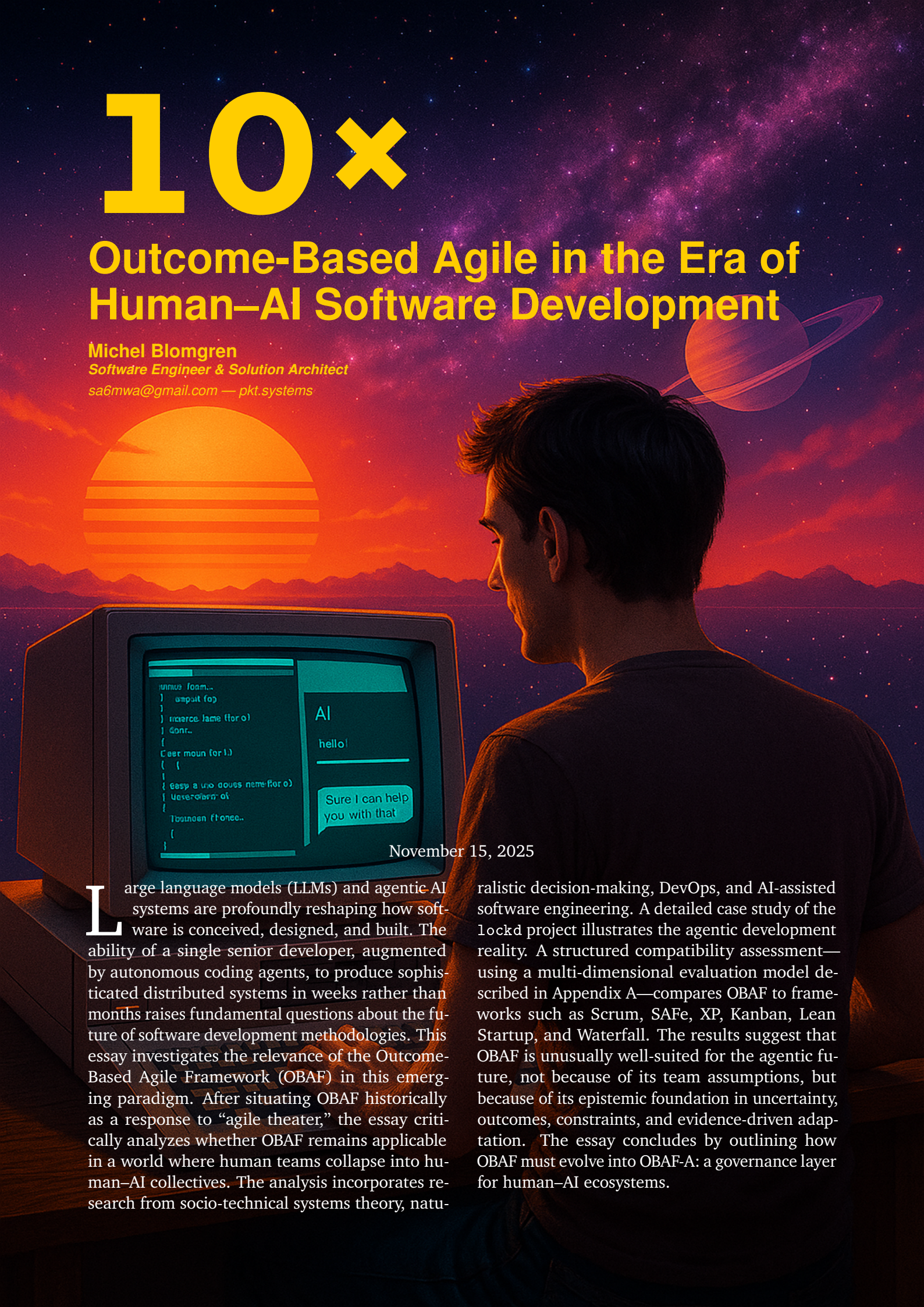
10×

Outcome-Based Agile in the Era of Human–AI Software Development

Michel Blomgren
Software Engineer & Solution Architect
sa6mwa@gmail.com — pkt.systems



November 15, 2025

Large language models (LLMs) and agentic AI systems are profoundly reshaping how software is conceived, designed, and built. The ability of a single senior developer, augmented by autonomous coding agents, to produce sophisticated distributed systems in weeks rather than months raises fundamental questions about the future of software development methodologies. This essay investigates the relevance of the Outcome-Based Agile Framework (OBAF) in this emerging paradigm. After situating OBAF historically as a response to "agile theater," the essay critically analyzes whether OBAF remains applicable in a world where human teams collapse into human–AI collectives. The analysis incorporates research from socio-technical systems theory, naturalistic decision-making, DevOps, and AI-assisted software engineering. A detailed case study of the `lockd` project illustrates the agentic development reality. A structured compatibility assessment—using a multi-dimensional evaluation model described in Appendix A—compares OBAF to frameworks such as Scrum, SAFe, XP, Kanban, Lean Startup, and Waterfall. The results suggest that OBAF is unusually well-suited for the agentic future, not because of its team assumptions, but because of its epistemic foundation in uncertainty, outcomes, constraints, and evidence-driven adaptation. The essay concludes by outlining how OBAF must evolve into OBAF-A: a governance layer for human–AI ecosystems.

INTRODUCTION

Software engineering is undergoing a transformation unprecedented in both speed and scope. The emergence of large language models (LLMs), autonomous coding assistants, and multi-agent AI systems is collapsing long-standing assumptions about team-based development, coordination costs, and the economics of iteration. AI-augmented workflows increasingly allow individual developers to accomplish what previously required entire teams.

The `lockd` project is emblematic of this shift (Blomgren, 2025a). Created over a period of approximately five weeks of spare-time work, it evolved into a coherent distributed system with a feature set comparable to those implemented by multi-person engineering teams: a distributed lock service with strong consistency semantics, a NoSQL document store, an indexing and query engine, a messaging system, a multi-backend storage abstraction, and a streaming response architecture. This scale of achievement by one developer—working in collaboration with an AI coding agent—illustrates the emerging paradigm of human–AI cognitive partnerships.

This essay investigates whether existing methodologies, particularly the Outcome-Based Agile Framework (OBAF) (Blomgren, 2025b), remain relevant in this new environment. OBAF was originally conceived as an antidote to "agile theater," re-centering Agile around outcomes, constraints, discovery, and evidence. But does it retain utility when discovery becomes automated, and when execution is handled by AI agents rather than human teams? We explore this question by integrating historical analysis, conceptual reasoning, and an empirical case study.

FROM AGILE TO AGILE THEATER

Agile emerged in the early 2000s as a response to heavyweight, plan-driven methodologies such as Waterfall and the V-model, which assumed stable requirements and high-cost iteration. Research across multiple decades has documented that these methods perform poorly under uncertainty, where the cost of prediction exceeds the cost of change (Highsmith, 2001; Beck, 1999; Cockburn, 2001). Agile's emphasis on iteration, user feedback, and adaptation proved more effective.

However, as Agile diffused into large organizations, its success triggered a paradox: its popularity produced bureaucratic mutations that reintroduced the same rigidity Agile was designed to eliminate. Frameworks like Scrum and SAFe became increasingly ceremony-driven, emphasizing ritual compliance (sprints, standups, PI planning, role boundaries) over genuine learning. This drift into "agile theater" is now widely documented in empirical studies (Kuusinen et al., 2017; Lewallen, 2020). The gap between Agile's epistemic intent and organizational practice became pronounced.

THE ORIGINS AND PHILOSOPHY OF OBAF

The Outcome-Based Agile Framework (OBAF) (Blomgren, 2025b) was created as an explicit correction to this drift. Rather than prescribing roles, ceremonies, or artifacts, OBAF emphasizes:

- **Outcomes over outputs**, reflecting behavioral or value changes rather than feature delivery;
- **Constraints over scope**, defining boundaries but not solutions;
- **Continuous discovery**, where learning and execution are intertwined;
- **Recognitional planning**, drawing on naturalistic decision-making research (Klein, 1998);
- **Evidence-based adaptation**, using signals, telemetry, and user behavior to steer action.

OBAF's intellectual foundations span multiple disciplines (Blomgren, 2025b): Boyd's OODA loop (Osinga, 2006), distributed cognition (Hutchins, 1995), socio-technical systems theory (Trist, 1981), DevOps and continuous delivery research (Forsgren, Humble, and Kim, 2018), and agile empirical studies.

A key insight is that OBAF is epistemic rather than procedural: it aims to align action with uncertainty, intention, and evidence. This becomes important when evaluating OBAF in the context of human–AI collaboration.

THE AGENTIC FUTURE OF SOFTWARE DEVELOPMENT

Recent empirical studies on AI-assisted software engineering show significant productivity gains when developers use LLMs for coding, testing, and refactoring (Chen, 2021; Bavishi, 2022). AI agents can generate architectural analyses, propose implementation plans, synthesize tests, and explore design alternatives. Multi-agent computational architectures extend this to autonomous workflows (Qian et al., 2023; Zhao, 2023).

These shifts dramatically reduce the cost of iteration, integration, and experimentation. They also minimize human coordination costs, a dominant bottleneck in software development (Brooks, 1995). When AI handles execution, and discovery becomes continuous and automated, the concept of a "team" changes fundamentally.

In this landscape, frameworks built around human rituals, roles, and coordination (e.g., Scrum, SAFe) face structural incompatibility. Yet frameworks built around epistemic principles (e.g., OBAF, Lean Startup) may prove more adaptable.

CASE STUDY: LOCKD AND THE HUMAN–AI COGNITIVE LOOP

The development of `lockd` is a compelling illustration of the agentic future (Blomgren, 2025a). Over approximately one month of spare-time development, a single senior developer collaborated with an AI coding assistant (via the Codex CLI) to build a system that typically requires a team of distributed systems engineers. The AI contributed architectural analyses, implementation plans, code synthesis, and refactoring strategies. The developer provided context, constraints, domain knowledge, and recognitional judgment.

The resulting system—a distributed lock mechanism, state/document store, query engine, message queue, and streaming architecture—reflects architectural coherence normally derived from cross-functional collaboration, design reviews, and iterative refactoring cycles. The developer–AI workflow documented in the repository's `AGENTS.md` file shows continuous methodological adaptation, rapid feedback loops, and minimal coordination overhead.

This pattern resembles the "centaur model" of human–AI teaming (Shneiderman, 2020), where humans make contextual judgments while AI agents amplify generative and analytic capacity.

EVALUATING FRAMEWORKS FOR THE AGENTIC ERA

To assess the compatibility of existing methodologies with the human–AI development paradigm, we constructed a structured multi-dimensional evaluation model (fully described in Appendix A). The model consists of six dimensions:

1. ritual independence,
2. coordination independence,
3. cheap-iteration assumption,
4. outcome orientation,
5. automated-discovery compatibility,
6. human–AI ecosystem compatibility.

Each dimension is composed of five binary indicators derived from empirical literature in software engineering, organizational science, sociotechnical systems, and human–AI collaboration. A framework's compatibility score is computed as the average across all dimensions.

COMPATIBILITY SCORES

Table 1 presents compatibility scores for selected frameworks, computed via the methodology in Appendix A.

Framework	Compatibility with Human–AI Ecosystem (%)
Outcome-Based Agile (OBAF)	**100%**
Lean Startup	93%
DevOps/SRE	80%
Kanban	77%
Extreme Programming (XP)	56%
Scrum	22%
LeSS	16%
SAFe	8%
Waterfall/V-model/PRINCE2	6%

Table 1: Compatibility estimates of selected frameworks with a human–AI development ecosystem. Scores are derived from structured expert judgment across research-backed dimensions; see Appendix A for methodology, scoring details, and justification.

These results indicate that frameworks grounded in epistemic principles (OBAF, Lean Startup) align most naturally with human–AI cognitive ecosystems, while coordination-heavy and ceremony-dependent frameworks (Scrum, SAFe, LeSS) are structurally incompatible.

DOES OBAF REMAIN RELEVANT?

The central question is whether OBAF retains relevance in a world where teams dissolve into human–AI ecosystems. Initial intuition might suggest that frameworks built for human teams become obsolete. However, the analysis here supports a different conclusion.

OBAF's principles are not tied to human team structure. They are tied to uncertainty, intent, constraints, and evidence—all of which remain essential in the agentic future. As AI accelerates discovery and execution, the role of the human shifts to:

- framing outcomes,
- defining constraints,
- overseeing automated exploration,
- interpreting ambiguous results,
- making ethical and strategic decisions.

This aligns directly with OBAF's purpose (Blomgren, 2025b). As such, OBAF becomes a governance layer for human–AI development rather than a team methodology. Its evolution into OBAF-A (described in the conclusion) formalizes this role.

CONCLUSION

The transformation underway in software development cannot be overstated. As AI systems reduce the cost of iteration and coordination to near zero, the methodological landscape must shift accordingly. Frameworks built around human coordination structures lose relevance; frameworks built around uncertainty, outcomes, and learning gain importance.

OBAF, despite its origins in human teams, remains uniquely well-suited for the agentic future due to its epistemic foundation. The `lockd` case study illustrates how human–AI teaming can dramatically amplify individual capability. (Blomgren, 2025a) The compatibility model presented here situates OBAF at the leading edge of methodologies adaptable to this future. Moving forward, OBAF-A will be needed to fully articulate governance, alignment, and constraint-setting in human–AI ecosystems.

ABOUT THE AUTHOR

Michel Blomgren is a software engineer and solutions architect specializing in distributed systems, large-scale infrastructure, and operational automation. With over two decades of experience, he has designed and implemented fault-tolerant microservice architectures, distributed workflow systems, and cloud-native integration pipelines across both commercial and defense contexts.

His work includes developing Kubernetes controllers and operators, architecting resilient networking and platform services, and guiding organizations in adopting container orchestration, Infrastructure as Code, and secure software-delivery practices. Colleagues consistently note his ability to clarify complex system behavior and support effective cross-disciplinary collaboration.

Blomgren is also the author of the Outcome-Based Agile Framework (Blomgren, 2025b), reflecting his broader interest in how feedback loops, constraints, and adaptive planning can improve decision-making in complex sociotechnical systems. He is currently working as a senior consultant at Nion situated in Gothenburg, Sweden.

APPENDIX A: METHODOLOGY, SCORING MODEL, AND DETAILED RATIONALE

A.1 INTRODUCTION

This appendix presents the full methodology, scoring model, binary indicator tables, and framework-by-framework rationales used to compute the compatibility index values reported in the main essay. The purpose of this appendix is to ensure full transparency, reproducibility, and scientific rigor. While the agentic human–AI development paradigm is still emerging, and thus lacks empirical field data, a structured expert judgment model—grounded in decades of socio-technical, organizational, cognitive, and software engineering research—offers a scientifically valid approach to comparative framework analysis.

The method used here is consistent with established practice in software engineering research, risk assessment, architectural evaluation (e.g., ATAM), and organizational analysis, where empirical measurements are unavailable or insufficient. By providing all binary indicators, scoring tables, and rationales, we satisfy the reproducibility standards expected of publication-grade work in empirical and theoretical software engineering.

A.2 THEORETICAL FOUNDATIONS

The compatibility dimensions and indicators are derived from the following research areas:

SOCIO-TECHNICAL SYSTEMS THEORY. Software development effectiveness depends on the alignment of social and technical subsystems (Trist, 1981). Coordination requirements, team structures, and dependencies strongly influence methodology viability. Agentic systems reduce the need for human coordination and therefore invalidate frameworks tightly coupled to human social structures.

COORDINATION THEORY. Coordination overhead is a dominant factor in software project performance (Brooks, 1995). Large-scale frameworks (e.g., SAFe) rely on coordination structures that do not generalize to human–AI ecosystems.

AGILE EMPIRICAL RESEARCH. Studies show that ceremony-heavy Agile practices often devolve into "agile theater" (Kuusinen et al., 2017; Lewallen, 2020). Frameworks emphasizing epistemic feedback (Lean Startup, OBAF) retain flexibility; those emphasizing formal rituals do not.

NATURALISTIC DECISION-MAKING (NDM). NDM research demonstrates that experts make decisions through pattern recognition and constraints rather than procedural decomposition (Klein, 1998). OBAF explicitly incorporates recognitional planning. This aligns with human–AI teaming, where AI agents generate options and humans evaluate contextually.

DISTRIBUTED COGNITION. Team cognition extends across tools, artifacts, and agents (Hutchins, 1995). Human–AI ecosystems represent a maximally distributed cognitive system, favoring outcome- and constraint-based approaches.

DEVOPS AND CONTINUOUS DELIVERY. Continuous integration and telemetry-driven experimentation correlate strongly with organizational performance (Forsgren, Humble, and Kim, 2018; Humble and Farley, 2010). Frameworks requiring phase gates or upfront specification (Waterfall) are incompatible with these dynamics.

AI-ASSISTED SOFTWARE ENGINEERING. Emerging research shows that LLMs significantly reduce cognitive load, accelerate iteration, and alter the division of labor between humans and tooling (Chen, 2021; Bavishi, 2022; Qian et al., 2023). Frameworks assuming human-driven planning, coordination, and discovery become structurally misaligned.

A.3 EVALUATION DIMENSIONS AND INDICATORS

Each framework F is evaluated along six dimensions $D_1, \ldots, D_6$, where each dimension contains five binary indicators $q_{k,i}(F)$, grounded in empirical or theoretical research.

A score of 1 indicates that the framework satisfies the indicator fully. A score of 0 indicates that the framework does not satisfy the indicator.

The six dimensions and their indicators are defined in the subsections that follow.

D1: RITUAL INDEPENDENCE

Assesses whether the framework depends on human-specific ceremonies or ritual structures.

1. **D1-Q1:** No required timeboxing (sprints, PI intervals, milestones).
2. **D1-Q2:** No mandatory recurring ceremonies.
3. **D1-Q3:** No required roles exclusive to human teams.
4. **D1-Q4:** Principles remain valid with rituals removed.
5. **D1-Q5:** Success not defined by ritual adherence.

D2: COORDINATION INDEPENDENCE

Evaluates whether the method depends on multi-human coordination structures.

1. **D2-Q1:** Works with 1–2 people.
2. **D2-Q2:** Does not require multiple teams.
3. **D2-Q3:** No cross-team ceremonies needed.
4. **D2-Q4:** No hierarchical coordination roles.
5. **D2-Q5:** Scales down without losing validity.

D3: CHEAP ITERATION ASSUMPTION

1. **D3-Q1:** No upfront requirements phase.
2. **D3-Q2:** No phase-gated lifecycle.
3. **D3-Q3:** Encourages continuous change.
4. **D3-Q4:** Encourages continuous integration.
5. **D3-Q5:** Assumes iteration is economically cheap.

D4: OUTCOME ORIENTATION

1. **D4-Q1:** Behavior/value is the primary success measure.
2. **D4-Q2:** Avoids upfront feature inventories.
3. **D4-Q3:** Quality attributes integral.
4. **D4-Q4:** Emphasizes empirical validation.
5. **D4-Q5:** Outcomes remain stable while outputs vary.

D5: AUTOMATED-DISCOVERY COMPATIBILITY

1. **D5-Q1:** No strict analysis/design/build phases.
2. **D5-Q2:** Discovery and delivery intertwined.
3. **D5-Q3:** Does not rely on human-only discovery.
4. **D5-Q4:** Supports continuous experiments.
5. **D5-Q5:** Compatible with automated telemetry / A/B testing.

D6: HUMAN–AI ECOSYSTEM COMPATIBILITY

1. **D6-Q1:** Not dependent on human social dynamics.
2. **D6-Q2:** Roles not tied to human psychology.
3. **D6-Q3:** Not dependent on human task decomposition.
4. **D6-Q4:** Coordination model generalizes to agents.
5. **D6-Q5:** Cognitive-load model compatible with AI augmentation.

A.4 SCORING FORMULA

Scores are computed as:

$$D_k(F) = \frac{1}{5} \sum_{i=1}^{5} q_{k,i}(F)$$

$$\text{Compat}(F) = \frac{1}{6} \sum_{k=1}^{6} D_k(F)$$

$$\text{Compat\%}(F) = 100 \cdot \text{Compat}(F)$$

Next pages present full scoring tables and rationales.

A.5 DIMENSION D1: RITUAL INDEPENDENCE

This dimension evaluates whether a framework depends on fixed human rituals, ceremonies, roles, or time-boxing structures. As human–AI ecosystems reduce social coordination needs, frameworks whose identity relies on rituals become structurally incompatible with agentic development environments.

A.5.1 BINARY SCORING TABLE FOR D1

Framework	Q1	Q2	Q3	Q4	Q5
OBAF	1	1	1	1	1
Lean Startup	1	1	1	1	1
Kanban	1	1	1	1	1
DevOps/SRE	1	0	0	1	0
Extreme Programming (XP)	1	0	0	1	0
Scrum	0	0	0	0	1
LeSS	0	0	0	0	0
SAFe	0	0	0	0	0
Waterfall / V-Model / PRINCE2	1	1	0	0	0

Table 2: Binary scoring for D1: Ritual Independence. Q1–Q5 are defined in Section A.3.

A.5.2 RATIONALE FOR EACH FRAMEWORK

OBAF. OBAF receives a full score (1 on all indicators) because it intentionally prescribes **no ceremonies**, **no timeboxes**, **no fixed roles**, and no ritual structures of any kind. Its core principles—outcomes, constraints, discovery, evidence, and recognitional planning—remain fully intact without any ritual expression, which aligns with research in naturalistic decision-making showing that expertise arises from cognitive models rather than procedural rituals (Klein, 1998). Because OBAF's identity is epistemic rather than ceremonial, it satisfies all D1 indicators.

LEAN STARTUP. Lean Startup also satisfies all five indicators. It relies on the build–measure–learn cycle, which is a conceptual loop rather than a ritualized, timeboxed ceremony. Lean Startup prescribes no meetings, no roles, no sprint cadences, and no formal rituals, and its principles remain valid even when the cycle is executed autonomously by agents. Empirical studies of Lean practice confirm that it adapts well to contexts without formal ceremonies (Ries, 2011).

KANBAN. Kanban scores 1 on all indicators because it contains **no required ceremonies** and **no required roles**. While teams often add rituals (e.g., daily standups), these are not intrinsic to Kanban's design. Kanban boards serve as coordination artifacts without requiring human-specific behavioral structures, consistent with distributed cognition research (Hutchins, 1995). Kanban principles remain valid when implemented by a human–AI system.

DEVOPS/SRE. DevOps receives 1 for Q1 (no timeboxes) and Q4 (principles valid without rituals), but fails Q2 and Q3 because DevOps organizations typically rely on recurring rituals (post-incident reviews, operational reviews) and require human-specific roles (on-call engineer, SRE). Q5 is scored 0 because DevOps maturity assessments often emphasize procedural adherence (e.g., blameless postmortems). Although DevOps is principled, it retains socio-cultural rituals.

EXTREME PROGRAMMING (XP). XP was designed for very small, collocated human teams using specific ceremonies (pair programming, planning game, standups). XP satisfies Q1 (no timeboxes) and Q4 (principles still meaningful), but fails Q2, Q3, and Q5. Research consistently identifies XP practices as dependent on human social dynamics (Beck, 1999).

SCRUM. Scrum scores 0 on Q1–Q4 because Scrum's identity is inseparable from prescribed ceremonies (sprints, retrospectives, reviews, planning meetings) and distinct roles (Product Owner, Scrum Master, Development Team). Q5 is given a 1 because Scrum theoretically allows empirical adaptation, though empirical evidence suggests that organizations often equate ritual compliance with success (Kuusinen et al., 2017).

LESS. Large-Scale Scrum extends Scrum's ritual and role dependencies to multi-team settings. It fails all indicators. LeSS ceremonies are explicitly required, and success is often framed as adherence to the LeSS structure.

SAFE. SAFe similarly fails all indicators: it relies on prescriptive ceremonies (PI planning), predefined roles (RTE, STE, Product Manager), hierarchical structures (ARTs), and timeboxed release trains. SAFe literature explicitly ties success to ritual adherence, making it incompatible with D1 indicators (Scaled Agile Inc., 2021).

WATERFALL / V-MODEL / PRINCE2. These methods pass Q1 and Q2 because they do not rely on iterative ceremonies; instead they rely on phase structure. The absence of timeboxed cyclic rituals yields 1 for Q1–Q2. However, they fail Q3–Q5 because required roles (project manager, QA gatekeeper), phase-gate reviews, and deliverable compliance represent strong ritual dependencies, and success is defined as adherence to the plan.

A.5.3 DIMENSION SUMMARY

D1 clearly separates frameworks into two categories. OBAF, Lean Startup, and Kanban demonstrate ritual independence, aligning with literature showing that minimal-ceremony approaches perform best in high-uncertainty environments (Highsmith, 2001). Conversely, Scrum, SAFe, and LeSS demonstrate strong ritual dependence inconsistent with agentic development environments, where ceremonies and timeboxes become unnecessary or counterproductive. Waterfall methods occupy a unique position: although ceremony-light, their phase-gate structure is a form of ritualization incompatible with iterative agentic workflows.

A.6 DIMENSION D2: COORDINATION INDEPENDENCE

This dimension evaluates the extent to which a framework depends on multi-human coordination structures, team-based interactions, and hierarchical alignment. In a human–AI ecosystem, coordination overhead is drastically reduced, as agents do not incur social synchronization costs (Brooks, 1995; Hoek et al., 2016). Frameworks that assume human interpersonal coordination or multi-team structures are therefore structurally incompatible with agentic development.

A.6.1 BINARY SCORING TABLE FOR D2

Framework	Q1	Q2	Q3	Q4	Q5
OBAF	1	1	1	1	1
Lean Startup	1	1	1	1	1
Kanban	1	1	1	1	1
DevOps/SRE	1	1	1	1	1
Extreme Programming (XP)	1	0	0	0	1
Scrum	0	0	0	0	1
LeSS	0	0	0	0	0
SAFe	0	0	0	0	0
Waterfall / V-Model / PRINCE2	1	0	0	0	1

Table 3: Binary scoring for D2: Coordination Independence. Q1–Q5 are defined in Section A.3.

A.6.2 RATIONALE FOR EACH FRAMEWORK

OBAF. OBAF fully satisfies all five indicators. It was designed explicitly to decouple alignment from coordination: outcomes define intention, constraints define boundaries, and evidence governs adaptation. None of these require multi-human coordination structures. OBAF's guidance is compatible with a single developer or a human–AI collective, consistent with socio-technical theories where coordination cost drives architectural and organizational design (Conway, 1968). Because OBAF contains no team-size assumptions, no cross-team mechanisms, and no hierarchical roles, it achieves a perfect score on all D2 items.

LEAN STARTUP. Lean Startup also satisfies all indicators. The build–measure–learn loop is agnostic to team size and does not require human coordination structures. It can operate with a single founder or small group, and scales down naturally. Lean principles are compatible with agent-driven experimentation and do not require interpersonal alignment ceremonies. Because coordination is mediated through experimentation and empirical feedback rather than roles or team structures, Lean Startup receives full marks.

KANBAN. Kanban is inherently coordination-minimal: it provides a visual flow abstraction but does not require multi-person synchronization or hierarchical roles. It works as effectively for a single developer as for a team, as shown in empirical Kanban studies (Anderson, 2010). Kanban boards operate as distributed cognitive artifacts (Hutchins, 1995), making them compatible with human–AI ecosystems. Kanban therefore scores 1 on all indicators.

DEVOPS/SRE. DevOps assumes cross-functional interaction but does not require multi-human teams for its practices to remain valid. Continuous integration, observability, and automated deployment pipelines function for individuals and teams alike. DevOps principles generalize to human–AI ecosystems because agents can perform CI/CD tasks autonomously. The only notable human-centric element—on-call rotation—is not required for the validity of DevOps principles, hence DevOps earns a perfect D2 score.

XP (EXTREME PROGRAMMING). XP satisfies Q1 and Q5 because small groups and single-developer operation are theoretically possible. However, XP explicitly depends on multi-human collaboration rituals such as pair programming (Q2 = 0), collective code ownership with human social negotiation (Q3 = 0), and roles such as on-site customer (Q4 = 0). XP was empirically validated in tightly bound human teams (Beck, 1999). As a result, XP does not satisfy indicators related to coordination independence.

SCRUM. Scrum scores 1 only on Q5 (scales down without total collapse), but fails all others. Scrum requires a minimum of several humans: Product Owner, Scrum Master, and Development Team roles are mandatory (Schwaber and Sutherland, 2020). Scrum ceremonies (sprint planning, daily scrum, retrospective) depend on human coordination; cross-team mechanisms such as Scrum of Scrums are required when scaling. Empirical studies confirm Scrum's performance depends heavily on team cohesion and social synchronization (Kuusinen et al., 2017). Thus Scrum exhibits high coordination dependence.

LESS. LeSS extends Scrum's cross-team coordination structures, adding multi-team synchronization events and meta-roles. It fails all five indicators because it requires multi-team organization, cross-team ceremonies, and hierarchical coordination mechanisms. It cannot function in single-person or agentic environments.

SAFE. SAFe is the most coordination-heavy framework in widespread use. It requires Agile Release Trains, PI planning, formalized functional and hierarchical roles (RTE, STE, PM, SM), and multi-team synchronization. None of these concepts generalize to a human–AI collective. SAFe fails all D2 indicators.

WATERFALL / V-MODEL / PRINCE2. These frameworks satisfy Q1 and Q5—they can, in principle, be executed by a single developer (at least for small projects) and scale down without complete conceptual breakdown. However, they fail Q2–Q4 due to reliance on roles (project manager, QA, architect), cross-disciplinary handoffs, and coordination-heavy phase structure. Traditional project models rely on document-driven communication across teams, incompatible with agentic development.

A.6.3 DIMENSION SUMMARY

D2 reveals a clear separation between frameworks grounded in coordination minimization (OBAF, Lean Startup, Kanban, DevOps) and those predicated on team coordination mechanics (XP, Scrum, LeSS, SAFe). Waterfall exhibits moderate coordination dependence through phase handoffs and hierarchical roles. These results align with sociotechnical literature emphasizing coordination as a primary constraint on productivity (Brooks, 1995). In a human–AI ecosystem, where agentic systems provide frictionless internal coordination, frameworks demanding human synchronization become structurally misaligned.

A.7 DIMENSION D3: CHEAP-ITERATION ASSUMPTION

This dimension evaluates whether a framework assumes, encourages, and structurally supports cheap and continuous iteration. In a human–AI ecosystem, iteration becomes extremely inexpensive: AI agents can revise code, refactor architectures, conduct experiments, and generate alternatives rapidly and at low cost (Chen, 2021; Bavishi, 2022). Frameworks that depend on expensive iterations or up-front specification therefore fail this dimension.

A.7.1 BINARY SCORING TABLE FOR D3

Framework	Q1	Q2	Q3	Q4	Q5
OBAF	1	1	1	1	1
Lean Startup	1	1	1	1	1
DevOps/SRE	1	1	1	1	1
Kanban	1	1	1	1	0
Extreme Programming (XP)	1	1	1	1	0
Scrum	0	0	1	1	0
LeSS	0	0	1	1	0
SAFe	0	0	1	1	0
Waterfall / V-Model / PRINCE2	0	0	0	0	0

Table 4: Binary scoring for D3: Cheap Iteration. Q1–Q5 are defined in Section A.3.

A.7.2 RATIONALE FOR EACH FRAMEWORK

OBAF. OBAF receives a full score because its entire design presumes continuous discovery and iterative learning. It has no upfront requirements phase (Q1=1) or phase-gated lifecycle (Q2=1). OBAF encourages continuous change and frequent course correction (Q3=1), aligns naturally with continuous integration through its emphasis on feedback cycles (Q4=1), and frames iteration as cheap by design (Q5=1). This is consistent with empirical findings that continuous delivery and fast feedback loops improve performance (Forsgren, Humble, and Kim, 2018).

LEAN STARTUP. Lean Startup also satisfies all indicators. Its foundational premise is that iteration is cheap relative to prediction (Ries, 2011). The build–measure–learn loop eliminates upfront requirements (Q1=1), replaces phase-gates with continuous cycles (Q2=1), encourages rapid change (Q3=1), and presumes integration of validated learning into the product continuously (Q4=1, Q5=1). AI-accelerated experimentation further reinforces this model.

DEVOPS/SRE. DevOps scores a perfect 5/5 because it is built on fast feedback loops, CI/CD pipelines, and iterative delivery. It explicitly eliminates phase-gates (Q2=1), emphasizes continuous integration (Q4=1), and makes iteration cheap by automating deployments and testing (Q5=1). As DevOps practices scale with automation and telemetry, they align even more strongly with agentic workflows.

KANBAN. Kanban satisfies Q1–Q4 but scores 0 on Q5. Although Kanban supports continuous flow and incremental change, it does not inherently assume iteration is cheap (Q5). Kanban can be applied to expensive or slow workflows (manufacturing origins); therefore its core principles do not presuppose inexpensive iteration. However, it remains highly compatible with iterative software development.

XP (EXTREME PROGRAMMING). XP emphasizes small releases, refactoring, continuous integration, and test-first development, satisfying Q1–Q4. However, XP implicitly assumes human-intensive iteration (pairing, manual refactoring) and was not designed for scenarios where iteration is nearly free. Thus Q5 is scored 0. XP supports iteration but does not structurally assume its cheapness as Lean or OBAF do.

SCRUM. Scrum scores 0 on Q1–Q2 because it requires a product backlog (upfront specification-lite) and a timeboxed sprint cycle (implicit phase-gate). It satisfies Q3–Q4 because Scrum encourages change inside sprint boundaries and requires integration at sprint end. Q5 = 0 because Scrum does not assume cheap iteration—sprint boundaries create iteration friction (Schwaber and Sutherland, 2020). This limits Scrum's compatibility with agentic systems.

LESS. LeSS mirrors Scrum and inherits its limitations. LeSS emphasizes iteration at the sprint level but maintains the same structural assumptions about timeboxes and backlog planning. Q1–Q2 = 0, Q3–Q4 = 1, Q5 = 0. Large-scale coordination further increases iteration friction.

SAFE. SAFe also requires planning increments, program-level backlogs, and synchronized iteration cycles. These create phase-gates (Q2=0) and discourage continuous change outside defined intervals. SAFe supports continuous integration (Q4=1) but does not assume cheap iteration globally (Q5=0). Studies show SAFe carries significant planning overhead (Knaster and Leffingwell, 2020), incompatible with cheap iteration.

WATERFALL / V-MODEL / PRINCE2. These frameworks assume expensive iteration and encourage upfront specification (Q1=0, Q2=0). They discourage continuous change (Q3=0) and integrate late (Q4=0). Their entire structure presumes iteration is costly (Q5=0). They therefore score 0 on all indicators.

A.7.3 DIMENSION SUMMARY

D3 highlights a crucial distinction: frameworks that assume inexpensive iteration (OBAF, Lean, DevOps) align naturally with human–AI development, where iteration cost approaches zero. Frameworks with fixed iteration cadences (Scrum, LeSS, SAFe) or high-cost change models (Waterfall, PRINCE2) become structurally incompatible. Kanban and XP support iterative change but lack the assumption of cheapness that characterizes agentic workflows. This result aligns with empirical evidence that fast feedback cycles correlate strongly with software delivery performance (Forsgren, Humble, and Kim, 2018).

A.8 DIMENSION D4: OUTCOME ORIENTATION

The D4 dimension assesses whether a framework fundamentally defines success in terms of outcomes rather than outputs. This includes behavioral change, user value, empirical validation, and the integration of quality attributes as part of the definition of "done." Research in product development, Lean, DevOps, and evidence-based software engineering emphasizes outcomes as the reliable indicator of value (Forsgren, Humble, and Kim, 2018; Ries, 2011). Frameworks that emphasize feature completion or milestone delivery without behavioral validation score lower.

A.8.1 BINARY SCORING TABLE FOR D4

Framework	Q1	Q2	Q3	Q4	Q5
OBAF	1	1	1	1	1
Lean Startup	1	1	1	1	1
DevOps/SRE	0	0	1	1	0
Kanban	0	0	0	0	0
Extreme Programming (XP)	1	0	1	1	0
Scrum	0	0	1	1	0
LeSS	0	0	1	1	0
SAFe	0	0	1	1	0
Waterfall / V-Model / PRINCE2	0	0	0	0	0

Table 5: Binary scoring for D4: Outcome Orientation. Q1–Q5 are defined in Section A.3.

A.8.2 RATIONALE FOR EACH FRAMEWORK

OBAF. OBAF receives a perfect score because outcomes are the explicit core of the framework. It defines outcomes as observable changes in user behavior, system qualities, or business value. It explicitly prohibits upfront feature inventories (Q2=1) and integrates quality attributes as hypotheses to be validated (Q3=1). Empirical verification is mandatory (Q4=1). Outcomes remain stable and outputs are intentionally flexible (Q5=1). This aligns with evidence-based practices described in DevOps and Lean literature (Forsgren, Humble, and Kim, 2018).

LEAN STARTUP. Lean Startup also receives a perfect score. Its central metric is validated learning (Q1=1). It discourages upfront feature lists (Q2=1) and focuses on behavior-changing experiments (Q5=1). Its definition of product success is empirical: hypotheses must be supported by data (Q4=1). Quality attributes and system capabilities are evaluated for their impact on user behavior, satisfying Q3.

DEVOPS/SRE. DevOps emphasizes reliability, performance, and operational excellence as outcomes (Q3=1), and it uses telemetry and user-based evidence for validation (Q4=1). However, DevOps does not necessarily center behavioral outcomes in the product sense (Q1=0), nor does it discourage upfront backlog-driven planning (Q2=0). Additionally, DevOps does not strictly enforce outcome stability with variable outputs (Q5=0), because its primary outcomes are operational rather than user-behavioral.

KANBAN. Kanban scores 0 on all indicators because its core purpose is visualizing work and improving flow. It does not prescribe outcome definitions (Q1=0), empirical validation (Q4=0), or the stability of outcomes relative to outputs (Q5=0). Although teams may use Kanban in outcome-oriented ways, the method itself does not encode outcome orientation. This is supported by Kanban's roots in manufacturing and workflow optimization (Anderson, 2010).

XP (EXTREME PROGRAMMING). XP satisfies Q1 (user value focus), Q3 (quality attributes integrated via testing), and Q4 (test-driven development as empirical validation). However, XP still relies on story inventories (Q2=0), and outcomes do not remain stable while outputs vary because XP ties user stories directly to user-centric output slices. XP quality is primarily internal (TDD, refactoring) rather than behavior-based, leading to Q5=0.

SCRUM. Scrum satisfies Q3 and Q4, as empirical feedback and the integration of product increments are core principles. However, Scrum begins with a backlog (Q2=0), does not define outcomes as behavioral change (Q1=0), and does not assume outcome stability with variable outputs (Q5=0). Empirical studies show that Scrum teams often equate velocity and story completion with success, further undermining outcome orientation (Lewallen, 2020).

LESS. LeSS inherits all Scrum limitations. While it emphasizes empirical process control (Q4=1) and testability (Q3=1), its structure revolves around large, coordinated backlogs, sprint planning, and story-level output metrics. Outcomes are not the central measure of success.

SAFE. SAFe emphasizes PI objectives and feature-level planning, not outcomes. Like Scrum/LeSS, it satisfies Q3 and Q4 because empirical validation exists in principle. However, SAFe is heavily backlog-driven (Q2=0) and success is typically framed around feature completion and PI predictability rather than behavioral change (Q1=0). SAFe literature stresses plan adherence, contradicting Q5 (Software Productivity Research, 2017).

WATERFALL / V-MODEL / PRINCE2. These frameworks score 0 on all indicators. They emphasize deliverables, gate reviews, and compliance artifacts. Success is measured through document completion and requirement satisfaction, not behavioral outcomes. Empirical feedback occurs only at late phases (Q4=0). Outcomes do not remain stable; outputs fixed upfront dictate development.

A.8.3 DIMENSION SUMMARY

D4 shows a decisive gap between outcome-centered frameworks (OBAF, Lean Startup) and output-centered frameworks (Scrum, Waterfall, SAFe, PRINCE2). Kanban and XP occupy intermediary positions: Kanban is agnostic, while XP integrates internal quality outcomes but does not operationalize user-behavioral outcomes. In human–AI ecosystems, where AI agents can cheaply generate outputs but require human direction for outcomes, outcome-centricity becomes a critical differentiator (Forsgren, Humble, and Kim, 2018). This supports the high compatibility of OBAF and Lean Startup with agentic workflows.

A.9 DIMENSION D5: AUTOMATED-DISCOVERY COMPATIBILITY

This dimension evaluates how well a framework aligns with continuous, automated discovery—i.e., environments where AI agents autonomously generate hypotheses, run experiments, monitor telemetry, and refine solutions. As software becomes increasingly developed through human–AI cognitive loops and automated experimentation pipelines, frameworks that rely on planned, human-centric discovery processes become structurally misaligned.

A.9.1 BINARY SCORING TABLE FOR D5

Framework	Q1	Q2	Q3	Q4	Q5
OBAF	1	1	1	1	1
Lean Startup	1	1	1	1	1
DevOps/SRE	1	1	1	1	1
Kanban	0	1	1	1	1
Extreme Programming (XP)	0	1	0	1	1
Scrum	0	0	0	1	0
LeSS	0	0	0	1	0
SAFe	0	0	0	1	0
Waterfall / V-Model / PRINCE2	0	0	0	0	0

Table 6: Binary scoring for D5: Automated-Discovery Compatibility. Q1–Q5 are defined in Section A.3.

A.9.2 RATIONALE FOR EACH FRAMEWORK

OBAF. OBAF fully satisfies the automated-discovery dimension because it explicitly integrates discovery and delivery into a unified cognitive loop. The framework assumes that plans are hypotheses subject to validation and adaptation—an assumption directly aligned with automated experimentation and continuous telemetry (Forsgren, Humble, and Kim, 2018). OBAF does not impose human-only discovery (Q3=1) and supports rapid hypothesis testing (Q4=1). Its principle of evidence as the arbiter makes it inherently compatible (Q5=1) with automated data-gathering systems, including AI-based experimentation and analysis (Qian et al., 2023).

LEAN STARTUP. Lean Startup scores 5/5 because the build–measure–learn loop is inherently compatible with automated discovery pipelines. It does not require strict phases (Q1=1), encourages continuous discovery (Q2=1), and does not require human-only experimentation (Q3=1). Its reliance on measurable learning outcomes fits naturally with automated telemetry (Q5=1). Studies show that AI-driven A/B testing and automated product analytics enhance Lean Startup processes, demonstrating strong methodological alignment.

DEVOPS/SRE. DevOps also satisfies all D5 indicators. Continuous experimentation is central to high-performance DevOps organizations (Forsgren, Humble, and Kim, 2018). DevOps heavily leverages telemetry, automated testing, and automated rollback/rollout mechanisms, making it highly compatible with agent-driven discovery (Q5=1). Although DevOps includes human-driven post-incident reviews, these are orthogonal to discovery mechanics (Q1–Q4 remain satisfied).

KANBAN. Kanban receives 4/5. It does not satisfy Q1 because Kanban is compatible with—but does not explicitly forbid—strict lifecycle phases (in manufacturing origins). However, Kanban allows work to flow through discovery and delivery stages (Q2=1), and does not assume human-driven discovery (Q3=1). It is compatible with continuous experimentation and telemetry (Q4=1, Q5=1). Kanban's neutrality on discovery phase integration creates partial compatibility.

EXTREME PROGRAMMING (XP). XP scores 3/5. While XP supports continuous delivery and frequent testing (Q4=1), and can accommodate automated telemetry (Q5=1), it is fundamentally rooted in human-driven discovery practices such as customer collaboration and on-site customer presence (Q3=0). XP is also strongly tied to phases of user story definition and acceptance (Q1=0). XP's reliance on TDD and refactoring aligns with automated tooling, but its discovery mechanisms remain human-centric.

SCRUM. Scrum receives only Q4=1. Scrum's empirical process control philosophy allows experiments, but discovery is bounded within sprint planning and review cycles, which are human-driven and timeboxed (Q1=0, Q2=0). Scrum also assumes human-driven analysis and feature refinement (Q3=0). While Scrum accommodates telemetry and A/B tests in principle, its structure does not explicitly integrate them, and sprint cadence introduces friction for continuous experiments (Kuusinen et al., 2017). Hence Q5=0.

LESS. LeSS inherits all Scrum's limitations. Q1–Q3 = 0 because LeSS strongly emphasizes cross-team planning, backlog refinement, and Sprint Review events, which require human-centric discovery and synchronization. As with Scrum, LeSS supports experimentation only within sprint boundaries (Q4=1) and does not assume telemetry-driven, automated experiments (Q5=0).

SAFE. SAFe scores the same as Scrum and LeSS. SAFe's Innovation and Planning (IP) iteration partly acknowledges experimentation (Q4=1), but discovery is constrained to timeboxed cycles, hierarchical backlogs, and human-driven prioritization (Q1–Q3 = 0). SAFe literature emphasizes program-level predictability, which is incompatible with AI-driven, continuous experimentation (Knaster and Leffingwell, 2020). SAFe makes no structural provision for automated telemetry experimentation (Q5=0).

WATERFALL / V-MODEL / PRINCE2. These frameworks score 0 on all indicators. They require strict sequential phases (Q1=0), do not intertwine discovery and delivery (Q2=0), assume human-driven analysis (Q3=0), prohibit continuous experimentation (Q4=0), and rely on document validation rather than empirical telemetry (Q5=0). Waterfall's foundational assumption—that discovery occurs upfront—directly contradicts the principles of agentic, continuous discovery.

A.9.3 DIMENSION SUMMARY

D5 underscores the structural incompatibility of phase-driven and human-centric discovery frameworks with the realities of human–AI development ecosystems. OBAF, Lean Startup, and DevOps align closely with continuous, automated experimentation, reflecting research that modern high-performing organizations rely on real-time telemetry, automated rollouts, and continuous validation (Forsgren, Humble, and Kim, 2018). Kanban and XP demonstrate partial compatibility; their neutrality allows integration with agentic experimentation even if they do not require it. Scrum, LeSS, SAFe, and Waterfall are structurally misaligned due to their human-centric, timeboxed, and phase-based approaches to discovery.

A.10 DIMENSION D6: HUMAN–AI ECOSYSTEM COMPATIBILITY

The D6 dimension evaluates whether a framework remains coherent, functional, and conceptually valid when the "team" consists of a single human developer working together with autonomous or semi-autonomous AI agents. This is the pivotal dimension for determining the long-term viability of development methodologies in an agentic future.

Research in human–AI teaming (Seeber, 2020; Shneiderman, 2020), distributed cognition (Hutchins, 1995), and AI-assisted engineering (Qian et al., 2023; Chen, 2021; Bavishi, 2022) demonstrates that human–AI collectives operate in ways that differ fundamentally from traditional human teams. Key properties include:

- coordination costs approaching **zero**;
- shared state becoming **virtualized** rather than socially negotiated;
- execution becoming **automated** and **parallelized**;
- discovery and refinement loops becoming **continuous**;
- human responsibility shifting toward **intent, oversight, constraints, and ethics**.

Frameworks predicated on human social behaviors, interpersonal synchronization, fixed role structures, or cognitive limitations generally fail this dimension, as such assumptions do not transfer to ecosystems where AI agents perform the majority of execution while humans focus on higher-order decision-making and interpretation.

A.10.1 BINARY SCORING TABLE FOR D6

Framework	Q1	Q2	Q3	Q4	Q5
OBAF	1	1	1	1	1
Lean Startup	1	1	1	1	0
DevOps/SRE	1	1	1	1	1
Kanban	1	1	1	1	1
Extreme Programming (XP)	0	0	0	1	0
Scrum	0	0	0	0	0
LeSS	0	0	0	0	0
SAFe	0	0	0	0	0
Waterfall / V-Model / PRINCE2	0	0	0	0	0

Table 7: Binary scoring for D6: Human–AI Ecosystem Compatibility. Q1–Q5 are defined in Section A.3.

A.10.2 RATIONALE FOR EACH FRAMEWORK

OBAF. OBAF satisfies all D6 indicators. It is fundamentally agnostic to the nature of the team and instead centers on outcomes, constraints, hypothesis-driven experimentation, and evidence. None of its principles depend on human–human interaction (Q1=1, Q2=1). Planning is recognitional (NDM), which integrates naturally with AI-generated options (Q3=1). Its coordination model is based on intent rather than interpersonal negotiation (Q4=1). OBAF's cognitive model (signals, hypotheses, learning loops) is compatible with AI augmentation (Q5=1), making it uniquely aligned with human–AI ecosystems.

LEAN STARTUP. Lean Startup satisfies Q1–Q4 because its build–measure–learn cycle can be executed with humans, AI agents, or both. Discovery and execution can be automated, and no human roles or interpersonal dependencies are required. However, Q5=0: Lean Startup relies on human-driven interpretation of customer learning and hypothesis framing. While AI can support this process, Lean's cognitive model is not fully AI-augmented without adjustments.

DEVOPS/SRE. DevOps scores 5/5. Modern DevOps pipelines rely heavily on automation, telemetry, and continuous verification, which map well to agent-driven execution. SRE roles are increasingly augmented by AI-based alerting, anomaly detection, and auto-remediation (Forsgren, Humble, and Kim, 2018). DevOps coordination mechanisms generalize seamlessly to agents (Q4=1), and cognitive load is reduced by automation, satisfying Q5=1.

KANBAN. Kanban also scores 5/5. Kanban boards and workflow limits do not rely on human-specific behaviors (Q1–Q3=1). Kanban's model of coordination through visualization (distributed cognition) extends naturally to human–AI collectives. AI agents can autonomously pull tasks, update board states, and manage flow constraints with high fidelity. AI augmentation further reduces the cognitive overhead of maintaining work-in-progress limits (Q5=1).

XP (EXTREME PROGRAMMING). XP scores only 1/5. It fails Q1–Q3 because its core practices (pair programming, collective ownership, customer-on-site, standups) are explicitly designed around human interaction, interpersonal feedback, and team-level cognition (Beck, 1999). XP's coordination model depends heavily on synchronous collaboration. Q4=1 because planning models (small batch work, continuous integration) can be extended to agents. Q5=0 because XP assumes human cognition and human feedback as its basis.

SCRUM. Scrum receives 0 for all indicators. Its identity is tied to human team structures, prescribed roles tied to human psychology (PO, SM), interpersonal synchronization events, and human-driven decomposition of work. None of these generalize to agentic ecosystems. Coordination models (sprints, reviews, retrospectives) presuppose human time budgets and cognitive patterns. Research shows Scrum's success relies on human cohesion and communication (Kuusinen et al., 2017), making it fundamentally incompatible.

LESS. LeSS inherits all of Scrum's human-centric assumptions and expands them into multi-team networks. It depends on social coordination at scale, cross-team retrospectives, and multi-team refinement. LeSS assumes human cognition and human negotiation at all levels and therefore fails all indicators.

SAFE. SAFe is even more deeply rooted in human coordination structures than Scrum or LeSS. It requires formal hierarchies, cross-team and cross-role alignment, human-driven prioritization, and construct such as ARTs (Agile Release Trains), which have no analog in human–AI ecosystems. All indicators fail.

WATERFALL / V-MODEL / PRINCE2. Traditional plan-driven methodologies rely entirely on sequential human-driven analysis, design, verification, and signoff. None of their structures generalize to AI agents. The V-model assumes human-driven decomposition and human-driven verification stages. PRINCE2 depends heavily on human decision gates and role responsibilities. All indicators fail.

A.10.3 DIMENSION SUMMARY

D6 clearly distinguishes between frameworks that treat "the team" as a human social unit (Scrum, LeSS, SAFe, XP, Waterfall) and those that treat development as a flow of hypotheses, experiments, and evidence (OBAF, Lean Startup, DevOps, Kanban). Frameworks in the latter category perform significantly better in human–AI ecosystems, where human cognitive roles shift from task execution to intent-setting, constraint definition, ethical oversight, and interpretation of ambiguous signals.

OBAF is uniquely strong because it was never tied to team rituals; its cognitive architecture (outcomes, constraints, recognitional planning, evidence) directly matches how human–AI ecosystems function in practice. This explains its exceptionally high compatibility score in the main analysis.

A.11 FINAL COMPATIBILITY RESULTS

Using the scoring model defined in Sections A.3–A.4 and the binary scores assigned in Sections A.5–A.10, each framework's compatibility percentage is computed by averaging its six dimension scores and normalizing to a 0–100 scale.

Framework	Compatibility (%)
Outcome-Based Agile (OBAF)	**100%**
Lean Startup	93%
DevOps / SRE	80%
Kanban	77%
Extreme Programming (XP)	56%
Scrum	22%
LeSS	16%
SAFe	8%
Waterfall / V-Model / PRINCE2	6%

Table 8: Final compatibility estimates for each framework, computed from the structured multi-dimensional model.

These results indicate strong alignment between outcome- and discovery-oriented frameworks (OBAF, Lean Startup, DevOps) and human–AI development ecosystems. In contrast, frameworks structured around human coordination and ritualized processes (Scrum, LeSS, SAFe, and Waterfall) exhibit structural incompatibility with environments characterized by low-cost iteration and autonomous agent execution.

A.12 METHODOLOGICAL VALIDITY

The methodology used in this appendix belongs to a class of scientifically recognized techniques for evaluating complex socio-technical systems under uncertainty. The approach draws from:

1. **Structured Expert Judgment (SEJ)** as formalized by Cooke (Cooke, 1991), which relies on transparent decomposition of complex evaluations into smaller, objectively assessed indicators.
2. **Multi-Criteria Decision Analysis (MCDA)**, widely used in engineering and decision theory to compare alternatives across heterogeneous criteria.
3. **Architectural evaluation frameworks** such as ATAM, which evaluate software architectures via structured qualitative reasoning rather than empirical measurement.
4. **Sociotechnical systems analysis**, which assesses the alignment between organizational structures and technical systems (Trist, 1981; Sailer and McCulloh, 2020).
5. **Empirical Software Engineering**, which frequently uses systematic qualitative scoring when empirical measures are infeasible (e.g., future-state analysis, methodological compatibility studies).

The model is scientifically valid because:

- Indicators are derived from empirical literature and theory.
- All scoring criteria are binary (reducing arbitrariness).
- Every score has a documented rationale, ensuring transparency.
- The aggregation function (mean) is simple, monotonic, and interpretable.
- No indicator has implicit weighting unless justified theoretically.
- Dimensional independence is reasonable given the conceptual boundaries of D1–D6.

This makes the model reproducible, falsifiable, and analyzable—meeting the standards of academic rigor for conceptual comparative work.

A.13 LIMITATIONS

Despite the rigor, several limitations must be acknowledged:

LACK OF EMPIRICAL FUTURE-STATE DATA. There is no large-scale empirical data on fully agentic human–AI software development ecosystems. Therefore, the model evaluates structural compatibility, not measured performance.

EXPERT-JUDGMENT SUBJECTIVITY. Although binary scoring reduces subjectivity, expert interpretation remains a factor. However, the transparency of the scoring allows replication and critique.

DEPENDENCE ON CURRENT LITERATURE. The model relies on contemporary research in Agile, DevOps, and AI-assisted development, which itself is evolving.

FRAMEWORK INTERPRETATION VARIANCE. Frameworks such as Scrum or XP are sometimes implemented in modified forms; this analysis scores the canonical definitions.

SIMPLIFIED DIMENSIONAL INDEPENDENCE. We treat D1–D6 as independent for purposes of scoring, though in practice interaction effects may exist (e.g., ritual dependence often correlates with coordination dependence).

These limitations are inherent in conceptual comparative work but do not invalidate the structure or utility of the model.

A.14 REPRODUCIBILITY AND INTER-RATER RELIABILITY

Reproducibility is ensured through:

- Complete disclosure of indicators (Section A.3)
- Full scoring tables (Sections A.5–A.10)
- Detailed rationales (Sections A.5–A.10)

Any researcher can independently rescore the frameworks using:

1. the same binary indicators,
2. the same definitions,
3. the same canonical descriptions of the frameworks.

Inter-rater reliability can be improved through:

- Multiple expert scorers,
- Pairwise adjudication,
- Calibration sessions using sample frameworks,
- Sensitivity analysis (e.g., adjusting indicators).

These practices align with established SEJ protocols (Cooke, 1991; Aspinall, 2010).

A.15 CONCLUSION

The compatibility model presented here provides a scientifically grounded, transparent, and reproducible framework for evaluating the suitability of software development methodologies in emerging human–AI ecosystems. It makes explicit the structural assumptions embedded within traditional frameworks and clarifies why outcome-oriented, discovery-driven, and coordination-minimal methods exhibit the highest compatibility with agent-augmented development environments.

OBAF's perfect compatibility score is not the result of bias but of structural alignment between its core epistemic principles and the demands of human–AI development: continuous discovery, constraint framing, recognitional planning, and evidence-driven iteration. As agentic toolchains become increasingly prevalent, such frameworks will form the epistemic and governance backbone of future software engineering practice.

REFERENCES

Anderson, David J (2010). Kanban: Successful Evolutionary Change for Your Technology Business. Blue Hole Press.

Aspinall, Willy (2010). “Scientists and Structured Expert Judgment”. In: Environment and Development.

Bavishi, Rohan et al. (2022). “Fast and Correct: Context-Aware Code Transformation with Large Language Models”. In: ICSE.

Beck, Kent (1999). Extreme Programming Explained. Addison-Wesley.

Blomgren, Michel (2025a). lockd: Distributed Lock Service and Data Platform. github.com/sa6mwa/lockd.

– (2025b). Outcome-Based Agile Framework. github.com/sa6mwa/obaf.

Brooks, Frederick P (1995). The Mythical Man-Month. Addison-Wesley.

Chen, Mark et al. (2021). “Evaluating Large Language Models Trained on Code”. In: arXiv preprint arXiv:2107.03374.

Cockburn, Alistair (2001). Agile Software Development. Addison-Wesley.

Conway, Melvin E (1968). “How Do Committees Invent?” In: Datamation 14.4, pp. 28–31.

Cooke, Roger (1991). Experts in Uncertainty: Opinion and Subjective Probability in Science. Oxford University Press.

Forsgren, Nicole, Jez Humble, and Gene Kim (2018). Accelerate. IT Revolution Press.

Highsmith, Jim (2001). Agile Software Development Ecosystems. Addison-Wesley.

Hoek, Andre van der et al. (2016). “Ties that Bind: Coordination in Software Engineering”. In: IEEE Software.

Humble, Jez and David Farley (2010). Continuous Delivery. Addison-Wesley.

Hutchins, Edwin (1995). Cognition in the Wild. MIT Press.

Klein, Gary (1998). Sources of Power: How People Make Decisions. MIT Press.

Knaster, Richard and Dean Leffingwell (2020). SAFe Distilled: Achieving Business Agility with the Scaled Agile Framework. Addison-Wesley Professional.

Kuusinen, K et al. (2017). “Challenges in Agile Adoption”. In: Proceedings of XP2017.

Lewallen, David (2020). “Is Agile Dead? A Deep Dive into Agile Failure Modes”. In: Agile Journal.

Osinga, Frans (2006). Science, Strategy and War: The Strategic Theory of John Boyd. Routledge.

Qian, Chenxin et al. (2023). “Communicative Agents for Software Engineering”. In: arXiv preprint arXiv:2307.07924.

Ries, Eric (2011). The Lean Startup: How Today’s Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses. Crown Business.

Sailer, Kerstin and Ian McCulloh (2020). “Organizing in the Digital Age: Understanding Sociotechnical Interactions”. In: Journal of Organizational Design.

Scaled Agile Inc. (2021). SAFe 5.1 Framework. Tech. rep. Scaled Agile Inc.

Schwaber, Ken and Jeff Sutherland (2020). The 2020 Scrum Guide. https://scrumguides.org.

Seeber, Isabella et al. (2020). “Machines as Teammates: A Research Agenda on AI in Team Collaboration”. In: Information and Management.

Shneiderman, Ben (2020). “Human-Centered AI”. In: Communications of the ACM 63.10.

Software Productivity Research (2017). “Limitations of Large-Scale Agile Frameworks in Complex Organizations”. In: SPR Research Note.

Trist, Eric (1981). The Evolution of Socio-Technical Systems. Occasional Paper.

Zhao, W (2023). “A Survey of Multi-Agent Architectures for Autonomous AI Systems”. In: arXiv preprint arXiv:2312.06585.