

Toward an Explanatory Theory of Adaptive Knowledge Orchestration in AI- Augmented Agile Software Development

Abdelaziz Omran Akhlaif

ABDELAZIZ.ABDELSALAM@uob.edu.ly

University of Benghazi

Perspective

Keywords: Adaptive Knowledge Orchestration, Agile Software Development, Distributed Cognition, Generative Artificial Intelligence, Human–AI Collaboration, Psychological Safety, Software Engineering Theory, Theory Building

Posted Date: October 15th, 2026

DOI: <https://doi.org/10.21203/rs.3.rs-11168141/v1>

License: © ⓘ This work is licensed under a Creative Commons Attribution 4.0 International License.

[Read Full License](#)

Additional Declarations: No competing interests reported.

Abstract

The rapid integration of generative artificial intelligence (GenAI) into Agile software development has fundamentally transformed how software teams acquire, negotiate, and apply knowledge throughout the development lifecycle. While existing Agile theories explain adaptation through iterative planning, customer collaboration, and self-organizing teams [1], they were developed under the implicit assumption that cognitive work is performed exclusively by human team members. This assumption is increasingly challenged as AI systems actively participate in software design, code generation, testing, documentation, and decision support. Existing theoretical perspectives—including Distributed Cognition Theory, Socio-Technical Systems Theory, Human–AI Teaming, and Psychological Safety—provide valuable but fragmented explanations of this emerging phenomenon. Individually, these theories explain knowledge distribution, socio-technical interaction, collaboration with intelligent systems, and team learning behaviours. However, none identifies the adaptive redistribution of cognitive responsibilities between human developers and AI cognitive infrastructure as a distinct causal mechanism requiring independent theoretical explanation. To address this theoretical gap, this paper develops Adaptive Knowledge Orchestration Theory (AKOT), a mid-range explanatory theory describing adaptive knowledge orchestration as the continuous and socially mediated process through which Agile teams dynamically redistribute cognitive responsibilities between human members and AI cognitive infrastructure in response to changing software development demands. The theory identifies three interdependent causal mechanisms—Cognitive Offloading, AI Output Negotiation, and Accountability Reconfiguration—that collectively explain how software quality emerges from adaptive cognitive coordination rather than from AI capability alone. The proposed theory is developed following established principles of theory building in information systems research [3, 4] and is grounded in existing literature on Agile software development, distributed cognition, socio-technical systems, human–AI collaboration, and psychological safety. Beyond explaining an emerging phenomenon in AI-augmented software engineering, AKOT provides a conceptual foundation for future measurement development, theory testing, and cumulative research on human–AI collaboration within Agile environments.

1. Introduction

Software development has historically evolved alongside advances in programming languages, development methodologies, and software engineering tools. During the past two decades, Agile software development has become the dominant paradigm for managing software projects in environments characterized by uncertainty, changing requirements, and continuous customer involvement [1]. Rather than emphasizing rigid planning and sequential development processes, Agile approaches promote iterative delivery, rapid feedback, close collaboration, and continuous adaptation. These principles have enabled organizations to respond more effectively to dynamic business environments and have consequently become the foundation of modern software engineering practice.

The emergence of Generative Artificial Intelligence (GenAI) represents one of the most significant technological disruptions experienced by software engineering since the widespread adoption of Agile

development itself. Unlike previous software development tools that primarily automated routine programming activities, contemporary AI systems actively participate in multiple cognitive activities throughout the software development lifecycle. Developers increasingly rely on AI systems to generate source code, explain existing software, produce technical documentation, identify defects, recommend architectural alternatives, generate test cases, and support design decisions. Consequently, AI is no longer confined to automating isolated development tasks; instead, it increasingly contributes to knowledge-intensive activities traditionally performed exclusively by human developers.

This transformation raises a fundamental theoretical question. Agile methodologies were originally developed under assumptions that implicitly position software teams as exclusively human cognitive systems. Knowledge sharing, collective decision-making, sprint planning, retrospective learning, and adaptive problem solving were all conceptualized as outcomes emerging from interactions among human team members [1]. As AI systems increasingly participate in these activities, the cognitive architecture of Agile teams changes in ways that existing theoretical explanations were never designed to explain.

The growing influence of AI within software engineering and Agile environments raises questions that recent work has begun to address from several angles, including team learning behavior, collaboration with intelligent systems, and the distribution of cognitive work. Similarly, recent investigations have demonstrated that psychological safety remains a critical determinant of software quality by encouraging knowledge sharing, admitting mistakes, and collaborative learning within Agile teams [5]. Other studies have investigated Human–AI Teaming from organizational and information systems perspectives, emphasizing collaborative interactions between human actors and intelligent systems [6]. Meanwhile, Distributed Cognition Theory [7] has provided an influential explanation of how cognitive activities are distributed across individuals and external artefacts, whereas Socio-Technical Systems Theory explains how organizational outcomes emerge through interactions between social and technological subsystems.

Collectively, these theoretical perspectives have significantly advanced understanding of software development in technology-rich environments. Nevertheless, they explain different aspects of the phenomenon rather than the phenomenon itself. Distributed Cognition Theory explains how cognition is distributed but does not explain how Agile teams continuously reorganize cognitive responsibilities as AI capabilities evolve. Socio-Technical Systems Theory conceptualizes interactions between people and technology but does not specify the adaptive cognitive mechanisms that emerge when AI becomes embedded within Agile software development processes. Human–AI Teaming literature explains collaborative work between humans and intelligent systems but does not explicitly address iterative Agile environments where cognitive responsibilities are repeatedly redistributed across development cycles. Likewise, recent work on psychological safety demonstrates how team learning behaviours contribute to software quality but does not explain how these behaviours are transformed when AI-generated knowledge becomes an integral component of collective decision-making.

Consequently, the theoretical challenge is not merely that existing theories are incomplete individually; rather, they lack an explanatory mechanism capable of describing how Agile teams dynamically reorganize cognitive responsibilities between human developers and AI cognitive infrastructure while maintaining adaptive performance and software quality. Simply combining existing theories does not resolve this limitation because theoretical integration alone does not generate new explanatory mechanisms. Instead, an explanation is required for the adaptive process through which cognitive work itself is continuously redistributed within AI-augmented Agile teams.

This paper argues that adaptation in contemporary Agile environments should be understood primarily as a cognitive phenomenon rather than solely as a managerial or process-oriented phenomenon. Specifically, Agile teams increasingly adapt by dynamically reorganizing cognitive responsibilities between human developers and AI cognitive infrastructure according to changing development demands. Software quality therefore emerges not merely from improved AI capability or improved human expertise independently, but from the effectiveness with which these cognitive responsibilities are orchestrated across the entire development team.

To explain this phenomenon, this paper develops Adaptive Knowledge Orchestration Theory (AKOT), a mid-range explanatory theory that conceptualizes adaptive knowledge orchestration as the continuous and socially mediated process through which Agile teams redistribute cognitive responsibilities between human members and AI cognitive infrastructure while preserving collective adaptive capability. Building upon established theory-building principles [3, 4], the proposed theory identifies the core constructs, underlying assumptions, causal mechanisms, boundary conditions, and theoretical propositions necessary to explain this emerging phenomenon and to support future empirical investigation.

The contributions of this paper are fourfold:

1. It identifies an emerging theoretical gap created by the integration of Generative AI into Agile software development and demonstrates why existing theories cannot fully explain this phenomenon independently or collectively.
2. It formally defines Adaptive Knowledge Orchestration as a new theoretical construct describing adaptive cognitive reconfiguration within AI-augmented Agile teams.
3. It proposes a coherent explanatory theory specifying the causal mechanisms through which adaptive knowledge orchestration influences software quality.
4. It develops theoretically grounded propositions intended to guide future empirical validation, measurement development, and cumulative theory building in AI-augmented software engineering.

The remainder of this paper is organized as follows: Section 2 reviews the theoretical foundations underpinning the proposed theory and critically examines their explanatory limitations. Section 3 develops Adaptive Knowledge Orchestration Theory by introducing its formal definition, assumptions, constructs, causal mechanisms, and theoretical propositions. Section 4 discusses the theoretical implications of the proposed theory and its relationship with existing software engineering literature.

Section 5 details practical implications. Finally, Sections 6 and 7 present the limitations, future research directions, and conclusions of the paper.

2. Theoretical Foundations and Meta-Theoretical Positioning

2.1 Agile Software Development as a Human-Centered Adaptive Paradigm

Since the publication of the Agile Manifesto, Agile software development has become the dominant paradigm for managing software projects under conditions of uncertainty and continuous change. Rather than prescribing fixed development plans, Agile emphasizes iterative delivery, customer collaboration, continuous feedback, and adaptive planning. Conboy [1] argued that agility should not be understood merely as a collection of development practices but as the capability of an organization or team to continuously sense and respond to change. Consequently, adaptation constitutes the central theoretical mechanism underlying Agile software development.

Traditional Agile literature conceptualizes adaptation as a fundamentally human activity. Sprint planning, backlog refinement, software design discussions, retrospective meetings, and collective decision-making all assume that knowledge is generated, interpreted, negotiated, and validated through interactions among human team members. Learning emerges through communication, shared experience, and continuous reflection, while software quality results from iterative collaboration and rapid feedback loops [1].

The emergence of Generative Artificial Intelligence fundamentally challenges this assumption. Contemporary AI systems increasingly perform activities that were historically considered exclusively human cognitive functions, including software design assistance, source-code generation, architectural reasoning, requirements interpretation, testing support, documentation generation, and technical explanation. Consequently, cognitive work within Agile teams is no longer exclusively distributed among human developers but increasingly shared with AI systems integrated into everyday development practices.

2.2 Distributed Cognition Theory and the Distribution of Cognitive Work

Distributed Cognition Theory (DCT), originally proposed by Hutchins [7], represents one of the most influential theoretical perspectives for understanding cognition in complex socio-technical environments. Instead of conceptualizing cognition as an internal property of individual minds, DCT argues that cognitive processes emerge through interactions among individuals, artefacts, representations, and environmental structures. Knowledge is therefore distributed across an entire cognitive system rather than residing solely within individual actors.

Within software engineering, this perspective provides an important explanation for collaborative development activities. Documentation, design diagrams, source-code repositories, issue-tracking systems, architectural models, and communication platforms all function as external cognitive artefacts supporting collective reasoning. Team cognition therefore extends beyond individual developers to encompass the broader development environment.

However, despite its explanatory power, Distributed Cognition Theory does not explicitly explain how cognitive responsibilities are continuously reconfigured as AI capabilities evolve. The theory explains how cognition is distributed across a system but provides limited explanation of the adaptive processes through which development teams deliberately reorganize this distribution in response to changing project demands.

2.3 Socio-Technical Systems Theory and Human–Technology Interaction

Socio-Technical Systems (STS) Theory conceptualizes organizational performance as the joint outcome of interactions between social and technological subsystems. Rather than optimizing technology independently of people, STS argues that organizational effectiveness depends upon the mutual adaptation of both components [12].

Nevertheless, STS primarily explains interactions between social and technological subsystems at an organizational level. It does not specify the internal cognitive mechanisms through which Agile teams continuously reorganize cognitive responsibilities during iterative software development.

2.4 Human–AI Teaming: From Collaboration to Cognitive Partnership

Human–AI Teaming (HAT) conceptualizes AI systems as collaborative partners that contribute to shared problem solving, decision making, and knowledge creation [6]. Existing research has identified several determinants of successful collaboration, including transparency, trust, explainability, complementary expertise, and adaptive coordination between human and artificial agents [6].

Despite these contributions, existing Human–AI Teaming literature remains largely domain-independent. Most theoretical models describe collaboration across generic organizational contexts without explicitly considering the iterative, knowledge-intensive, and continuously adaptive characteristics of Agile software development.

2.5 Psychological Safety and Collective Learning in Agile Software Development

Psychological Safety refers to a shared team belief that interpersonal risk-taking can occur without fear of embarrassment, punishment, or negative interpersonal consequences [8]. Alami et al. [5] showed that

Psychological Safety significantly influences software quality through behaviors such as knowledge sharing, speaking up, and admitting mistakes.

Current Psychological Safety literature successfully explains why open communication improves collaborative outcomes but does not explicitly explain how AI-generated knowledge alters these communication processes or how teams reorganize cognitive responsibilities while maintaining psychologically safe collaboration.

2.6 Meta-Theoretical Positioning

Table 1 summarizes the explanatory scope and limitations of the principal theoretical perspectives discussed above and identifies the specific contribution proposed by Adaptive Knowledge Orchestration Theory.

Table 1
Explanatory Scope and Limitations of Existing Theoretical Perspectives

Theory	Explains	Does Not Explain	AKOT Contribution
Agile Theory [1]	Organizational adaptation through iterative development	Adaptive redistribution of cognitive responsibilities between humans and AI	Explains cognitive adaptation as the primary mechanism of Agile evolution
Distributed Cognition Theory [7]	Distribution of cognition across people and artefacts	Dynamic reconfiguration of cognitive work during Agile iterations	Introduces Adaptive Knowledge Orchestration as a continuous redistribution process
Socio-Technical Systems Theory [12]	Interaction between social and technological subsystems	Internal cognitive mechanisms operating within Agile teams	Specifies the cognitive mechanisms through which socio-technical adaptation occurs
Human–AI Teaming [6]	Collaboration between humans and intelligent systems	Iterative cognitive adaptation in Agile software engineering	Positions AI as cognitive infrastructure embedded in continuous Agile adaptation
Psychological Safety [5, 8]	Team learning, knowledge sharing, and software quality	AI-mediated transformation of collaborative cognition	Explains how Psychological Safety moderates Adaptive Knowledge Orchestration

3. Adaptive Knowledge Orchestration Theory (AKOT)

3.1 Formal Definition of Adaptive Knowledge Orchestration

Definition

Adaptive Knowledge Orchestration (AKO) is defined as the continuous, dynamic, and socially mediated process through which Agile software teams reconfigure the distribution of cognitive responsibilities between human developers and AI cognitive infrastructure in response to evolving software development demands, with the objective of sustaining collective problem-solving capability and software quality.

The construct possesses five defining attributes: Continuous, Dynamic, Socially Mediated, Adaptive, and Quality-Oriented.

3.2 Core Theoretical Constructs

These constructs draw on established theoretical foundations outside Agile software engineering. Infrastructure Theory [9] explains how a technology becomes embedded within routine work rather than remaining an external tool, which grounds the treatment of AI as cognitive infrastructure. Cognitive Load Theory [10] explains how the reasoning capacity available to a person change once part of a task is delegated elsewhere, which grounds the treatment of Human Reflective Capacity as a resource that is freed, not fixed. The explanatory capability of AKOT relies on a well-defined set of constructs summarized in Table 2.

Table 2
Core Constructs of Adaptive Knowledge Orchestration Theory (AKOT)

Construct	Formal Definition	Primary Theoretical Foundation	Role within AKOT
AI Cognitive Infrastructure Maturity	Degree to which AI becomes embedded within collective cognitive work rather than remaining an external development tool	Infrastructure Theory [9]; Distributed Cognition [7]	Antecedent
Adaptive Knowledge Orchestration	Continuous redistribution of cognitive responsibilities between humans and AI	Proposed construct	Central mechanism
Human Reflective Capacity	Cognitive resources available for critical reasoning after routine cognition has been delegated	Cognitive Load Theory [10]	Mediator
Human–AI Trust Calibration	Team's capability to appropriately regulate trust toward AI-generated outputs	Human–AI Teaming [6]	Mediator
Team Psychological Safety	Shared belief that interpersonal risk-taking is safe during collaborative work	Psychological Safety Theory [8]	Moderator
Software Quality	Overall quality of software products emerging from collective development activities	Software Engineering Quality Literature	Outcome

3.3 Mathematical Formulation of Scientific Notation

To maintain rigorous mathematical notation according to Springer style guidelines, physical variables, unknown quantities, and operational parameters within AKOT are represented in *italics*, vector spaces in **bold**, and operational functions in upright/roman text:

Let C_m denote the *AI Cognitive Infrastructure Maturity*. The effective cognitive output of the hybrid human–AI team, denoted as **Q** (*Software Quality* vector), is modeled as a function of *Adaptive Knowledge Orchestration* (O), *Human Reflective Capacity* (R), and *Human–AI Trust Calibration* (T), moderated by *Team Psychological Safety* (ψ):

$$\mathbf{Q} = f(O(C_m), R, T; \psi)$$

3.4 Causal Mechanisms and Integrated Explanatory Model

AKOT identifies three interacting cognitive mechanisms operating continuously throughout Agile development:

1. **Cognitive Offloading:** The adaptive transfer of routine or computationally intensive cognitive activities to AI cognitive infrastructure.
2. **AI Output Negotiation:** The collaborative process through which teams evaluate, interpret, modify, or reject AI-generated knowledge.
3. **Accountability Reconfiguration:** The continuous redistribution of responsibility for software decisions within hybrid human–AI cognitive systems.

3.5 Theoretical Propositions

- **P1:** *AI Cognitive Infrastructure Maturity (C_m) positively influences Software Quality (**Q**) through the sequential mediating effects of Cognitive Offloading, Human Reflective Capacity (R), and AI Output Negotiation.*
- **P2:** *The positive relationship between Adaptive Knowledge Orchestration (O) and Software Quality (**Q**) is strengthened when Team Psychological Safety (ψ) is high because psychological safety facilitates effective Accountability Reconfiguration.*
- **P3:** *Human–AI Trust Calibration (T) and AI Output Negotiation reinforce one another through a reciprocal relationship in which improvements in either process enhance the effectiveness of subsequent Adaptive Knowledge Orchestration (O).*
- **P4:** *Adaptive Knowledge Orchestration (O) mediates the relationship between AI Cognitive Infrastructure Maturity (C_m) and Software Quality (**Q**) such that improvements in AI capability alone are insufficient to improve software quality without effective cognitive orchestration.*
- **P5:** *The relationship between Cognitive Offloading and Software Quality (**Q**) follows an inverted U-shaped pattern such that moderate levels of cognitive redistribution maximize software quality, whereas excessive cognitive delegation reduces Human Reflective Capacity (R) and ultimately diminishes software quality.*

Table 3
Theoretical Foundations and Expected Empirical Analysis of Proposed Relationships

Proposition	Principal Supporting Theory	Nature of Relationship	Expected Empirical Analysis
P1	Distributed Cognition [7] + Cognitive Load Theory [10]	Sequential Mediation	Structural Equation Modeling (SEM)
P2	Psychological Safety Theory [5, 8]	Moderation	Interaction Effects / Multi-group SEM
P3	Human–AI Teaming [6]	Reciprocal	Cross-lagged / Longitudinal Models
P4	Infrastructure Theory [9] + Distributed Cognition [7]	Mediation	Structural Equation Modeling (SEM)
P5	Cognitive Load Theory [10] + Adaptive Automation [11]	Non-linear (Inverted U)	Polynomial Regression / Quadratic SEM

4. Discussion

4.1 Reframing Agile Adaptation in the Era of Generative AI

AKOT proposes that adaptation in AI-augmented Agile environments can no longer be understood solely as the iterative reorganization of development tasks, sprint planning activities, or collaborative workflows. Instead, adaptation increasingly involves the continuous redistribution of cognitive responsibilities between human developers and AI cognitive infrastructure.

4.2 Positioning AKOT Within Existing Theoretical Landscape

Where Distributed Cognition Theory [7] explains the architecture of cognition, AKOT explains the adaptive reconfiguration of that architecture under conditions of continuous AI participation. Furthermore, AKOT extends Human–AI Teaming [6] by shifting the unit of analysis to the entire team cognitive system rather than dyadic interactions between a single human and a single AI agent. Finally, it positions Psychological Safety [5, 8] as a crucial enabling cognitive condition that allows teams to negotiate AI outputs and reconfigure accountability without fear of interpersonal repercussions.

4.3 Interaction Among the Three Causal Mechanisms

The three causal mechanisms are not independent of one another; instead, each mechanism shapes the conditions under which the other two operate. Cognitive Offloading determines how much AI-generated content enters the team’s workflow, which in turn defines the volume and difficulty of the evaluative work required during AI Output Negotiation. When offloading increases faster than the team’s negotiation capacity, AI-generated outputs tend to be accepted with less scrutiny, and Accountability Reconfiguration becomes harder to trace because fewer decisions are explicitly discussed and validated. This coupling

explains why Cognitive Offloading cannot be treated as uniformly beneficial. Consistent with Parasuraman et al.'s [11] account of adaptive automation, human performance degrades not only when automation is absent but also when automation is applied beyond the level that operators can meaningfully monitor and correct. Applied to AKOT, this logic supports Proposition P5: moderate Cognitive Offloading preserves enough Human Reflective Capacity for effective AI Output Negotiation, whereas excessive offloading narrows the reflective capacity needed to catch errors before they propagate into Accountability Reconfiguration and, ultimately, into Software Quality. Team Psychological Safety moderates this coupling by determining whether negotiation failures and accountability gaps are surfaced and corrected early or left unaddressed until they affect delivered software.

5. Practical Implications

The theoretical mechanisms proposed by AKOT translate into actionable practices for organizations seeking to integrate generative AI into Agile software development. Table 4 summarizes the practical recommendations derived from each mechanism, with corresponding responsible roles.

Table 4
Practical Implications of AKOT Mechanisms

AKOT Mechanism	Practical Recommendation	Responsible Role
Cognitive Offloading	Identify and document routine cognitive tasks suitable for AI delegation at the start of each Sprint. Establish clear criteria for which activities should remain human.	Scrum Master
AI Output Negotiation	Incorporate systematic review of AI-generated outputs as a fixed agenda item in Sprint Review meetings. Document negotiation outcomes.	Development Team
Accountability Reconfiguration	Document decisions influenced by AI in the Sprint Log, explicitly noting the level of AI involvement and human validation performed.	Tech Lead
Human Reflective Capacity	Allocate dedicated time in Sprint Planning to discuss how released cognitive resources will be directed toward higher-order reasoning.	Product Owner
Human–AI Trust Calibration	Conduct periodic surveys (every 3 Sprints) to assess team trust in AI outputs. Use results to adjust practices and provide targeted training.	Agile Coach

6. Limitations and Future Research

6.1 Theoretical Limitations

First, the theory remains conceptually grounded in existing literature rather than empirically validated. Second, the theory assumes that Agile environments represent the appropriate organizational context

for Adaptive Knowledge Orchestration. Third, it does not extensively address variations in AI capabilities across different generative models.

6.2 Empirical Research Agenda

Future research should focus on:

1. **Measurement Development:** Developing valid and reliable instruments for AKO, Human Reflective Capacity, and Trust Calibration.
2. **Cross-Sectional & Longitudinal Validation:** Testing P1–P5 using structural equation modeling and longitudinal research designs across multiple sprints.
3. **Qualitative Exploration:** Performing ethnographies and case studies to understand how developers experience cognitive redistribution in practice.

7. Conclusion

To address the theoretical gap created by GenAI integration into Agile teams, this paper developed Adaptive Knowledge Orchestration Theory (AKOT). By conceptualizing AI as cognitive infrastructure and identifying Cognitive Offloading, AI Output Negotiation, and Accountability Reconfiguration as fundamental causal mechanisms, AKOT explains how software quality emerges from adaptive cognitive coordination rather than AI capability alone. The theory offers a conceptual foundation to guide future empirical validation and cumulative research on human–AI collaboration within software engineering.

Declarations

Ethical Approval

Not Applicable.

Consent to Participate

Not Applicable.

Consent to Publish

Not Applicable.

Data Availability Statement

Not Applicable.

Funding

No funding was received for conducting this study.

Competing Interests

The author has no relevant financial or non-financial interests to disclose.

Acknowledgments

The author extends sincere appreciation to the Department of Information Technology at the University of Benghazi for institutional support during the preparation of this research.

Use of Large Language Models (LLMs) / Generative AI

Generative AI tools were utilized solely for AI-assisted copy editing (improving language, readability, and ensuring grammatical accuracy) in accordance with Springer's publishing policies. No autonomous content creation or generative editorial reasoning was performed by AI, and full human accountability for the final text rests with the author.

Author Contribution

A.O.A. conceived the theoretical framework, developed Adaptive Knowledge Orchestration Theory (AKOT), performed the literature synthesis, and wrote and reviewed the main manuscript text.

References

1. Conboy K. Agility from first principles: Reconstructing the concept of agility in information systems development. *Inf Syst Res.* 2009;20(3):329–54.
2. Yang N, Wang X, Zhang Z, Siemon D, Hyrynsalmi S et al. Core theories in agile software development. In: Peter, S. editors *Proc. 26th Int. Conf. Agile Software Development (XP 2025)*, pp. 3–18. Springer, Heidelberg (2025).
3. Gregor S. The nature of theory in information systems. *MIS Q.* 2006;30(3):611–42.
4. Whetten DA. What constitutes a theoretical contribution? *Acad. Manage Rev.* 1989;14(4):490–5.
5. Alami A, Zahedi M, Krancher O. The role of psychological safety in promoting software quality in agile teams. *Empir Softw Eng.* 2024;29:119.
6. Seeber I, Bittner E, Briggs RO, de Vreede T, de Vreede GJ, Elkins A, Maier R, Merz AB, Oeste-Reiß S, Randrup N, Schwabe G. Machines as teammates: A research agenda on AI in team collaboration. *Inf*

Manage. 2020;57(2):103174.

7. Hutchins E. Cognition in the Wild. Cambridge: MIT Press; 1995.
8. Edmondson A. Psychological safety and learning behavior in work teams. Adm Sci Q. 1999;44(2):350–83.
9. Star SL, Ruhleder K. Steps toward an ecology of infrastructure: Design and access for large information spaces. Inf Syst Res. 1996;7(1):111–34.
10. Sweller J. Cognitive load during problem solving: Effects on learning. Cogn Sci. 1988;12(2):257–85.
11. Parasuraman R, Sheridan TB, Wickens CD. A model for types and levels of human interaction with automation. IEEE Trans Syst Man Cybern Syst Hum. 2000;30(3):286–97.
12. Trist EL, Bamforth KW. Some social and psychological consequences of the longwall method of coal-getting. Hum Relat. 1951;4(1):3–38.

Figures

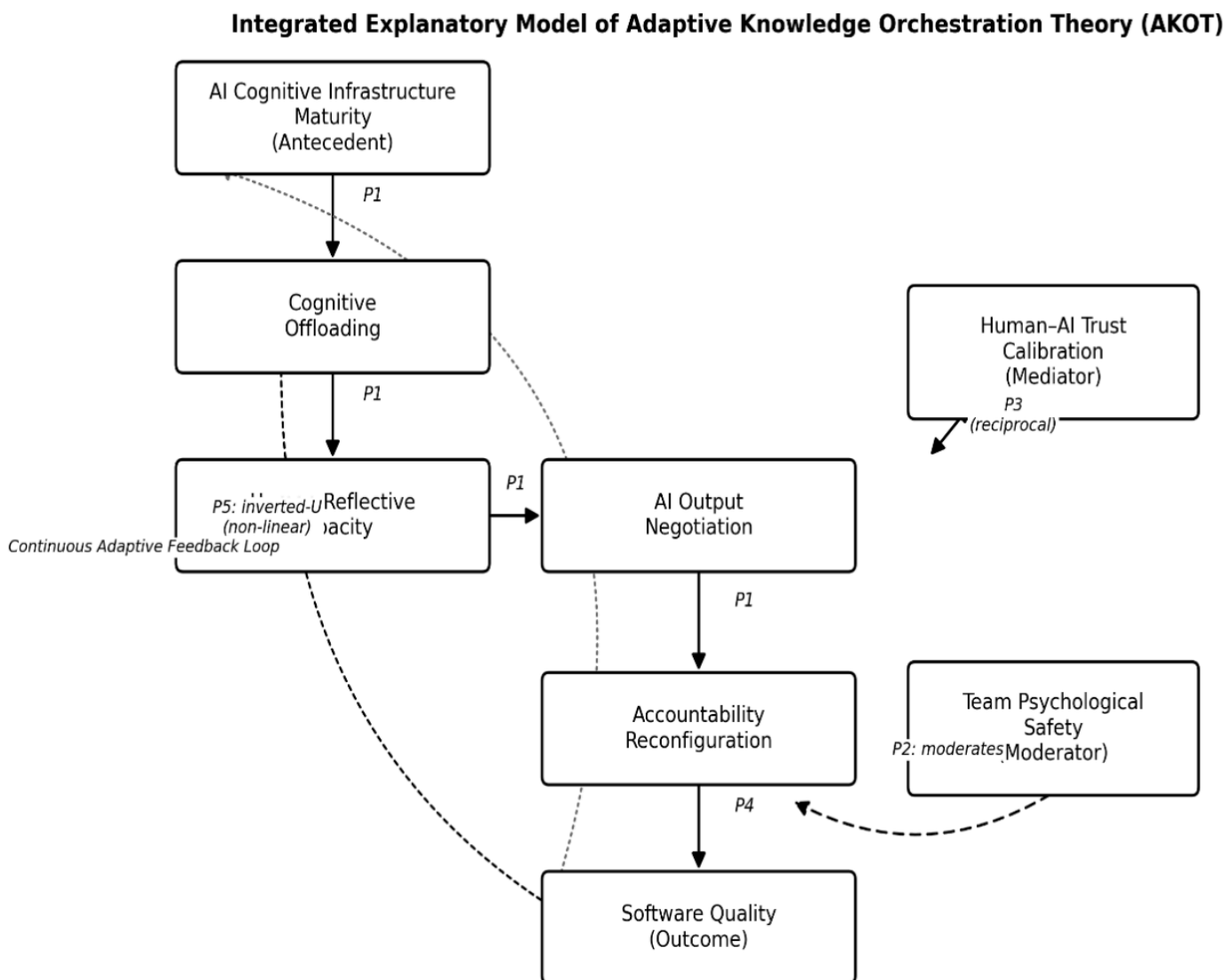


Figure 1

Integrated Explanatory Model of Adaptive Knowledge Orchestration Theory (AKOT) showing recursive relationships and moderating effects.