COLLECTIVE INTELLIGENCE IN PRACTICE: A CASE STUDY OF SAF-E IMPLEMENTATION IN LARGE-SCALE AUTOMOTIVE SOFTWARE DEVELOPMENT

KOLEKTIVNA INTELIGENCIJA U PRAKSI: STUDIJA SLUČAJA IMPLEMENTACIJE SAF-E U RAZVOJU SOFTVERA ZA AUTOMOBILSKU INDUSTRIJU VELIKIH RAZMERA

Maria Alessandra Montenegro[1] Sebastian Boenisch[2]

[1]Project Management College, Serbia, [2]CARIAD SE, Germany

Abstract: This paper presents a case study of the application of the Scaled Agile Framework (SAF-e) in a software development project within the German automotive industry. Conducted between 2021 and 2024, the project involved over 100 team members and multiple organizations across the automotive supply chain. The project focused on integrating artificial intelligence (AI) into in-vehicle systems, requiring high levels of coordination, adaptability, and compliance with safety standards. SAF-e was selected to enable structured collaboration, cross-team synchronization, and continuous value delivery across a distributed environment. The study adopts a qualitative approach based on the author's role as a Scrum Master, supported by internal documentation and informal interviews. Key aspects of SAF-e implementation—team structure, agile roles, backlog management, and synchronization events—are analyzed. Particular attention is given to how the framework fostered decentralized decision-making and distributed knowledge sharing, exemplifying principles of collective intelligence within large-scale project environments. The findings contribute to the understanding of agile at scale, especially in innovation-driven and safety-critical sectors like automotive. The paper concludes with practical implications and recommendations for future research on scaling agile practices in complex, regulated, and technologically evolving domains.

Keywords: Scaled Agile Framework (SAF-e), Collective Intelligence, Automotive Software Development, Agile Project Management, Large-Scale Agile Implementation

Corresponding author: Alessandra Montenegro, alessandramontenegro9@gmail.com

https://doi.org/10.56889/xxcb1864
PAGE 66-74

1. INTRODUCTION

Agile methods have become essential in software development, especially in complex sectors like the automotive industry, where software defines vehicle functionality and user experience (Pretschner et al., 2020). Traditional project management struggles to meet the fast pace and uncertainty in such environments. In contrast, agile offers iterative delivery, collaboration, and responsiveness to change (Beck et al., 2001). However, scaling agile across large, multi-team systems introduces new challenges (Dikert et al., 2016). Frameworks like the Scaled Agile Framework (SAF-e) address these by aligning teams, synchronizing delivery, and incorporating lean thinking (Leffingwell, 2018). Particularly in the automotive sector, where alignment across hardware, software, and compliance is essential, SAF-e proves highly applicable (Hohl et al., 2018).

As organizations transition to agile at scale, the concept of collective intelligence—emerging from distributed knowledge sharing, decentralized decision-making, and coordinated action—becomes increasingly relevant. In large-scale agile programs, such as those enabled by SAF-e, collective intelligence manifests through structured collaboration across cross-functional teams, shared cadences, and frequent synchronization points. These mechanisms help harness the diverse expertise and perspectives needed to navigate complexity and deliver value continuously.

This paper presents a case study of an AI-based software development project (2021–2024) in the German automotive industry, showcasing how SAF-e was applied to coordinate over 100 team members from multiple organizations. It explores project organization, roles, artifacts, and events, aiming to offer practical insight into scaled agile implementation and its relationship to collective intelligence in complex systems development.

2. THEORETICAL BACKGROUND

Agile principles, introduced by the Agile Manifesto (Beck et al., 2001), prioritize collaboration, flexibility, and delivery of working software. Frameworks like Scrum, Kanban, and XP have improved team productivity, time-to-market, and stakeholder engagement (Dingsøyr et al., 2012; Highsmith, 2002; Abrahamsson et al., 2017). In the automotive context, software-driven innovation demands such adaptability, especially for infotainment, ADAS, and electric vehicle systems (Broy, 2006; Pretschner et al., 2007). While effective for small teams, agile's expansion to enterprise-level projects introduces coordination issues, unclear roles, and challenges aligning with business strategies (Moe et al., 2012; Hoda et al., 2011). Automotive-specific constraints, like functional safety (ISO 26262), traceability, and hardware dependencies, further complicate traditional agile use (Peters et al., 2020).

To support agile at scale, frameworks such as LeSS, Nexus, DAD, and SAF-e have emerged. SAFe has become widely implemented due to its balance between structure and flexibility (Knaster & Leffingwell, 2017; Uludag et al., 2021). Organizations use SAF-e to ensure synchronized delivery and compliance (Hohl et al., 2018; Uludag et al., 2021). SAF-e integrates lean and systems thinking into a layered structure: Team, Program, Large Solution, and Portfolio levels (Leffingwell, 2018; Scaled Agile Inc., 2021). The core unit, the Agile Release Train (ART), brings together 5–12 teams around shared goals. ARTs deliver value through

Program Increments (PIs), planned and reviewed via synchronized events (Hikichi et al., 2017). This makes SAF-e particularly suited to safety-driven domains like automotive, requiring alignment across hardware, software, and vendors.

In this context, the concept of collective intelligence becomes increasingly relevant. Collective intelligence refers to the enhanced capacity that emerges from collaboration, information sharing, and coordinated decision-making among diverse actors (Malone & Bernstein, 2015). In large-scale agile environments, such as those organized around ARTs, collective intelligence is operationalized through practices like cross-team planning, decentralized ownership of work, and continuous integration of feedback across organizational boundaries. SAF-e structures these interactions to maximize shared knowledge and foster adaptive problem-solving, thus creating a systemic foundation for collective intelligence in complex project ecosystems.

3. CASE STUDY

3.1. PROJECT OVERVIEW AND STRUCTURE

The selected case involved a long-term software project in the German automotive industry that aimed to integrate artificial intelligence into in-vehicle systems. Conducted between 2021 and 2024, the project spanned several international locations, including Germany, the Czech Republic, and Serbia. The scope of the project, involving more than 100 participants and several partnering companies, necessitated a highly structured and collaborative management approach. To meet the project challenges, the organization adopted SAF-e as the primary framework for structuring agile delivery.

The project was structured around a single Agile Release Train (ART), internally referred to as the **Component Team**, which functioned as the central coordination unit. The team was responsible for delivering integrated and potentially shippable increments of value in a synchronized and time-boxed manner. The team had its own **Component** Team Product Owner (CT PO) and a dedicated Scrum Master, who together managed the backlog, facilitated team coordination, and ensured alignment with the product roadmap.

Within this ART, there were nine Scrum teams, locally referred to as **Unit Teams**. Each Unit Team operated semi-independently and was guided by a Unit Team Product Owner (UT PO) responsible for maintaining the team's sprint backlog and ensuring alignment with Component Team objectives. The Unit Teams were cross-functional and distributed across multiple locations, collaborating closely through digital tools such as JIRA, Confluence, and MS Teams. The forthcoming figure illustrates the hierarchical organization of the Component Team and its associated Unit Teams.

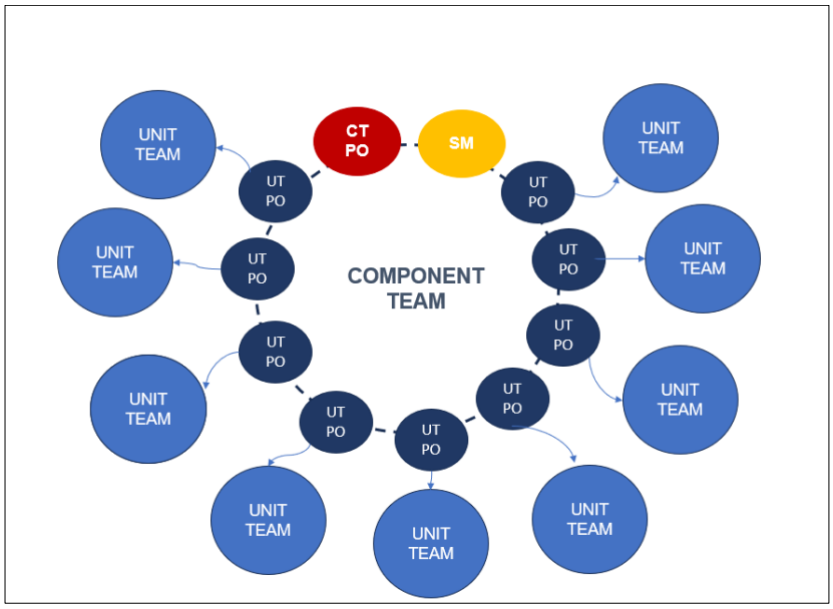

Figure 1. Project organization

This layered team structure allowed the organization to scale agile practices effectively while maintaining visibility and coordination across all levels of delivery.

3.2. METHODOLOGY

A qualitative single-case study approach was adopted. The author, serving as Scrum Master, participated directly in team events, planning sessions, and day-to-day operations, providing a detailed insider perspective. Anonymization of the client and system details ensured confidentiality.

Multiple qualitative methods were used: participant observation during agile ceremonies, analysis of internal documentation (e.g., JIRA boards, Confluence pages), and reflective journaling. Informal discussions with Product Owners, Scrum Masters, developers, and Release Train Engineers enriched the dataset. Thematic analysis was guided by SAF-e principles, with focus on coordination, backlog management, and role clarity.

3.3. PROJECT ROLES

The success of the SAF-e implementation heavily depended on clear role distribution and consistent role execution across all levels of the organization. This structure ensured that responsibilities were transparent and that collaboration was fostered both within and across teams.

The **Component Team Product Owner (CT PO)** was responsible for defining and maintaining the high-level backlog that encapsulated system-level capabilities and features. This role ensured that the backlog reflected business priorities and that items were continuously aligned with the long-term roadmap. The CT PO participated in key planning sessions, coordinated dependencies across Unit Teams, and integrated stakeholder feedback. They also played a pivotal role in adjusting priorities using methods like WSJF to ensure maximum value delivery.

The **Unit Team Product Owners (UT POs)** worked at the team level, refining the high-level features into smaller, actionable user stories. These POs ensured the team understood the scope of the work and facilitated ongoing backlog refinement. By maintaining a close relationship with the CT PO and the development teams, they served as a bridge between business objectives and technical execution. UT POs also ensured that backlog items met the Definition of Ready (DoR) before they entered the sprint cycle.

The **Scrum Master** was instrumental in ensuring smooth agile delivery. Beyond facilitating Scrum ceremonies, they acted as servant leader, promoting agile principles and removing organizational impediments. Their facilitation of both team-level and ART-level interactions was essential in maintaining rhythm and continuous improvement. Scrum Master also supported distributed team coordination and helped uphold team focus, ensuring that each sprint goal remained realistic and aligned.

The **Development Team members**, organized into cross-functional Unit Teams, took full ownership of delivering working software each sprint. They participated in sprint planning, daily stand-ups, reviews and retrospectives. Their role extended beyond implementation—they contributed to identifying technical risks, proposing architecture improvements, and continuously improving team performance. Autonomy, trust, and shared accountability characterized their contribution.

Together, this role configuration enabled a balance between decentralized execution and centralized coordination, which is critical in large-scale, distributed SAF-e environments.

3.4. PROJECT ARTEFACTS

A well-defined artefact hierarchy supported visibility, traceability, and alignment from strategic objectives down to sprint-level tasks. Artefacts were managed collaboratively and updated regularly to reflect evolving requirements and capacity constraints.

The **Component Team Backlog** served as the strategic planning layer. It included epics, capabilities, and features aligned with the long-term product roadmap. Items were evaluated based on business value, technical complexity, and stakeholder input.

The **Component Sprint Backlog** represented the tactical plan for each sprint. During sprint planning, features selected from the Component Backlog were decomposed into smaller user stories and tasks. These items were distributed across Unit Teams, considering interdependencies and team capacity.

Each **Unit Team Sprint Backlog** contained user stories and tasks assigned to a specific team. These backlogs were managed in JIRA and visualized through digital boards. The UT PO ensured that the backlog was always refined and prioritized, and team members took ownership of updating the status and progress throughout the sprint. Backlog items had clearly defined acceptance criteria and were continuously refined through team discussions.

At the end of each sprint, completed items across all layers were integrated into a **Product Increment**. This increment had to meet the Definition of Done (DoD), which included code completion, unit testing, peer review, and integration testing. It was validated during sprint reviews and served as the main deliverable for stakeholder inspection and roadmap alignment.

3.5. PROJECT EVENTS

Regular and well-structured events were central to maintaining SAF-e cadence, team alignment, and organizational learning. These ceremonies served both coordination and inspection-adaptation purposes, enabling teams to respond to change while maintaining strategic direction.

Component Team Sprint Planning brought together CT POs, UT POs and Scrum Masters to define the scope of the upcoming sprint. The session began with a review of the product roadmap and current status. Teams shared capacity estimates and raised potential blockers. Dependencies were identified, mitigation strategies agreed upon, and sprint goals finalized.

Unit Team Sprint Planning followed component-level planning and focused on operationalizing the sprint goals. Teams broke down prioritized user stories into concrete tasks, estimated the effort using planning poker, and committed to a realistic scope based on team velocity. This session enabled a shared understanding of deliverables and aligned expectations.

Component Daily Meetings were held to track high-level progress and synchronize activities across teams. Each meeting lasted 15 minutes and focused on addressing cross-team impediments, monitoring progress against the sprint goal, and raising escalations. The Scrum Master typically facilitated, and UT POs reported to CT PO on delivery status and any alignment needs.

Unit Team Daily Stand-ups were more granular and focused on task-level execution. Team members shared to UT PO what they had done, what they planned to do, and any impediments. These stand-ups reinforced individual accountability and encouraged peer support. They were also a mechanism for early detection of misalignments or delays.

Sprint Reviews took place at the end of each sprint. Unit Teams showcased completed user stories, and the CT PO evaluated overall progress toward roadmap objectives. Feedback from stakeholders, including external partners and business representatives, was collected and documented for future planning cycles. This ceremony confirmed that the increment met DoD and provided a learning opportunity.

Retrospectives were held separately at the Unit Team and Component Team levels. Unit Retrospectives focused on team dynamics, process efficiency, and collaboration challenges. Component Retrospectives addressed systemic issues such as tooling, cross-team dependencies,

or ART-level coordination gaps. Action items from retrospectives were tracked and reviewed regularly to ensure continuous improvement.

This robust event structure provided a stable rhythm for the project, enabling short feedback cycles, fast adaptation, and enhanced collaboration in a high-complexity, multi-team setting.

4. DISCUSSION

As presented in the introduction, the increasing complexity of software development—especially when it involves artificial intelligence and multiple stakeholders—necessitates new ways of organizing collaboration and knowledge integration. SAF-e not only provides a governance framework for scaling agile practices but also acts as an enabler of collective intelligence through its structured roles, ceremonies, and artefacts.

The case demonstrated that clearly defined roles, such as Component Team Product Owner (CT PO), Unit Team Product Owner (UT PO), Scrum Master, and Development Team, supported distributed leadership and decentralized decision-making. This division of responsibilities empowered teams to act autonomously while remaining aligned with common goals, a key characteristic of collective intelligence (Malone & Bernstein, 2015).

Artefacts such as layered backlogs (Component Team Backlog, Sprint Backlogs) and the Product Increment served as shared knowledge repositories, providing transparency and traceability across teams. These artefacts allowed distributed teams to operate from a common understanding of priorities and dependencies, enabling real-time coordination and alignment without constant centralized control.

SAF-e ceremonies—especially cross-team sprint planning, daily stand-ups, sprint reviews, and retrospectives—were essential in activating the collective cognitive capacity of the system. They provided time-boxed opportunities for synchronization, problem-solving, and feedback integration. Through these recurring events, collective awareness and adaptability were cultivated, allowing the system to self-correct and improve iteratively.

In essence, the structured interactions prescribed by SAF-e transformed a potentially fragmented multi-vendor setup into a cohesive project network capable of learning and adapting. This aligns with theoretical perspectives that view collective intelligence as an emergent property of well-orchestrated collaboration and knowledge flow (Malone & Bernstein, 2015).

From a practical standpoint, the case illustrates that cultivating collective intelligence is not a by-product but a deliberate outcome of how roles, artefacts, and events are configured and executed in large-scale agile systems. SAF-e provided the scaffolding for this orchestration, making it possible to harness distributed expertise in a complex, regulated, and high-stakes domain like automotive software development. These insights suggest that future adaptations of SAF-e or similar frameworks could explicitly embed collective intelligence principles to further enhance system-level learning, adaptability, and resilience in large projects.

5. CONCLUSION

Through an in-depth exploration of team structure, agile roles, artefacts, and coordination events, the study highlighted how SAF-e facilitates structured collaboration and continuous delivery in complex, distributed environments.

A key contribution of this study lies in its interpretation of SAF-e not only as a scaling framework but as a mechanism for enabling collective intelligence. The intentional design of roles, shared artefacts, and synchronized ceremonies allowed over 100 contributors from different organizations to integrate their expertise, coordinate decisions, and learn together. This emergent collective capability was critical for managing interdependencies, aligning stakeholder expectations, and responding to dynamic project conditions.

The findings demonstrate that cultivating collective intelligence is essential for agile success at scale—particularly in innovation-driven and safety-critical sectors such as automotive. SAF-e's layered structure supports the flow of knowledge, decentralized decision-making, and adaptive problem-solving, all of which are foundational elements of collective intelligence.

For practitioners, this case reinforces the value of explicitly designing agile environments to support collaborative cognition and shared ownership. Organizations implementing SAF-e should prioritize not only technical alignment but also cultural and procedural conditions that foster collective learning and distributed leadership.

Future research should explore how principles of collective intelligence can be further embedded in agile scaling frameworks and how their presence influences project performance, resilience, and innovation outcomes in various industrial contexts.

REFERENCES

Abrahamsson, P., Salo, O., Ronkainen, J., & Warsta, J. (2017). Agile software development methods: Review and analysis. VTT Publications.

Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., … Thomas, D. (2001). Manifesto for Agile Software Development. Retrieved from https://agilemanifesto.org/

Broy, M. (2006). Challenges in automotive software engineering. In Proceedings of the 28th International Conference on Software Engineering (pp. 33–42). ACM.

Dikert, K., Paasivaara, M., & Lassenius, C. (2016). Challenges and success factors for large-scale agile transformations: A systematic literature review. Journal of Systems and Software, 119, 87–108.

Dingsøyr, T., Nerur, S., Balijepally, V., & Moe, N. B. (2012). A decade of agile methodologies: Towards explaining agile software development. Journal of Systems and Software, 85(6), 1213–1221.

Highsmith, J. (2002). Agile Project Management: Creating Innovative Products. United States of America: Pearson Education Inc.

Hikichi, R., Sugimura, T., Tani, S., & Hirao, Y. (2017). A study on PI Planning and Inspect and Adapt workshop in SAFe. In 2017 IEEE International Conference on Software Architecture Workshops (ICSAW), 1–4.

Hoda, R., Noble, J., & Marshall, S. (2011). The impact of inadequate customer collaboration on self-organizing Agile teams. Information and Software Technology, 53(5), 521–534.

Hohl, P., Klünder, J., van Bennekum, A., Lockard, R., Gifford, J., Münch, J., & Schneider, K. (2018). Back to the future: Origins and directions of the "Agile Manifesto" – Views of the originators. Journal of Software: Evolution and Process, 30(1), e1953.
Knaster, R., & Leffingwell, D. (2017). SAFe® 4.0 Distilled: Applying the Scaled Agile Framework® for Lean Software and Systems Engineering. United States of America: Addison-Wesley Professional.
Leffingwell, D. (2018). SAFe® 4.5 Reference Guide: Scaled Agile Framework® for Lean Enterprises. United States of America: Addison-Wesley Professional.
Malone, T. W., & Bernstein, M. S. (2015). Collective intelligence. In T. W. Malone & M. S. Bernstein (Eds.), Handbook of Collective Intelligence. MIT Center for Collective Intelligence.
Moe, N. B., Smite, D., Ågerfalk, P. J., & Jørgensen, M. (2012). Understanding the dynamics in distributed agile teams: A case study of two agile teams. Information and Software Technology, 54(3), 1067–1081.
Peters, D., Stojanovic, N., & Dahanayake, A. (2020). Agile in the automotive domain: Challenges and benefits in an environment of regulated product development. Journal of Systems and Software, 163, 110518.
Pretschner, A., Broy, M., Krüger, I. H., & Stauner, T. (2007). Software engineering for automotive systems: A roadmap. In 2007 Future of Software Engineering (pp. 55–71). IEEE Computer Society.
Pretschner, A., Berger, C., Behnke, D., & Krieg, C. (2020). Software-defined vehicles: Challenges and research directions. Proceedings of the IEEE, 108(4), 593–605.
Scaled Agile Inc. (2021). SAFe® 5.0 for Lean Enterprises. Retrieved from https://www.scaledagileframework.com/
Uludag, Ö., Kleehaus, M., & Matthes, F. (2021). Analyzing the adoption and application of SAFe in industry. In 2021 IEEE International Conference on Software Architecture (ICSA), 61–70.
VersionOne. (2020). 14th Annual State of Agile Report. Retrieved from https://stateofagile.com/
Yin, R. K. (2018). Case Study Research and Applications: Design and Methods (6th ed.). United States of America: Sage Publications.