

Developer Well-being, Team Dynamics, and Organizational Sustainability in Fintech Solution Development

Muhammad Ovais Ahmad

ovais.ahmad@kau.se

Department of Computer Science, Karlstad University
Karlstad, Sweden

Abstract

Large-scale agile (LSA) frameworks are increasingly adopted by financial institutions to balance rapid digital innovation with complex regulatory requirements. However, the systemic impact of these frameworks on developer cognitive capacity, psychological safety, and long-term socio-technical sustainability remains under-researched. This paper presents exploratory case study of a Swedish financial services organization navigating a Scaled Agile Framework (SAFe) transformation. We conducted fifteen semi-structured interviews across multiple organizational roles and analyzed the data using reflexive thematic analysis. Our analysis identify six recurring socio-technical tensions shaping developer experience and team functioning. Rather than representing isolated challenges, these tensions form an interacting system. We synthesize these findings to explain how coordination intensity, governance ambiguity, and delivery pressure recursively reinforce cognitive overload, defensive team behavior, and socio-technical debt accumulation. We argue that developers well-being is a leading indicator of socio-technical sustainability, not a peripheral human resources concern.

CCS Concepts

• **Software and its engineering**; • **Software creation and management**; • **Software development process management**; • **Software development methods**; • **Agile software development**

Keywords

Agile, SAFe, psychological safety, team dynamics, socio-technical debt, fintech

ACM Reference Format:

Muhammad Ovais Ahmad. 2026. Developer Well-being, Team Dynamics, and Organizational Sustainability in Fintech Solution Development. In *34th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (FSE Companion '26)*, July 5–9, 2026, Montreal, QC, Canada. ACM, New York, NY, USA, 6 pages. <https://doi.org/10.1145/3803437.3805549>

1 Introduction

Agile methodologies have fundamentally reshaped software engineering by optimizing delivery cycles, product quality, and team

engagement in small, co-located settings. However, as organizations grow and the number of teams requiring coordination has increased, large-scale agile (LSA) frameworks (e.g., Scaled Agile Framework (SAFe), Large-Scale Scrum (LeSS)) have emerged to address challenges of scaling agile in enterprise contexts [2, 4, 13]. LSA frameworks introduce organizational structures, governance mechanisms, and coordination processes designed to manage dependencies across teams while preserving agile values. Yet human impact of these mechanisms remains understudied. While research has documented improvements in delivery metrics and process efficiency following LSA adoption [2, 13, 14], less attention has been paid to how scaling structures influence developer cognitive load, psychological safety (PS), and long-term sustainability of development work. This gap carries direct implications for responsible software engineering (RSE). RSE frames the human, social, and ethical dimensions of software development as engineering concerns, not merely managerial ones [cf. [18–20]]. RSE requires organizations to account for the conditions under which software is made, not only the artifacts it produces.

Studies on developer well-being in software engineering has documented concerns about stress, burnout, and work-life balance [15, 22]. Organizational research has established relationships between team stability, PS, and both performance outcomes and employee satisfaction [8, 12]. However, these human-centric perspectives remain insufficiently integrated with LSA research. Understanding these relationships is important for organizational performance. Research demonstrated that team dysfunction and burnout act as primary stressors that negatively impact code quality and retention [18, 21]. Consequently, these human factors determine long-term sustainability of LSA transformations

Our study addresses this gap through exploratory case study of a large Swedish organization. We investigate how coordination mechanisms required for LSA impact developer well-being, team dynamics, and organizational sustainability within the fintech sector. Rather than treating scaling challenges as isolated issues, we examine how they form interacting socio-technical tensions shaping developer well-being, team cohesion, and organizational sustainability. The primary contribution of this work is descriptive and exploratory. Rather than testing specific hypotheses, we sought to understand developers and team leaders experience in a LSA context, with particular attention to factors affecting their capacity for focused work, team cohesion, and sustainable pace. A secondary contribution is an organizing taxonomy of six socio-technical tensions that helps practitioners and researchers reason about developer well-being risks in SAFe transformations.



This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License.

FSE Companion '26, Montreal, QC, Canada

© 2026 Copyright held by the owner/author(s).

ACM ISBN 979-8-4007-2636-1/26/07

<https://doi.org/10.1145/3803437.3805549>

2 Background

Fintech solutions development represents a unique intersection of rapid digital innovation and stringent institutional constraints. Unlike general software engineering, fintech is characterized by high-stakes regulatory compliance (e.g., PSD2, GDPR), the necessity for 24/7 system resilience, and integration of modern microservices with legacy banking cores. These factors necessitate robust governance models that can scale across hundreds of developers without compromising security or auditability. As organizations transition from small-team to enterprise-wide agility, they frequently adopt LSA frameworks. Rooted in Information Processing Theory [10], LSA frameworks are designed to reduce task uncertainty and manage inter-team dependencies through synchronized cadences (e.g., Agile Release Trains and Program Increment Planning (PI)). However, these formal mechanisms can introduce socio-technical debt (STD). While intended to streamline communication, the density of synchronized meetings can paradoxically increase cognitive load and effect the technical autonomy that agile promised [3, 8].

The psychological state of developer is a critical, yet often overlooked, component of software organizational sustainability. Successful engineering requires reaching a “flow state” meaning that a period of deep cognitive immersion essential for solving complex architectural problems [7]. Recent empirical studies [19, 20] indicate that developer well-being is highly sensitive to environmental stressors, including constant context-switching and “meeting fatigue.” In poorly scaled agile environments, the push for high-velocity delivery often compromises a sustainable pace, leading to burnout. Burnout in software teams is not merely a human resources concern; it is a technical risk that correlates directly with increased technical debt, reduced code quality, and high turnover rates [3, 19, 20].

Organizational psychology highlights that team stability and personality compatibility are stronger predictors of performance than individual technical aptitude [16]. Stable teams develop Shared Mental Models, which allow for coordination and faster problem-solving [23, 24]. A core tension in LSA arises when top-down organizational restructuring or rigid role assignments disrupt these established team dynamics. Central to this stability is “PS”. It is defined as the belief that one will not be punished for admitting mistakes [3, 11]. In large-scale transformations, increased hierarchical oversight and delivery pressure can inadvertently erode this safety, stifling the transparency required for genuine agile functioning. Despite these known links, there remains a significant research gap in understanding how specific LSA rituals (e.g., PI Planning and standardized reporting) impact long-term mental health and cohesion of engineering teams in the fintech sector, how these dynamics can be understood and addressed within RSE frame.

3 Methodology

This study employs a qualitative single-case study design, an approach suited for investigating complex, contemporary phenomena within their real-world context [25]. Our study adopts a critical realist stance. Real organisational structures generate causal mechanisms that shape developer experience, even if these are accessible only through participant interpretation and language. This allowed us to treat participant accounts as meaningful evidence of socio-technical dynamics, while remaining alert to the partial and situated

nature of each individual’s perspective. Our theoretical scaffolding draws on socio-technical systems theory, Information Processing Theory, and literature on developer well-being [3, 10, 18–20]. This helps during data collection and analysis, particularly around coordination demands, autonomy distribution, and PS as systemic rather than individual properties. This follows reflexive thematic analysis [6] concept which is flexible but not theoretical.

The research site is a Swedish financial services organization that underwent a comprehensive SAFe implementation approximately five years before data collection. At the time of our study, organizational structure comprised approximately 40–50 development teams organized into various agile release trains. We maintained an ongoing orientation toward how our prior research influenced the analysis. We had no prior relationship with case organisation and used analytic memos to identify and bracket potential confirmation bias regarding known LSA tensions. This process ensured openness to disconfirming data (e.g., participant accounts of useful governance structures), which provided the nuance for ambivalent experiences documented in Tensions 4 & 6.

We conducted fifteen semi-structured interviews across various organizational roles (developers, Scrum Masters, Product Owners, managers, UX designers, and testers). Participants were purposively selected to capture diverse perspectives on coordination, governance and team functioning. This purposive sampling strategy was deliberately used to capture a multifaceted range of perspectives across software development lifecycle. Consistent with Braun and Clarke [6] data were analyzed using reflexive thematic analysis [6]. We treat themes not as entities discovered in data but as actively constructed through the researcher engagement with data. Themes represent shared meaning, underpinned by a central organizing concept. This means the tensions reported reflect our work, shaped by theory and reflexivity. The analytic process involved familiarization, inductive open coding, iterative clustering into conceptual categories, and refinement into higher-order themes. Approximately 140 initial open codes were consolidated into 18 conceptual categories and ultimately six systemic tensions. At each stage, candidate themes were compared and contrasts identified. Memos were used to ensure analysis remained grounded in participant data.

4 Findings

SAFe implementation in case organization has resulted in significant systemic friction. Our thematic analysis showed that coordination mechanisms intended to streamline enterprise agility have instead produced six core tensions that impact developer well-being and organizational sustainability.

4.1 Coordination Overhead Versus Developer Focus Time

LSA core challenge is managing dependencies across multiple teams working on the same product. The case organization used multiple mechanisms for coordination. Developers and team leads spend substantial portions of their work time in meetings (e.g., daily stand-ups, weekly team leader meetings, bi-weekly sprint planning and retrospectives, monthly community of practice meetings, and quarterly PI planning events spanning multiple days). The cumulative

Table 1: Primary Themes, Definitions, and Exemplary Quotes

Primary Theme (Tension)	Representative Codes	Definition	Exemplary Quote
1. Coordination Overhead vs. Focus Time	Meeting load; Context switching; Fragmented day	Time spent in meetings and ceremonies that fragments focus time	"The problem for team is we are doing so many things at the same time."
2. Organizational Structure vs. PS	Authority ambiguity; Defensive practices; Manager interference	Unclear decision authority leading to reduced confidence and protective behaviors	"Sometimes they want to hijack your retrospective... it's ours."
3. Team Composition vs. Personality Fit	Technical hiring; Interpersonal conflict; Skill vs. fit	Hiring prioritizes technical skill while underweighting personality fit	"There is a very skilled person who everybody else thinks is an a**hole."
4. Selective Autonomy vs. Intrinsic Motivation	Autonomy boundaries; Unpredictable control	Teams experience high autonomy in some areas but constraints in others	"Selective autonomy creates tension about where the boundaries are."
5. Delivery Pressure vs. Sustainable Pace	Time pressure; Deferred coordination; Rework	Aggressive schedules lead to deferring coordination work to meet deadlines	"We forget some dependency... or we will not be able to finish it in time."
6. Process Standardization vs. Contextual Fit	Process mismatch; Discovery vs. delivery; Design marginalization	SAFe processes optimized for implementation do not fit discovery work	"The whole process... it's rigged for development. UX works differently."

cognitive load of such events often overwhelmed developers capacity for deep work. A senior Scrum Master expressed that the issue is not the duration of individual meetings, but complexity of parallel work streams: *"When we get eight or nine or ten people and then of course, stand-up and everything takes a bit longer time, but that's not big problem from my point of view. The problem for team is we are doing so many things at the same time. It's difficult to manage. To understand what is happening and where and how will it affect our system in the end, then there are too many things that we need to know. Too complex, too complex."* This suggests that when coordination requirements exceed cognitive bandwidth, developers struggle to maintain a coherent mental model of the product, leading to a fragmented development experience.

4.2 Organizational Structure Versus Psychological Safety

The organization maintained a dual-governance model, where agile roles (POs, Scrum Masters) operated alongside traditional line management. This overlap created ambiguity regarding decision-making authority. For instance, Scrum Masters reported instances where line managers attempted to "hijack" retrospectives to enforce top-down mandates. Such hijacking actions on team-owned spaces forced teams to adopt defensive postures to protect their autonomy. A Scrum Master highlighted: *"Sometimes they want to try to hijack your retrospective to add it, to tell you what you should do during this retrospective. Then you need to be calm and say 'No, no. The retrospective is ours. It's totally our team members' session.'" "... they act as they have their own structure, totally different from the SAFe structure. Sometimes you need to step on your toes and be so it could be difficult."* Such boundary ambiguity erodes PS, as teams feel their internal processes are subject to arbitrary external interference.

4.3 Team Composition Based on Skills Versus Personality Fit

Software engineering hiring practices typically emphasize technical skills (e.g., programming ability, technical knowledge, problem-solving capability). The case organization was no exception. However, team functioning depends not solely on individual technical competence but also on interpersonal compatibility and team composition dynamics. Our findings indicate that interpersonal

compatibility is a significant predictor of team stability. A team lead recounted a messy period where hiring of a highly skilled but toxic developer led to attrition of three experienced members. *"..... last half year have been very messy for the team, and three people have decided to leave the team because they don't get along with each other. There is a very skilled person who everybody else thinks is an asshole."* *"The most difficult thing is to put people together with their personalities, so they work together. That's very difficult. And it's very clear now and have been clear for a long time: We are not specialists. We are not good at this. To take that aspect into consideration."* This tension highlights a gap in LSA implementation that it offer little guidance on socio-psychological team formation. This leads to high performing individuals inadvertently destabilizing collective team dynamics.

4.4 Selective Autonomy Versus Intrinsic Motivation

Autonomy in the case organization was found to be context-dependent. Teams reported autonomy over local technical decisions (e.g., refactoring approaches, design choices, and sequencing of technical debt work), while experiencing constraints in other areas (e.g., release timing, cross-team dependencies, and governance requirements). A developer explained: *"We are autonomous, you could say. We make the decision in the team. It's kind of known by the company that it is this way. When you are supposed to deliver something that we promised to deliver 15th of March, and 80 percent of the code has to be tested, and time pressure means we say, 'You have to do it'. OK, we won't do it before the release, but you have to make a plan to do it later."* This highlights that teams do not have equivalent autonomy and such selective autonomy perceived to create psychological tension.

4.5 Delivery Pressure Versus Sustainable Pace

Organizations face pressure to deliver functionality to customers. However, when delivery pressure leads to decisions that defer necessary coordination, this creates STD [2]. For example, dependency identification and communication is essential for coordinating work across teams. However, when time pressure is intense, teams may defer this work to focus on implementation. A team lead described: *"Sometimes we forget some dependency or we misunderstand some things with other team. It was yesterday. It was like 'You are supposed*

to be done by Monday, but if we are supposed to release at this date, we needed to have it on this date or we will not be able to finish it in time.” This pattern creates a vicious cycle where short-term delivery gains are negated by high cost of rework, ultimately threatening the long-term sustainability of development pace.

4.6 Process Standardization Versus Contextual Appropriateness

SAFe provides a comprehensive set of prescribed practices and processes intended to create consistency across teams. However, our findings showed a persistent tension between centralized process and unique operational requirements of teams. While the organization sought one way of working to simplify management, developers reported that rigid standardization failed to account for technical diversity across different agile release trains. This mismatch was particularly evident in how UX design work was integrated into the coding work. A UX designer described the mismatch: *“The whole process as I see it now, maybe it’s not like that in original SAFe form, but it’s rigged for development. Development has such a solid structure of meetings. The way UX works and the way developer works is like we can’t really have everything on same board.”* *“User testing for example, where would that happen here? Do we do that a part when it comes to acceptance test or where, when should I plan for that? Or like, ‘no, that is not part of our work right now.”* This one-size-fits-all approach often leads to ceremonial compliance, where teams perform rituals of the framework without deriving intended value. The underlying issue is that standardization, while beneficial for high-level organizational visibility, can act as cognitive straitjacket for teams dealing with highly specialized fintech challenges (e.g., legacy integration or high-frequency transaction processing). However, long-term sustainability requires a balance. Enough standardization to ensure cross-team alignment, but sufficient contextual flexibility to allow teams to optimize for their specific technical and domain-driven aspects of work.

5 Discussion

Our findings show that LSA frameworks can provide a structural roadmap for enterprise coordination but they introduce systemic friction that specifically targets psychological and cognitive resources of developers. We discuss these findings through three primary lenses: coordination complexity, structural ambiguity, and socio-technical sustainability.

Our findings extend information processing view through Tension 1 (Coordination vs. Focus) and Tension 5 (Pressure vs. Pace) [10]. Participants accounts show that when coordination overhead exceeds a certain level, it hampers the ability to do quality software work. Coordination overhead is related to the challenge of maintaining shared understanding when teams are large and work is complex; larger teams face greater difficulty achieving shared understanding. Contingency theory in organizational behavior suggests that organizational structures should align with task requirements [9]. From this perspective, LSA software development creates task interdependencies that require coordination. SAFe structures and processes are designed to manage these interdependencies. However, contingency theory also suggests that when organizational

structures create inefficiencies greater than the interdependencies they address, the structure should be reconsidered.

We suggest the existence of cognitive coordination ceiling that extends information processing theory [10]. While increasing synchronization initially reduces uncertainty, beyond a threshold it fragments developer attention, increases context switching, and diminishes deep-work capacity. When the volume of parallel work streams and synchronized rituals exceeds developers’ cognitive bandwidth, shared mental models deteriorate rather than improve. From an RSE perspective, a cognitive coordination ceiling is a well-being concern. Organisations that design structures that routinely exceed cognitive bandwidth are, in effect, creating structures that produce stress and fragmentation. Literature argues that such conditions require active organizational redress [18, 22].

Participants linked frequent coordination rituals and parallel work streams to difficulty achieving sustained focus required for complex engineering tasks such as architectural analysis and reliability improvements. Such fragmentation prevents developers from achieving “flow state” [7]. Czerwinski et al. [7] describes flow is important for sustained attention on complex work. This suggests that scaling mechanisms intended to enhance predictability may paradoxically undermine cognitive effectiveness when coordination density surpasses human processing capacity. Moreover, coordination overload does not remain isolated. Fragmented attention increases mistakes likelihood and rework, which in turn generates additional coordination demands. This reinforcing dynamic constitutes a coordination overload loop, where synchronization intended to reduce uncertainty progressively amplifies it.

Tension 2 and 4 highlight a gap in current LSA literature. Our results indicate that simply overlaying SAFe onto existing hierarchy creates dual-governance paradox. This misalignment directly threatens PS [3, 8]. PS is enhanced by clear authority structures, transparent decision processes, and leadership behaviors that signal openness to input and error [1, 8]. In the case organization, line management occasionally intervened in team-owned spaces (e.g., retrospectives). Such actions were perceived as interference which prompting defensive behavior. Our findings indicate that structural ambiguity can trigger governance safety loop. Ambiguous authority leads teams to adopt protective practices to preserve autonomy. Defensive behavior reduces transparency and willingness to surface problems, weakening PS. Reduced openness increases hidden coordination failures, which may prompt further managerial intervention, thereby reinforcing perceptions of control and diminishing trust. This dynamic suggests that PS in LSA is not solely an interpersonal phenomenon but structurally mediated. Even formally empowered teams may experience diminished safety when governance boundaries are unclear or inconsistently enacted. Team autonomy is not a binary state but a fragile equilibrium easily disrupted by external organizational interference [17]. Shared understanding, PS and task interdependence are key factors for team effectiveness [12]. In fintech sector, where regulatory pressure often drives command-and-control mindset, protecting these autonomous spaces is vital for maintaining intrinsic motivation.

Tension 3 and 6 demonstrate that LSA frameworks are often implemented as technical or structural solutions while underemphasizing socio-psychological dimensions of team functioning. The

attrition of three experienced members due to “personality fit” issues underscores that technical proficiency is necessary but insufficient condition for team effectiveness. Developer well-being is a leading indicator of project health [5, 19, 20]. Similarly, rigid standardization may optimize reporting visibility while marginalizing context-specific work practices, particularly in discovery-oriented roles such as UX design. These tensions connect directly to a broader pressure sustainability loop. Delivery pressure encourages teams to defer coordination activities and relational maintenance work in favor of short term implementation goals. Deferred coordination accumulates STD, increasing instability and rework. Escalating instability heightens delivery pressure and burnout risk, ultimately effecting team cohesion and retention. Developer well-being thus emerges not as a peripheral human-resources issue but as a leading indicator of socio-technical sustainability. Burnout and defensive behaviors collectively degrade long-term adaptive capacity.

In a nutshell, all these six tensions forming an interacting socio-technical system rather than a set of independent organizational trade-offs. Our analysis indicates that these tensions operate through three reinforcing feedback dynamics. (1) *Coordination Overload Loop* emerges when increased inter-team dependencies require greater synchronization, which fragments cognitive focus, reduces deep-work capacity, and increases errors and rework. This resulting into more coordination demand. (2) *Governance Safety Loop* arises when ambiguous authority structures prompt defensive team behavior, reducing transparency and PS; diminished openness increases hidden coordination failures. This leads to more managerial intervention and reinforce perceived loss of autonomy. (3) *Pressure Sustainability Loop* emerge when delivery pressure leads teams to defer coordination and relational maintenance work and contribute to STD. This results into increase in delivery pressure and burnout risk. Collectively, these reinforcing dynamics demonstrate that scaling decisions simultaneously shape cognitive load, relational climate, and motivational stability. Sustainable LSA requires balancing coordination intensity, governance clarity and autonomy distribution, rather than optimizing any single dimension in isolation.

5.1 Limitations

This study is based on a single organization in specific Fintech context. The identified tensions may reflect characteristics particular to fintech services, where regulatory requirements and risk-averse culture create particular pressures for governance and control. Similar analyses in technology companies or government agencies might identify different tensions or different severity levels for the identified tensions. Additionally, our analysis is descriptive rather than causal. The tensions are constructed through qualitative data, but data do not enable ruling out alternative explanations. For example, low PS might result from team composition issues or leadership behaviors rather than from the dual governance structure. Rigorous causal analysis would require comparative designs (teams with clear authority structures versus ambiguous structures) or longitudinal designs tracking changes in authority structure and PS over time. Reflexivity was enacted through analytic memos and iterative engagement with disconfirming data rather than through inter-analyst triangulation. This is consistent with reflexive thematic analysis positioning of researcher as the instrument of analysis [6],

but findings should be interpreted with awareness of this single-analyst context. Organizational transformation typically requires longer periods to reach full stabilization, and longer-term follow-up would be needed to assess whether initial changes sustained.

6 Conclusion

This exploratory case study identified six critical tensions that emerge in LSA context: coordination overhead versus developer focus time, organizational structure versus PS, team composition versus personality dynamics, selective autonomy versus intrinsic motivation, delivery pressure versus sustainable pace, and process standardization versus contextual appropriateness. These tensions reflect fundamental challenges in scaling agile to large organizations. LSA frameworks need to coordinate work across teams while maintaining favorable conditions for individual cognitive work and team functioning. The tensions are not problems to be solved but rather inherent trade-offs that must be managed. Organizations implementing LSA frameworks accept substantial coordination overhead to manage dependencies across teams. The challenge is managing such overhead thoughtfully to preserve sufficient focus time for development work. Organizations necessarily establish governance structures to coordinate work; the challenge is ensuring clarity about authority and decision-making to preserve PS.

Theoretically, our study first, extends information-processing theory; second, explains how structural ambiguity mediates PS in LSA environments. Third, positions developer well-being as a leading indicator of socio-technical sustainability. Fourth, explaining interactions between coordination, governance, and delivery dynamics rather than isolated scaling trade-offs. Sustainable LSA depends on managing systemic tensions. Our findings contribute to understanding of how organizational scaling decisions influence both individual developer experiences and team-level outcomes. Future research should investigate these tensions across multiple organizations, industries, and LSA frameworks to establish analytical generalizability and identify factors that influence how different organizations navigate these tensions.

Practitioners should prioritize coordination integrity over isolated optimization. To protect deep-work capacity, organizations must monitor coordination loads and treat unsustainable loads as a design failure requiring structural intervention, not individual resilience. Clear decision authority is essential for PS. Furthermore, team stability should be viewed as a strategic and ethical asset. Hiring processes should incorporate structured assessment of interpersonal compatibility alongside technical competence. Finally, standardized frameworks must allow for local, contextual adaptation. This is particularly important for discovery-oriented roles such as UX design, whose contributions are essential to responsible, user-centered software development but whose work rhythms are poorly served by implementation-oriented process structures.

Acknowledgments

This work was supported by Helge Ax:son Johnsons Stiftelse, and Swedish Foundation for International Cooperation in Research and Higher Education (IB2020-8720). During the preparation of this work the author used DeepL to improve the language.

Table 2: Tension Framework Matrix with Potential impacts

Tension	Manifestation	Potential impact on Work	Affected Stakeholder
Coordination Overhead vs. Developer Focus Time	8–10 recurring meetings/week; ad hoc cross-team dependency syncs; escalations	Cognitive overload; fragmented focus time; reduced deep work blocks; context switching fatigue	Developers (primary), Team Leads, Architects, Product Owners
Organizational Structure vs. Psychological Safety	Dual governance: agile roles + line management; ambiguous decision authority; conflicting signals	Reduced autonomy perception; teams adopt defensive practices (protecting spaces from managers); hesitation to raise concerns	Teams, Scrum Masters, Team Leads, Product Owners
Skill-Based Hiring vs. Personality Fit	Senior developer hired for technical expertise; interpersonal conflicts emergent over 6 months; personality dynamics unassessed in hiring	Team departure (experienced members), attrition, knowledge loss, reduced team velocity, damaged team cohesion	Teams, HR, Organization (retention)
Selective Autonomy vs. Intrinsic Motivation	Autonomy for technical debt trade-offs; constrained autonomy on other decisions; unpredictable boundaries	Psychological tension; reduced sense of ownership; ambiguous accountability; adaptation fatigue	Developers, Team Leads, Managers
Delivery Pressure vs. Sustainable Pace	Aggressive sprint/release schedules; deferred dependency communication to meet deadlines; insufficient slack	Late discovery of unmanaged dependencies; rework; accumulating “coordination debt”; burnout risk; reduced quality	Developers, Product Owners, Architects, Release Planning, Organization
Process Standardization vs. Contextual Fit	SAFe processes optimized for code-implementation work; discovery work (design, UX research) proceeds informally and invisibly	Design work lacks organizational visibility; coordination gaps between design and development; knowledge silos; reduced design input to strategy	UX/Designers, Product Owners, Development Teams, Architects

References

- [1] Muhammad Ovais Ahmad. 2023. Psychological safety, leadership and non-technical debt in large-scale agile software development. In *18th FedCSIS*. IEEE, 327–334.
- [2] Muhammad Ovais Ahmad. 2025. Illusion of Agile Scaling: Large-Scale Scrum and Scaled Agile Framework Insights from Two Swedish Companies. *IT Professional* 27, 6 (2025), 71–80.
- [3] Muhammad Ovais Ahmad. 2025. Strengthening large-scale agile teams: the interplay of high-quality relationships, psychological safety, and learning from failures. *Journal of Software: Evolution and Process* 37, 1 (2025), e2759.
- [4] Muhammad Ovais Ahmad, Hadi Ghanbari, Tomas Gustavsson, and Bikesh Raj Upreti. 2025. It all starts with structure: investigating learning dynamics in large-scale agile software development. *Journal of Systems and Software* 230 (2025), 112561.
- [5] Muhammad Ovais Ahmad and Tomas Gustavsson. 2024. Social Sustainability and Large-Scale Agile Software Development. In *ISD2024 Proceedings*, B. et al. Marcinkowski (Ed.). <https://doi.org/10.62036/ISD.2024.98>
- [6] Virginia Braun and Victoria Clarke. 2019. Reflecting on reflexive thematic analysis. *Qualitative Research in Sport, Exercise and Health* 11, 4 (2019), 589–597.
- [7] Mary Czerwinski, Eric Horvitz, and Susan Wilhite. 2004. A diary study of task switching and interruptions. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. 175–182.
- [8] Amy C Edmondson. 2012. *Teaming: How organizations learn, innovate, and compete in the knowledge economy*. John Wiley & Sons.
- [9] Fred E Fiedler et al. 1974. The contingency model: New directions for leadership utilization. *Journal of contemporary business* 3, 4 (1974), 65–79.
- [10] Jay R Galbraith. 1995. *Designing organizations: An executive briefing on strategy, structure, and process*. Jossey-Bass/Wiley.
- [11] Lucas Gren. 2017. *Psychological group processes when building agile software development teams*. Göteborgs universitet.
- [12] J Richard Hackman and Ruth Wageman. 2005. A theory of team coaching. *Academy of management review* 30, 2 (2005), 269–287.
- [13] Pavithra Herath and Muhammad Ovais Ahmad. 2025. A case study on decoding human factors and socio-technical debt in large scale agile project. In *Proceedings of 29th Conf. on Evaluation and Assessment in Software Eng. Companion*. 1–9.
- [14] Pavithra Herath, Muhammad Ovais Ahmad, and Tomas Gustavsson. 2025. Product Guardian Role and Socio-Technical Debt Management in Large-Scale Agile. In *Proceedings of the 2025 29th International Conference on Evaluation and Assessment in Software Engineering Companion*. 127–135.
- [15] Bin Lin, Nathan Cassee, Alexander Serebrenik, Gabriele Bavota, Nicole Novielli, and Michele Lanza. 2022. Opinion mining for software development: a systematic literature review. *TOSEM* 31, 3 (2022), 1–41.
- [16] John E Mathieu, Tonia S Heffner, Gerald F Goodwin, Eduardo Salas, and Janis A Cannon-Bowers. 2000. The influence of shared mental models on team process and performance. *Journal of applied psychology* 85, 2 (2000), 273.
- [17] Nils Brede Moe, Torgeir Dingsøy, and Tore Dybå. 2010. A teamwork model for understanding an agile team: A case study of a Scrum project. *Information and software technology* 52, 5 (2010), 480–491.
- [18] Antônio Carlos M. de Paula, Carlos Frederico J. Muakad, Sávio Freire, Rodrigo Spinola, and Manoel Mendonça. 2024. Burnout in Software Projects: An Analysis of Stack Exchange Discussions. In *Proceedings of the XXIII Brazilian Symposium on Software Quality (SBQS '24)*. ACM, New York, NY, USA, 321–330.
- [19] Daniel Russo, Paul HP Hanel, Seraphina Altnickel, and Niels Van Berkel. 2021. The daily life of software engineers during the covid-19 pandemic. In *IEEE/ACM 43rd ICSE-SEIP*. 364–373.
- [20] Daniel Russo, Paul HP Hanel, Seraphina Altnickel, and Niels van Berkel. 2021. Predictors of well-being and productivity among software professionals during the COVID-19 pandemic—a longitudinal study. *Empirical Software Engineering* 26, 4 (2021), 62.
- [21] Sabine Sonnentag, Felix C Brodbeck, Torsten Heinbokel, and Wolfgang Stolte. 1994. Stressor-burnout relationship in software development teams. *Journal of occupational and organizational psychology* 67, 4 (1994), 327–341.
- [22] Tien Rahayu Tulili, Andrea Capiluppi, and Ayushi Rastogi. 2023. Burnout in software engineering: A systematic mapping study. *Information and Software Technology* 155 (2023), 107116.
- [23] Hee-Dong Yang, Hye-Ryun Kang, and Robert M Mason. 2008. An exploratory study on meta skills in software development teams: antecedent cooperation skills and personality for shared mental models. *European Journal of Information Systems* 17, 1 (2008), 47–61.
- [24] Murat Yilmaz, Rory V O'Connor, Ricardo Colomo-Palacios, and Paul Clarke. 2017. An examination of personality traits and how they impact on software development teams. *Information and Software Technology* 86 (2017), 101–122.
- [25] Robert K Yin. 2018. *Case study research and applications*. Vol. 6. Sage Thousand Oaks, CA.