controlled failure injection can institutionalize organizational learning. Robbins et al. [31] have discussed resilience engineering principles for learning-oriented operations under failure. On the data-source side, Garousi et al. [13] have provided guidelines for grey literature reviews that legitimize engineering blogs and incident reports as research inputs. Korling [32] has discussed developer experience as a first-class software engineering concern, and Lwakatare et al. [33] have examined DevOps dimensions and their implications for process standardization.

We intentionally retain older foundational studies (1990s–2000s) because they define core constructs still central to engineering-culture research [19, 27, 11]; newer references supply contemporary operationalization and external validity checks.

4. Results

This section presents literature- and public-source-informed results, not new empirical findings. We integrate (a) reported quantitative evidence from public sources, (b) established empirical software-engineering findings, and (c) constructs from MIT and Harvard culture research. The figures are explicitly distinguished as reported, conceptual, or recreated evidence.

4.1. Results Construction and Processing Pipeline

The proposed four-step pipeline is: (1) collect and filter public traces, postmortems, and artifacts in a dated source register [34, 35, 36]; (2) code texts using Table 2 categories and report corpus size and coder reliability; (3) triangulate mechanism claims only when at least two streams converge; (4) estimate

associations only when a harmonized dataset exists. No such dataset was assembled here; the following results do not claim correlations or causal effects [13].

4.2. Hard, soft, and tough-love culture: governance balance and performance (Harvard evidence)

A central debate in management research is whether performance is supported or counteracted by hard (pressure, accountability, direction) and soft (nurturing, participation, warm climate) approaches. Kelman and Hong report interaction effects in cross-organizational collaboration settings that are consistent with "tough love": hard features perform differently in the presence of soft features, while pressure without support can be detrimental [12]. Figure 3 is a recreated visualization of selected values from that prior study; it is not new data and its crime-reduction outcome is not an engineering outcome. For engineering culture, the transferable proposition is limited: psychological safety need not mean an absence of standards, and accountability may be paired with supportive learning routines (for example, reliability standards, error budgets, and learning-oriented postmortems).

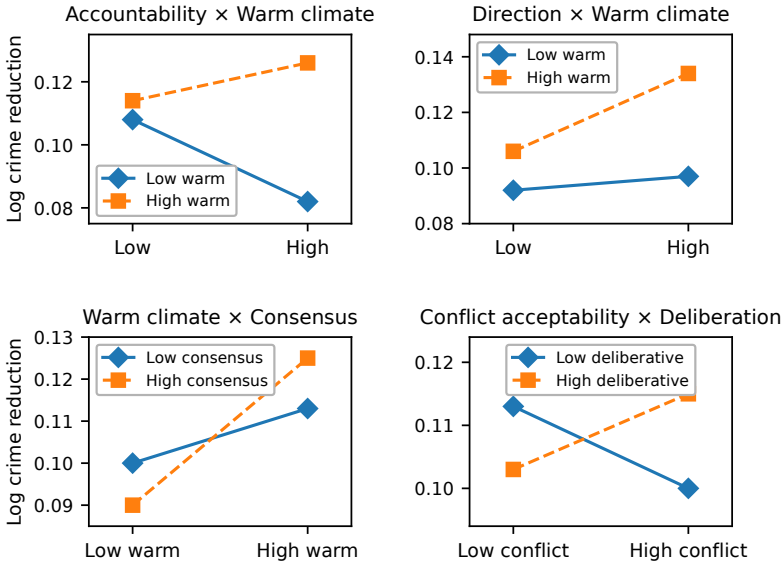


Figure 3: Recreated interaction plots from selected results reported by Kelman and Hong [12]. These visualizations are illustrative, are not derived from new data, and use log-crime-reduction rather than software-engineering outcomes.

Result R1 (employee/developer perspectives complement traces). Culture 500 shows that 1.4 million Glassdoor reviews can be mapped to stable value dimensions using an AI-assisted, human-in-the-loop method [4]. We do not re-code that dataset or apply an AI model in this paper. The proposed application maps narratives about autonomy to agility/innovation and narratives about blamelessness to integrity/respect through Table 2. These texts can complement, but are not equivalent to, trace telemetry because self-selection and temporal bias limit interpretation.

Result R2 (platform standardization can enable safer autonomy). Public Spotify engineering reports associate Backstage-centered workflows and release tooling with lower manual migration effort, automated migration pull requests, and shorter release-cycle latency [9, 10]; Figure 4 reproduces selected reported relative values. These self-reported values were not measured, normalized, or cross-validated by the authors and may reflect reporting bias. They motivate, but do not establish, the proposition that shared platforms can reduce coordination cost and make local delivery decisions safer.

Result R3 (culture-process alignment conditions systems thinking at scale). MIT systems-engineering evidence indicates that standardized process artifacts can support shared mental models and collaborative systems thinking when culture-process alignment is present (synergistic loop, Figure 2 left) but can prompt workarounds and resentment when alignment is weak (inhibiting loop, right) [11]. This theoretical synthesis applied to software platforms, playbooks, and service catalogs requires direct

validation in software-engineering case studies.

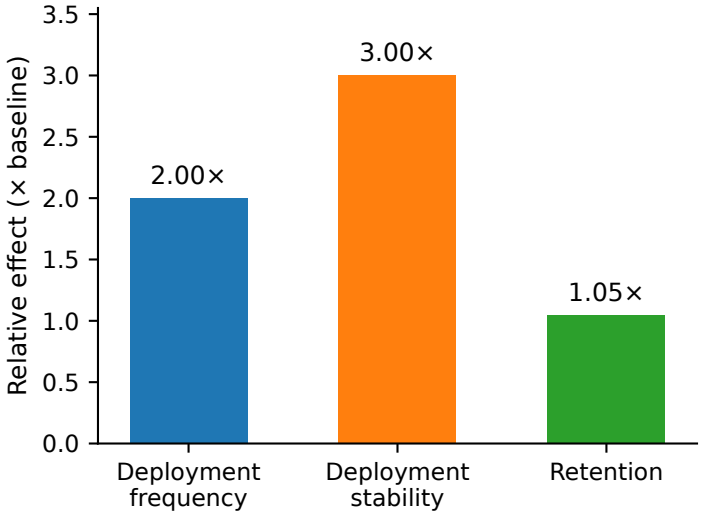


Figure 4: Selected relative impacts reported by Spotify engineering in connection with Backstage usage [9, 10]. This figure reproduces public, self-reported values; it is not derived from a new author-collected dataset and does not establish causation.

Result R4 (balanced governance may outperform extremes). Harvard evidence suggests that interaction between soft and hard cultural features can improve performance under a tough-love configuration (Figures 5 and 3) [12]. Figure 5 is an illustrative normalized surface, not a measurement. Because the original evidence concerns cross-organizational collaboration rather than software teams, the engineering implication is a hypothesis: pair psychological safety with accountability and clear delivery expectations, then test the association with suitable team-level data.

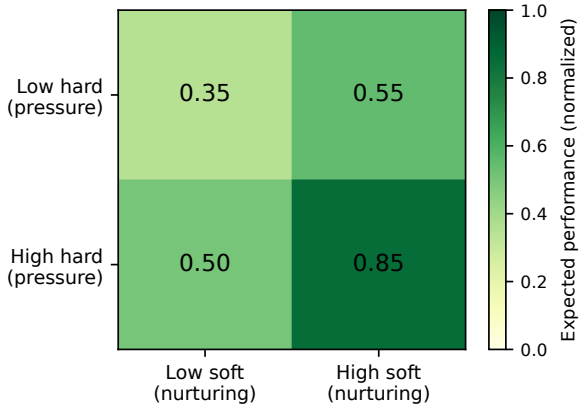


Figure 5: Illustrative normalized tough-love surface, constructed to communicate the hypothesized joint role of hard and soft features. The values are conceptual, not derived from new data or a software-engineering measurement.

4.3. Mapping results to research questions

Regarding RQ1, results R1 through R4 identify associative links between mechanisms and outcomes reported in prior literature and public sources; no new correlation coefficients are computed because no harmonized dataset was assembled. A future empirical execution must estimate these associations with uncertainty after dataset construction. Regarding RQ2, Section 2 and Table 1 specify the indicator set and the triangulation protocol, and the pipeline above defines the steps for a reproducible, proprietary-access-free analysis. Regarding RQ3, results R3 and R4 point to conditions under which Spotify-like bundles may require adaptation: weak culture-process alignment triggers inhibiting loops (R3), and scaling without balanced governance can degrade performance (R4).

5. Discussion

The framework treats engineering culture as an interaction system rather than a static list of values. Culture 500 demonstrates that employee/developer perspectives carry stable cultural signals that complement delivery and reliability indicators [4]. The MIT systems-thinking lens adds an important constraint: platforms and processes are enabling only when they co-evolve with local culture and information-flow needs [11]. Harvard tough-love evidence, in turn, underscores the value of examining accountability and psychological safety as a pair rather than in isolation [12]; evidence from open-source software teams similarly indicates that leaders who combine clear direction with individualized mentorship strengthen commitment and retention [16]. These are conceptual results, not findings from a new empirical evaluation.

What does this mean in practice? The framework offers a bounded diagnostic sequence: select observable indicators for the five mechanisms; collect provenance-preserving traces, employee/developer perspectives, and artifacts; look for convergence and contradiction across streams; and resist treating any single public metric as proof. Spotify's reported Backstage effects motivate this sequence but do not establish causation. Any future empirical application must also address marketing bias in public artifacts, self-selection and temporal bias in reviews, and confounding by company size, domain, and regulation.

5.1. Open Questions

Four open questions guide future work: (1) What causal mechanisms link platform standardization to delivery and reliability outcomes? A difference-in-differences design tracking DORA metrics could isolate platform effects. (2) How transferable are Spotify-like mechanism bundles to regulated industries and smaller organizations? (3) Which measurement designs best validate psychological safety and autonomy as predictors of engineering outcomes? (4) How should organizations balance team autonomy with cross-team governance at scale?

6. Conclusion

This paper presents a comparative engineering-culture framework that draws on MIT Sloan's Culture 500 for employee/developer-perspective measurement, MIT systems-engineering research on culture-process alignment, and Harvard evidence on the interplay of hard and soft cultural features. Spotify serves as the public anchor case. The framework puts forward testable claims about mechanism bundles: safe autonomy enabled by platform standardization, psychologically safe learning paired with accountability, and processes that co-evolve with cultural assumptions. Tables, protocol steps, and explicitly labeled figures are meant as guides for future comparative and longitudinal studies, not as a completed empirical evaluation.

The contribution is a reusable operationalization that links five cultural mechanisms to observable indicators and specifies how to triangulate public traces, employee/developer perspectives, and artifacts. A conceptual contribution is the CPAL-SAIM diagram, which adapts the culture-process alignment loop into distinct synergistic and inhibiting modes for software-engineering contexts. Practitioners can use it to define a diagnostic question, select indicators, retain source provenance, and identify where evidence converges or conflicts; they should not infer causal effects from the illustrative or self-reported values presented here.

Future research should apply the protocol to a documented multi-organization corpus, report data volumes and preprocessing decisions, assess coding reliability, and estimate associations with appropriate uncertainty before considering longitudinal or quasi-experimental causal designs.

Data availability statement

This paper is a conceptual and methodological contribution; no new dataset was collected or analyzed.

Three publicly accessible sources are identified as candidate inputs for future empirical execution: (1) **GitHub Archive** (gharchive.org, 100M+ repositories) for DORA metric extraction [34]; (2) **Public postmortem collection** (github.com/danluu/post-mortems, 300+ incident reports) for qualitative coding of psychological safety and learning loops [35]; and (3) **Spotify Engineering Blog Archive** (engineering.atspotify.com, 100+ articles) as the public-artifact stream for the anchor case [36]. All sources are open and publicly licensed. Any future application must document collection dates, query terms, inclusion/exclusion decisions, and preprocessing outcomes.

Conflicts of interest

The authors declare no conflicts of interest.

Acknowledgments

The work reported here was supported by Erasmus+ ICM 2023 No. 2023-1-SK01-KA171-HED-000148295, the Cultural and Educational Grant Agency of Slovak Republic (KEGA) Model-based explication support for personalized education (Podpora personalizovaného vzdelávania explikovaná modelom) - KEGA (014STU-4/2024) and the Slovak Research and Development Agency under Contract no. APVV-23-0408.

Declaration on Generative AI

The authors did not use any generative AI tools to generate content, text, or ideas in this paper. Grammarly was additionally used for grammar and spell-checking.

References

[1] N. Forsgren, M.-A. Storey, C. Maddila, T. Zimmermann, B. Houck, J. Butler, The space of developer productivity, Queue 19 (2021) 20–48. URL: https://dl.acm.org/doi/10.1145/3454122.3454124. doi:10.1145/3454122.3454124.

[2] N. Forsgren, J. Humble, G. Kim, Accelerate: The Science of Lean Software and DevOps, IT Revolution Press, Portland, OR, 2018. URL: https://itrevolution.com/product/accelerate/.

[3] N. Forsgren, J. Humble, G. Kim, Accelerate: Building and Scaling High Performing Technology Organizations, IT Revolution Press, Portland, OR, 2024. URL: https://itrevolution.com/product/accelerate/.

[4] MIT Sloan Management Review, The research | culture 500, Web article, 2026. URL: https://sloanreview.mit.edu/culture500/research.

[5] D. Šmite, N. B. Moe, G. Levinta, M. Floryan, Spotify guilds: How to succeed with knowledge sharing in large-scale agile organizations, IEEE Software 36 (2019) 51–57. URL: https://doi.org/10.1109/MS.2018.2886178. doi:10.1109/MS.2018.2886178.

[6] V. Stray, N. B. Moe, R. Hoda, Autonomous agile teams, in: Proceedings of the 19th International Conference on Agile Software Development: Companion, XP 2018, ACM, Porto, Portugal, 2018, pp. 1–5. URL: https://dl.acm.org/doi/10.1145/3234152.3234182. doi:10.1145/3234152.3234182.

[7] K. Dikert, M. Paasivaara, C. Lassenius, Challenges and success factors for large-scale agile transformations: A systematic literature review, Journal of Systems and Software 119 (2016) 87–108. URL: https://doi.org/10.1016/j.jss.2016.06.013. doi:10.1016/j.jss.2016.06.013.

[8] L. Bass, H. Holz, P. Rimba, A. B. Tran, L. Zhu, Securing a deployment pipeline, in: 2015 IEEE/ACM 3rd International Workshop on Release Engineering, RELENG 2015, IEEE, Florence, Italy, 2015, pp. 4–7. URL: https://ieeexplore.ieee.org/document/7081818. doi:10.1109/RELENG.2015.11.

[9] Spotify Engineering Blog, Background coding agents: Supercharging downstream consumer dataset migrations (honk, part 4), Engineering blog post, 2026. URL: https://engineering.atspotify.com/2026/4/background-coding-agents-dataset-migrations-honk-part-4/.
[10] Spotify Engineering Blog, How we release the spotify app: A look under the hood (part 2), Engineering blog post, 2026. URL: https://engineering.atspotify.com/2026/2/how-we-release-the-spotify-app-part-2/.
[11] C. T. Lamb, D. H. Rhodes, Promoting Systems Thinking Through Alignment of Culture and Process: Initial Results, Technical Report Paper #061, MIT, 2000.
[12] S. Kelman, S. Hong, Hard, Soft, or Tough Love: What Kinds of Organizational Culture Promote Successful Performance in Cross-Organizational Collaborations?, Technical Report RWP12-005, Harvard Kennedy School, 2012. URL: http://nrs.harvard.edu/urn-3:HUL.InstRepos:8506868.
[13] V. Garousi, M. Felderer, M. V. Mäntylä, Guidelines for including grey literature and conducting multivocal literature reviews in software engineering, Information and Software Technology 106 (2019) 101–121. URL: https://doi.org/10.1016/j.infsof.2018.09.006. doi:10.1016/j.infsof.2018.09.006.
[14] MIT Sloan Management Review, J. Rao, J. Weintraub, How innovative is your company's culture?, Web article, 2013. URL: https://sloanreview.mit.edu/article/how-innovative-is-your-companys-culture/.
[15] M. Russell, How to build – and improve – company culture, Harvard DCE Professional & Executive Development, 2024. URL: https://professional.dce.harvard.edu/blog/how-to-build-and-improve-company-culture/, updated 2025.
[16] M. Aleksić, P. Dakić, The role of transformational leadership in open-source software development: Insights from survey data on enhancing organizational commitment and innovation in AI-driven business environments, SCIENCE International Journal 4 (2025) 107–112. URL: https://doi.org/10.35120/sciencej0404107a. doi:10.35120/sciencej0404107a.
[17] B. Bagilhole, A. Powell, S. Barnard, A. Dainty, Researching Cultures in Science, Engineering and Technology: An Analysis of Current and Past Literature, Technical Report No. 7, UKRC Research Report Series, 2008. URL: https://core.ac.uk/outputs/154296645/.
[18] ASCE Library, Building learning organizations in engineering cultures: Case study, Journal of Management in Engineering 16 (2000) 72–83. URL: https://ascelibrary.org/doi/10.1061/%28ASCE%290742-597X%282000%2916%3A4%2872%29. doi:10.1061/(ASCE)0742-597X(2000)16:4(72).
[19] G. Kunda, Engineering culture: Control and commitment in a high-tech corporation, Organization Science 6 (1995) 228–230. URL: https://pubsonline.informs.org/doi/10.1287/orsc.6.2.228. doi:10.1287/orsc.6.2.228.
[20] M. Skelton, M. Pais, Team Topologies: Organizing Business and Technology Teams for Fast Flow, IT Revolution Press, Portland, OR, 2019. URL: https://itrevolution.com/product/team-topologies/.
[21] A. Noda, M.-A. Storey, N. Forsgren, M. Greiler, Devex: What actually drives productivity?, Communications of the ACM 66 (2023) 44–49. URL: https://dl.acm.org/doi/10.1145/3610285. doi:10.1145/3610285.
[22] M. Greiler, M.-A. Storey, C. Bird, Code reviewing in the trenches: Challenges and best practices, IEEE Software 39 (2022) 34–42. URL: https://doi.org/10.1109/MS.2017.265100500. doi:10.1109/MS.2017.265100500.
[23] C. Sadowski, E. Söderberg, L. Church, M. Sipko, A. Bacchelli, Modern code review, in: Proceedings of the 40th International Conference on Software Engineering: Software Engineering in Practice, ICSE-SEIP '18, ACM, Gothenburg, Sweden, 2018, pp. 181–190. URL: https://dl.acm.org/doi/10.1145/3183519.3183525. doi:10.1145/3183519.3183525.
[24] M.-A. Storey, T. Zimmermann, C. Bird, J. Czerwonka, B. Murphy, E. Kalliamvakou, Towards a theory of software developer job satisfaction and perceived productivity, IEEE Transactions on Software Engineering 47 (2021) 2125–2142. URL: https://doi.org/10.1109/TSE.2019.2944354. doi:10.1109/TSE.2019.2944354.
[25] A. N. Meyer, E. T. Barr, C. Bird, T. Zimmermann, Today was a good day: The daily life of

software developers, IEEE Transactions on Software Engineering 47 (2021) 863–880. URL: https://doi.org/10.1109/TSE.2019.2904957. doi:10.1109/TSE.2019.2904957.

[26] P. Runeson, M. Höst, A. Rainer, B. Regnell, Case Study Research in Software Engineering: Guidelines and Examples, Wiley, Hoboken, NJ, 2012. URL: https://doi.org/10.1002/9781118181034. doi:10.1002/9781118181034.

[27] S. Easterbrook, J. Singer, M.-A. Storey, D. Damian, Selecting empirical methods for software engineering research, in: Guide to Advanced Empirical Software Engineering, Springer, 2008, pp. 285–311. URL: https://doi.org/10.1007/978-1-84800-044-5_11. doi:10.1007/978-1-84800-044-5_11.

[28] B. Beyer, C. Jones, J. Petoff, N. R. Murphy, Site Reliability Engineering: How Google Runs Production Systems, O'Reilly, Sebastopol, CA, 2016. URL: https://www.oreilly.com/library/view/site-reliability-engineering/9781491929117/.

[29] G. Kim, K. Behr, G. Spafford, The Phoenix Project: A Novel about IT, DevOps, and Helping Your Business Win, IT Revolution Press, 2013. URL: https://itrevolution.com/product/the-phoenix-project/.

[30] A. Basiri, et al., Chaos engineering: System resiliency in practice, IEEE Software 38 (2021) 24–31.

[31] J. Robbins, K. Krishnan, J. Allspaw, T. A. Limoncelli, Resilience engineering: Learning to embrace failure, Queue 10 (2012) 20–28. URL: https://dl.acm.org/doi/10.1145/2367376.2371297. doi:10.1145/2367376.2371297.

[32] M. Korling, The developer experience, in: Proceedings of the 7th International Conference on Mobile and Ubiquitous Multimedia, MUM '08, ACM, 2008, pp. 214–214. URL: https://dl.acm.org/doi/10.1145/1543137.1543181. doi:10.1145/1543137.1543181.

[33] L. E. Lwakatare, P. Kuvaja, M. Oivo, Dimensions of devops, in: Agile Processes in Software Engineering and Extreme Programming, volume 212 of *Lecture Notes in Business Information Processing*, Springer International Publishing, 2015, pp. 212–217. URL: https://doi.org/10.1007/978-3-319-18612-2_19. doi:10.1007/978-3-319-18612-2_19, xP 2015.

[34] GitHub Archive Project, Github archive, Web dataset, 2026. URL: https://www.gharchive.org/.

[35] D. Luu, Postmortem database, GitHub repository, 2026. URL: https://github.com/danluu/post-mortems.

[36] Spotify Engineering Blog, Spotify engineering blog feed, RSS feed, 2024. URL: https://engineering.atspotify.com/feed/.