Engineering Verification Through Independent Reasoning

A Reflective Engineering Case Study and Exploratory Methodology for AI-Assisted Engineering Review

Project ID	PS-ERS-001
Document	Working Paper 1.0
Status	PUBLIC RELEASE — ONGOING RESEARCH PROGRAMME
Released	01 Aug. 2026
Author	Gabor Zoltan Nyarfadi
Published by	Poplar Systems Ltd
Company No.	17100764

Poplar Systems Ltd. — Engineering Research Series

Contents

Publication Note

This released working paper shares an observation made inside an active engineering workshop. The software platform remains under development, substantial implementation work remains, and the cases reported here belong to the development phase rather than to mature production operation.

The concepts and interpretations are therefore provisional. The purpose of this publication is to make the work visible, describe what has been observed so far, and invite informed scrutiny. Working Paper 1.0 is a stable, citable release of PS-ERS-001, not a claim that the underlying engineering programme or theory is complete.

Scope boundary. PS-ERS-001 covers the verification phenomena analysed to date. Later experimental prompts and newly observed behaviour that have not yet been examined are intentionally excluded; they may form the subject of PS-ERS-002.

1. Introduction

Engineering has always relied on observation, measurement, experimentation, and reproducibility. Whether designing a bridge, validating a control system, or commissioning an industrial installation, confidence is not created by intuition alone. It is built through independent verification, repeatable procedures, and evidence that withstands critical examination.

Software engineering follows the same principles. Modern development processes rely on peer reviews, automated testing, architecture reviews, security audits, and operational monitoring to reduce uncertainty before systems reach production. Each additional layer of verification does not guarantee correctness, but it increases confidence that critical errors have been identified before deployment.

The emergence of large language models (LLMs) introduces a new opportunity and a new challenge. AI systems can now analyze source code, documentation, architectures, and operational decisions with remarkable speed. However, they also introduce a fundamental problem: how should engineers verify the output of an AI system when the AI itself may be incorrect?

This question cannot be answered by simply choosing a stronger model. Every AI system possesses different strengths, weaknesses, assumptions, and reasoning styles. A single highly capable model may still overlook architectural inconsistencies, misinterpret domain-specific requirements, or confidently produce an incorrect conclusion. Traditional engineering would describe this as a single point of failure.

This research originated from a practical engineering project rather than from an academic laboratory. During the development of the Dragon Tools platform, multiple independent AI systems were gradually introduced into the engineering workflow. Initially, each model served a conventional role as an assistant for coding, documentation, or review. Over time, however, an unexpected pattern emerged. Different models repeatedly identified different categories of problems while frequently converging on the same final engineering conclusions through different reasoning paths.

This paper is therefore positioned as a reflective engineering case study. It does not report a controlled comparison of AI models. It reconstructs recurring verification behaviour observed during real engineering work and uses that experience to propose an exploratory methodology for further application and independent evaluation.

This observation fundamentally changed the objective of the project. The primary research question was no longer whether artificial intelligence could assist software development. Instead, the investigation shifted toward a broader engineering problem:

> *Can independent reasoning performed by multiple AI systems be organized into a reproducible engineering verification methodology?*

Rather than comparing benchmark scores or model capabilities, this work focuses on engineering processes. The objective is to understand whether multiple independent reasoning systems can increase confidence in engineering decisions through structured disagreement, evidence-based criticism, and reproducible consensus.

Throughout the research, the participating AI systems were intentionally treated as independent reviewers rather than collaborative agents. They were asked to analyze identical engineering problems separately, without being influenced by the reasoning process of the others. Their conclusions were then compared, challenged, and synthesized using conventional engineering review principles.

An important outcome of this research was that the AI systems themselves were not the primary discovery. Instead, the investigation gradually revealed a broader methodology describing how engineering confidence can be progressively increased through independent verification, structured criticism, and evidence-based convergence. Multi-model AI verification therefore became only one implementation of a more general engineering verification framework.

Consequently, this paper should not be interpreted as a study of artificial intelligence itself. It is an investigation into engineering methodology, using multiple AI systems as experimental participants. The proposed concepts—including progressive engineering verification, multi-model consensus, and knowledge authority—are presented not as fixed implementations but as reproducible engineering principles that may remain applicable even as future AI systems evolve.

The contribution of this paper is therefore twofold. First, it provides a reflective account of verification practices that emerged during a real software engineering project. Second, it proposes that independent artificial reasoning can become an additional verification layer within established engineering practice, provided that the process remains transparent, traceable, evidence-driven, and continuously open to criticism. The proposed concepts are methodological propositions, not claims of universal or experimentally proven behavior.

Chapter 2 — Background and Motivation

2.0 Related Foundations

The proposed methodology sits at the intersection of established engineering verification, software diversity, AI risk management, and recent work on multiple reasoning paths. N-version software research demonstrates both the promise of independent implementations and the danger of assuming that their failures are truly independent [1]. Contemporary AI research likewise shows benefits from self-consistency and multi-agent debate while also identifying limits in unsupported self-correction [2-4]. The NIST AI Risk Management Framework provides a broader governance context in which testing, evaluation, verification, validation, documentation, and human accountability remain continuous lifecycle concerns [5].

2.1 Engineering Verification Before Artificial Intelligence

Engineering has never relied on a single opinion. Long before the emergence of software engineering or artificial intelligence, complex engineering disciplines developed systematic methods to reduce uncertainty through independent verification. The underlying principle remained remarkably consistent across industries: confidence is not established by authority but by repeatable examination from multiple independent perspectives.

In civil engineering, structural calculations are routinely reviewed by independent engineers before construction begins. Aerospace projects employ multiple design reviews and certification authorities before aircraft enter service. Nuclear facilities require layered verification procedures in which independent teams validate calculations, safety analyses, and operational assumptions. Medical devices undergo repeated technical, regulatory, and clinical evaluations before approval. Although these industries differ significantly in technology and regulation, they share a common engineering philosophy: critical decisions should not depend on a single reviewer.

This principle extends beyond design. Engineering verification continues throughout the entire lifecycle of a system. Construction inspections, commissioning procedures, operational monitoring, maintenance audits, and periodic recertification all represent independent verification activities performed after the original engineering work has been completed. Verification is therefore not a single event but a continuous process accompanying the system from conception to retirement.

The same philosophy governs everyday engineering practice in industrial environments. Consider the commissioning of a Building Management System (BMS). A control strategy may be designed by one engineer, implemented by another, tested by a commissioning engineer, reviewed by an independent consultant, and finally accepted by the client. None of these participants necessarily possesses greater authority than the others. Instead, each contributes a different perspective that reduces the probability of undetected errors remaining within the final system.

An important observation emerges from these examples. Independent verification is valuable not because reviewers are expected to agree, but because they frequently disagree. Disagreement forces assumptions to become explicit, hidden constraints to surface, and engineering evidence to be examined more critically. In this sense, disagreement is not an obstacle to engineering confidence but one of its primary mechanisms.

This distinction is fundamental for the research presented in this paper. Traditional engineering does not attempt to eliminate differing opinions. Instead, it provides structured processes through which disagreements are investigated, evidence is collected, and justified conclusions are eventually reached. Confidence therefore grows through critical examination rather than through immediate consensus.

For this reason, engineering verification should be understood as a process of progressively reducing uncertainty rather than proving absolute correctness. Every independent review, inspection, simulation, or validation activity contributes additional information. None of them individually guarantees correctness. Collectively, however, they increase confidence that the engineering decision can withstand further scrutiny.

This philosophy forms the foundation upon which the remainder of this research is built. The introduction of artificial intelligence does not invalidate traditional engineering verification principles. Instead, it raises a new question: if engineering has relied for decades on independent human verification, can independent artificial reasoning become an additional verification layer without abandoning the principles that engineering has always considered essential?

2.2 Verification in Modern Software Engineering

Modern software engineering applies verification through several specialized mechanisms rather than through a single universal test. Compilation, static analysis, unit testing, integration testing, architecture review, security assessment, operational monitoring, and production telemetry each examine a different property of the system.

These mechanisms are intentionally non-equivalent. Compilation can confirm that a program can be built, but not that its architecture is coherent. Unit tests can demonstrate expected component behavior, but not necessarily correct interaction between components. Security review can reveal attack paths that functional testing was never designed to detect. Operational monitoring can expose failures that become visible only under real workloads.

The practical strength of software verification therefore lies in coverage by difference. Each layer is limited, but the limitations are not identical. When independent mechanisms examine the same system from different perspectives, their combined evidence reduces the probability that a single class of defect remains invisible.

This layered arrangement also establishes an important boundary: passing one verification activity does not validate the entire system. It demonstrates only that no relevant defect was identified by that activity under the conditions in which it was performed. Engineering confidence accumulates across layers; it is not transferred from one successful check to all other concerns.

Software engineering consequently provides a mature precedent for combining partial and independent evidence. The discipline does not expect one reviewer, test suite, or analytical tool to establish correctness alone. It instead organises multiple forms of verification so that their strengths overlap while their blind spots differ.

The relevance of this precedent to AI-assisted engineering is structural rather than technological. Artificial reasoning does not replace established verification layers. The question is whether it can be introduced as another independent source of engineering analysis while remaining subject to the same requirements for traceability, reproducibility, and critical examination.

2.3 The Engineering Trust Problem

Large language models differ from conventional engineering tools because they contribute interpretative reasoning rather than executing a fully specified deterministic procedure. A compiler reports whether defined syntax and build rules were satisfied. A simulation follows an explicit mathematical model. An AI system may instead generate an architectural argument, implementation proposal, or risk assessment whose quality depends on context, assumptions, and probabilistic inference.

This creates a distinct engineering trust problem. The output may be correct, partially correct, incomplete, or wrong, while each possibility may be expressed with similar fluency and confidence.

The difficulty is therefore not merely detecting calculation error; it is determining how much authority should be assigned to a reasoning process whose internal basis may be only partially observable and whose result may change when the context or formulation changes.

The common response is to seek a more capable model. Improved models can reduce error frequency and increase practical usefulness, but capability does not remove dependence on a single reasoning source. A highly capable model may still accept a false premise, overlook a domain constraint, or produce a coherent explanation for an incorrect conclusion.

From an engineering perspective, this dependency resembles a single point of failure. The risk is not that one model is uniquely unreliable; the risk is that one reasoning path becomes the sole authority for a decision that may affect architecture, security, cost, or operation.

The verification problem must therefore be framed differently. The relevant question is not whether an AI system is intelligent enough to be trusted without review. The question is whether its reasoning can be incorporated into a process that makes assumptions visible, exposes disagreement, requires evidence, and preserves human responsibility for the final decision.

Independent artificial reasoning offers one possible response. Different systems may interpret the same evidence differently, prioritise different risks, or reach the same conclusion through unrelated analytical paths. Agreement can then become a confidence signal, while disagreement can identify unresolved assumptions or missing evidence.

This does not make consensus equivalent to truth. Multiple systems may share training data, misconceptions, or undocumented assumptions. Independent review must therefore complement—not replace—testing, source inspection, domain evidence, and engineering judgement.

The research question emerging from this problem is consequently procedural: can multiple independent artificial reasoning systems be organised so that their convergence, divergence, and evidence collectively strengthen engineering verification beyond the contribution of any single system?

2.4 Evolution of the Experimental Environment

Unlike most AI-related studies, this research did not begin as an investigation into artificial intelligence itself. It emerged organically during the development of a real engineering project. The objective was to improve software architecture, documentation quality, and engineering decision-making rather than to compare AI models.

The first AI system incorporated into the workflow was ChatGPT. Initially, it served primarily as a discussion partner for architecture, brainstorming sessions, documentation, and the evaluation of external engineering articles. Its ability to maintain long-term conceptual discussions proved valuable during the early design stages. However, a significant practical limitation soon became apparent. As an online conversational system, it lacked persistent awareness of the evolving software repository. Maintaining context required repeated manual uploads of documentation and project artifacts, making continuous engineering work increasingly inefficient as the project grew.

This practical limitation motivated the search for repository-aware development environments. The introduction of Codex represented the first major shift toward AI-assisted implementation directly connected to the project workspace. For the first time, implementation, review, and architectural reasoning could occur within the same engineering context.

As development continued, Cursor became the primary implementation platform. Rather than functioning as a single AI assistant, Cursor enabled orchestration of multiple specialized reasoning agents within a unified development environment. Throughout the research different underlying

models were evaluated through Cursor, while the platform itself consistently remained the preferred implementation environment. Because Cursor operated in automatic and, at times, multi-model modes during the documented period, the identities of all underlying reasoning models cannot be reconstructed reliably.

Not every model proved equally suitable for every engineering task. Gemini occasionally produced valuable ideas and alternative viewpoints, particularly during conceptual discussions, but it was ultimately not retained as a primary participant in the verification workflow because its overall contribution was less consistent for the project's engineering objectives.

Economic constraints also influenced the composition of the experimental environment. Claude Code demonstrated promising engineering capabilities, particularly during implementation tasks. However, during the research period its token consumption was significantly higher than competing solutions, making continuous daily use economically impractical. Since this limitation originated from the development environment rather than from engineering quality itself, Claude Code remains a candidate for future reintroduction should this limitation be resolved.

A similar observation applies to MiMo. From a purely economic perspective, MiMo could not be used continuously because identical audit workloads exhausted its available usage significantly faster than comparable sessions executed with Codex. Nevertheless, MiMo remained one of the most valuable participants in the research for a different reason. Its reasoning frequently reflected assumptions and perspectives that differed from the dominant Western engineering approach represented by the other participating models. Rather than treating this divergence as an inconsistency, the research deliberately preserved it as an additional independent viewpoint during verification.

An important observation gradually emerged throughout the project. Individual AI systems did not maintain fixed responsibilities. Their roles evolved according to project requirements and accumulated experience. Codex, for example, served at different stages as an implementation assistant, later as an orchestration coordinator directing tasks among multiple AI systems, and eventually as one of the principal engineering auditors alongside MiMo. ChatGPT likewise evolved from an architectural discussion partner into a research collaborator responsible for conceptual synthesis, methodology development, and critical examination of engineering processes. The specific reasoning model used within Cursor changed over time, while Cursor itself consistently functioned as the implementation environment hosting multiple specialized agents.

Consequently, the experimental environment should not be understood as a static collection of AI models but rather as a dynamically evolving engineering team in which responsibilities continuously adapted to the changing needs of the project. This evolution itself became one of the unexpected findings of the research.

2.5 The Unexpected Observation

The original objective of the engineering project was neither to study artificial intelligence nor to compare language models. The participating AI systems were introduced incrementally as practical engineering tools to support software development, documentation, architectural design, and technical review.

Initially, each newly introduced model was evaluated according to a familiar engineering question:

Can this system perform a particular engineering task more effectively than the previous one?

The expectation was straightforward. Better models would gradually replace weaker ones, eventually leading to a single preferred engineering assistant.

The project, however, evolved in an unexpected direction.

As additional AI systems became part of the engineering workflow, comparisons rarely produced a clear winner. Instead, different systems repeatedly identified different categories of engineering problems while often agreeing on the final conclusion through entirely different reasoning processes.

One model frequently recognised architectural inconsistencies that others overlooked.

Another demonstrated stronger implementation capabilities while missing broader governance implications.

A third questioned assumptions that the others accepted without discussion.

Occasionally, several independent systems reached the same engineering recommendation despite producing substantially different chains of reasoning.

From a benchmarking perspective these observations appeared inconsistent.

From an engineering perspective they suggested something entirely different.

The participating AI systems were not behaving as interchangeable tools.

They were behaving like independent engineering reviewers.

This distinction fundamentally changed the direction of the research.

Rather than asking which model produced the best answer, the investigation began asking why different models disagreed, why they agreed, and what engineering value emerged from comparing their independent conclusions.

An equally important observation followed.

Disagreement between models was rarely a failure of the verification process.

On the contrary, disagreements frequently exposed hidden assumptions, incomplete documentation, ambiguous architectural decisions, or insufficient engineering evidence. In many cases the disagreement itself proved more valuable than the individual answers that produced it.

The engineering objective therefore shifted.

The project no longer attempted to optimise the performance of individual AI systems.

Instead, it explored whether structured comparison between multiple independent reasoning systems could itself become a repeatable engineering verification method.

This transition marked the beginning of the research presented in this paper.

The subject of the investigation was no longer artificial intelligence.

It became the engineering process created by the interaction of multiple independent reasoning participants.

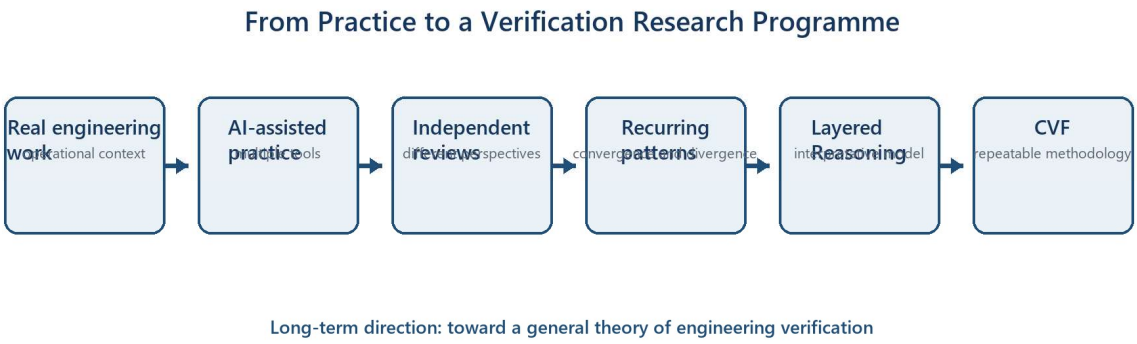

Figure 1. The path from operational engineering practice to an exploratory verification methodology.

Looking back, this shift was never planned.

It emerged naturally from day-to-day engineering practice, where practical problem solving gradually revealed patterns that conventional model benchmarking could neither explain nor measure.

Those patterns ultimately became the foundation of the methodology described in the following chapters.

2.6 From Tool Evaluation to Methodology Research

One of the most significant outcomes of this project was not a technical discovery but a change in research direction.

The engineering work did not begin with the intention of developing a new verification methodology. The initial objective was considerably more practical: to identify which AI systems could most effectively support software engineering activities such as implementation, documentation, architectural analysis, and technical review.

At this stage, individual AI systems were evaluated primarily as engineering tools. Questions focused on implementation quality, reasoning capability, practical usefulness, operating cost, context handling, and overall contribution to the engineering workflow.

This approach gradually became insufficient.

As more engineering decisions were reviewed by multiple independent AI systems, it became increasingly apparent that comparing individual models answered the wrong question.

The objective was never to identify the universally "best" model.

Instead, the practical engineering challenge was to determine how different reasoning processes could be combined to produce more reliable engineering decisions.

This distinction fundamentally changed the nature of the research.

The experimental focus shifted away from model comparison and toward the engineering process created by their interaction.

The participating AI systems became experimental participants rather than experimental subjects.

Their individual performance remained relevant, but no longer represented the primary research outcome.

Instead, the research began examining relationships between independent analyses.

Questions evolved accordingly.

The project no longer asked:

Which model writes better code?

Which model explains architecture more clearly?

Which model performs the strongest audit?

Instead, new engineering questions emerged:

Why do independent analyses disagree?

When multiple models agree, what increases our confidence?

What engineering information is contained within disagreement itself?

Can disagreement become a measurable engineering signal?

Can consensus be treated as engineering evidence rather than opinion?

These questions did not arise from theoretical speculation.

They emerged directly from practical engineering work performed over an extended period within an active software development project.

This evolution also changed the role of the engineer.

Rather than selecting the "correct" AI, the engineer increasingly became responsible for designing the verification process itself: selecting independent reviewers, interpreting disagreements, evaluating evidence, and determining when sufficient confidence had been achieved to make an engineering decision.

Viewed from this perspective, artificial intelligence did not replace engineering judgement.

It increased the importance of engineering judgement.

The responsibility shifted from producing answers toward designing trustworthy verification processes.

This transition marks the conceptual boundary between the background presented in this chapter and the research methodology introduced in the next.

From this point onward, the paper no longer investigates AI systems as engineering tools.

It investigates a new engineering verification methodology that emerged through their practical application.

An important characteristic of this investigation is that the research methodology itself was not predefined. The engineering work preceded the research questions. Only after recurring verification patterns had been observed repeatedly during practical software development did the need for a formal, reproducible verification framework become apparent. In this sense, the methodology documented in this paper is itself the result of the investigation rather than its starting assumption.

Chapter 3 — Research Methodology

3.1 Research Philosophy

The objective of this research was not to determine which artificial intelligence model performed best under controlled benchmark conditions. Instead, the investigation focused on a practical engineering question:

> *Can multiple independent reasoning systems increase confidence in engineering decisions through structured verification?*

Accordingly, the work is presented as a reflective engineering case study and methodology proposal rather than as a controlled model-comparison experiment.

Engineering confidence. In this paper, engineering confidence means the degree to which a decision is supported by independently examined assumptions, traceable evidence, resolved or explicitly documented disagreement, and verification proportionate to the consequence of error. It is not equivalent to subjective certainty, model agreement, or proof of correctness.

Study classification. The observations were reconstructed from practical engineering activity conducted in an evolving project environment. The account is exploratory and interpretative: it aims to make a recurring practice explicit and inspectable, not to establish population-level causal claims.

The software project itself remained the primary engineering activity throughout the entire observation period. Artificial intelligence was never introduced as the subject of the project but as an engineering participant assisting implementation, architectural review, documentation, and technical analysis.

This distinction significantly influenced the experimental design.

Instead of constructing artificial benchmark tasks, all observations originated from real engineering work performed within an active software development project. Every architectural discussion, implementation decision, documentation review, security consideration, and engineering audit represented genuine project activities whose outcomes directly influenced the evolving software system.

Consequently, the research environment prioritised ecological validity over laboratory control. The participating AI systems were observed while solving authentic engineering problems under realistic constraints including incomplete information, changing project requirements, limited computational budgets, evolving documentation, and continuously changing software architecture.

The study therefore resembles an engineering field investigation rather than a controlled laboratory experiment.

This choice inevitably reduces experimental control while substantially increasing practical relevance.

The objective was not to demonstrate ideal behaviour under simplified conditions but to observe how independent reasoning systems behave within the complexity and uncertainty characteristic of real engineering practice.

Throughout the project, no attempt was made to force participating AI systems toward consensus.

Each system was encouraged to analyse engineering problems independently using its own reasoning process.

Agreement between models was therefore considered an observation rather than a design objective.

Similarly, disagreement was never interpreted as experimental failure.

Instead, disagreement represented valuable engineering data requiring further investigation.

This principle became one of the defining characteristics of the methodology.

The engineer remained responsible for evaluating evidence, interpreting disagreements, and making final engineering decisions.

Artificial intelligence contributed reasoning.

Engineering responsibility remained entirely human.

This separation of responsibilities was deliberately maintained throughout the research and constitutes one of the fundamental principles of the proposed methodology.

Rather than replacing engineering judgement, the methodology seeks to strengthen it by introducing additional independent sources of reasoning into an already established engineering verification process.

3.2 Field Study Design

The experimental environment was not established as a fixed laboratory configuration. Instead, it evolved incrementally alongside the engineering project itself. As new AI systems became available or existing systems changed in capability, they were incorporated into the workflow according to practical engineering needs rather than predefined research protocols.

This evolutionary approach reflects the reality of contemporary AI-assisted engineering, where both software projects and AI technologies develop continuously. Rather than freezing the experimental environment, the study recorded how methodological principles remained stable despite changes in the participating systems.

The unit of observation throughout the research was not the AI model itself.

Instead, the unit of observation was the engineering decision.

Each observation began with a genuine engineering problem arising during software development, architectural design, documentation, security analysis, or project governance. The same engineering problem could then be analysed independently by one or more AI systems.

Where practical, participating systems received the problem without knowledge of each other's conclusions. This independence was considered essential to minimise unintentional convergence caused by shared context rather than genuine agreement.

Each independent response was treated as an engineering review rather than as a final answer.

The engineer then compared the resulting analyses using a structured verification process.

The comparison considered multiple dimensions, including:

- technical correctness,
- architectural consistency,
- completeness,
- identification of assumptions,
- recognition of potential risks,
- consistency with existing project documentation,
- and practical applicability.

Importantly, disagreement between systems was not interpreted as evidence that one participant was necessarily incorrect.

Instead, disagreement initiated an additional verification cycle.

This frequently involved reviewing documentation, consulting source material, refining the engineering question, requesting clarification, or introducing additional independent reviewers.

Consequently, verification became an iterative engineering process rather than a single evaluation event.

The engineer remained outside the reasoning process itself.

Rather than selecting a preferred AI response, the engineer evaluated the quality of evidence produced by each independent analysis.

Final engineering decisions were therefore based on accumulated evidence rather than model preference.

Throughout the study, AI systems were free to assume different functional roles depending on the engineering task.

A participant might primarily contribute implementation expertise during one activity while acting as an architectural reviewer, documentation auditor, or methodological critic during another.

Roles were therefore considered dynamic properties of the engineering process rather than permanent characteristics of individual AI systems.

This flexibility reflects established engineering practice, where human specialists routinely assume different responsibilities depending on project requirements.

The experimental design therefore investigated not only independent reasoning, but also the interaction between dynamically changing reasoning roles within a structured engineering verification framework.

3.3 Participants and Dynamic Roles

Unlike conventional comparative studies, the participating artificial intelligence systems were not assigned fixed experimental roles throughout the investigation. Their responsibilities evolved according to the engineering task under consideration and the practical strengths demonstrated during previous verification cycles.

This decision reflected established engineering practice, where experienced professionals routinely assume different responsibilities depending on the nature of the work rather than holding permanently assigned functions.

Consequently, the study did not classify participants according to model identity alone.

Instead, each participant was evaluated according to the role it fulfilled within a specific engineering verification cycle.

Typical roles included, but were not limited to:

- implementation support,
- architectural analysis,
- technical documentation,
- security review,
- governance evaluation,
- consistency auditing,
- methodological criticism,

- and independent engineering review.

The same participant could perform different roles across successive verification cycles.

Likewise, multiple participants could simultaneously fulfil the same role, allowing independent analyses of identical engineering problems.

Role allocation therefore emerged from demonstrated capability rather than predetermined hierarchy.

An equally important characteristic of the methodology was the absence of a permanently designated "primary AI."

No participant possessed privileged authority over engineering decisions.

Every analysis constituted an independent contribution requiring subsequent verification.

Engineering responsibility remained exclusively with the human investigator, who designed the verification process, evaluated the collected evidence, and determined whether sufficient confidence had been established to support an engineering decision.

This distinction proved essential.

The objective was never to identify an AI capable of replacing engineering judgement.

Instead, the participating systems collectively formed a distributed reasoning environment whose value depended upon diversity, independence, and critical interaction rather than individual superiority.

As the project progressed, certain participants consistently demonstrated particular strengths within specific engineering domains.

Some repeatedly identified architectural inconsistencies.

Others excelled in implementation detail, documentation analysis, or methodological critique.

Rather than treating these observations as evidence of overall superiority, they were interpreted as indications of complementary expertise.

This interpretation gradually influenced the experimental methodology.

Instead of attempting to maximise the performance of individual participants, the verification process increasingly sought to combine complementary reasoning capabilities while preserving independence during the initial stages of analysis.

Accordingly, the effective unit of investigation became neither the individual participant nor the engineering problem in isolation.

It became the interaction between independently generated engineering perspectives within a structured verification framework.

3.4 Data Collection and Verification Cycles

The research did not treat individual AI responses as isolated observations.

Instead, the fundamental unit of data collection was the engineering verification cycle.

Each cycle originated from a genuine engineering problem encountered during the development of the software platform. These problems included software implementation, architectural design, documentation, governance, security, workflow optimisation, and methodological questions.

A verification cycle typically consisted of several consecutive stages.

First, the engineering problem was formulated by the human investigator.

The same problem was then analysed independently by one or more participating reasoning systems.

Each participant produced its own engineering analysis without assuming that alternative analyses would necessarily exist.

At this stage, every response was considered an independent engineering hypothesis rather than a verified conclusion.

Following the initial analyses, the engineer compared the independent outputs.

Rather than selecting the preferred response, the comparison focused on identifying:

- areas of agreement,
- conflicting conclusions,
- implicit assumptions,
- missing evidence,
- overlooked constraints,
- complementary observations,
- and unanswered questions.

Whenever significant discrepancies appeared, the verification process did not terminate.

Instead, additional verification cycles were initiated.

These could include clarification of the engineering problem, consultation of project documentation, inspection of implementation details, introduction of further independent reviewers, or iterative refinement of the original question.

Consequently, engineering confidence emerged progressively through repeated verification rather than through acceptance of a single response.

Every completed verification cycle therefore produced considerably more information than the individual analyses from which it originated.

Besides technical conclusions, each cycle generated observations regarding reasoning quality, evidence usage, consistency, uncertainty, and interaction between independently developed engineering perspectives.

This distinction proved increasingly important as the research progressed.

The collected body of project material gradually shifted from recording engineering answers toward documenting engineering reasoning.

Accordingly, the primary research material became not only what conclusions were reached, but also how independent reasoning processes converged, diverged, or evolved through repeated verification.

The resulting collection of observed verification cycles formed the practical and observational basis for the subsequent methodological interpretation.

3.5 Recursive Cross-Review Protocol

One of the most significant methodological developments emerged unexpectedly during the course of the investigation.

Initially, independent analyses produced by participating reasoning systems were evaluated exclusively by the human investigator. The objective was to compare engineering conclusions originating from independent reasoning processes while preserving their initial independence.

As the research progressed, an additional verification step was introduced.

Following the completion of an independent verification cycle, the analyses generated by each participant were redistributed among the remaining participants. Each reasoning system was then asked to evaluate not only the original engineering problem but also the independent analyses produced by the others.

This modification fundamentally changed the nature of the verification process.

The object of analysis was no longer solely the engineering problem itself.

Instead, the reasoning processes became explicit subjects of investigation.

During the earliest iterations, these reciprocal evaluations frequently exhibited competitive characteristics. Participants often defended their own conclusions while emphasising perceived weaknesses in alternative analyses.

This behaviour was neither expected nor intentionally encouraged.

It emerged naturally from the evaluation process and represented an unplanned observation rather than an experimental objective.

However, after several recursive review cycles, a qualitative change became apparent.

The focus gradually shifted away from defending individual conclusions toward objectively analysing differences in reasoning, evidence, assumptions, and methodological approaches.

Participants increasingly acknowledged both the strengths and limitations of their own analyses while identifying complementary insights provided by others.

This transition marked a significant methodological milestone.

The recursive review process no longer generated only corrections or improved engineering answers.

It began revealing relationships between independent reasoning processes.

Repeated cross-review exposed recurring patterns that were not apparent within any single analysis.

These higher-order observations frequently concerned:

- complementary reasoning strategies,
- shared implicit assumptions,
- systematic blind spots,
- recurring verification paths,
- and alternative conceptual models leading to the same engineering conclusion.

Perhaps the most important observation was that independent participants often converged toward remarkably similar meta-level interpretations after analysing one another's reasoning.

While their original engineering analyses remained distinct, their evaluation of the collective reasoning process became increasingly consistent.

The convergence therefore occurred not primarily at the level of engineering answers, but at the level of reasoning analysis itself.

This observation suggested that higher-order engineering knowledge could emerge through recursive comparison of independent reasoning processes rather than from improvements in any individual participant.

Consequently, the recursive cross-review protocol evolved from a verification technique into a methodological instrument for discovering emergent engineering patterns.

These observations ultimately motivated the development of the layered reasoning concepts described in the subsequent chapters.

3.6 Emergence of Layered Reasoning

The recursive cross-review protocol produced an observation that had not been anticipated at the beginning of the investigation.

Initially, each verification cycle was expected to improve the quality of individual engineering analyses through independent review and correction.

Instead, repeated comparison revealed a different phenomenon.

As participants analysed not only engineering problems but also one another's reasoning processes, increasingly consistent patterns began to emerge that were not explicitly present within any individual analysis.

These observations suggested that engineering knowledge was developing across multiple conceptual layers.

The first layer consisted of direct engineering reasoning.

At this level, participants analysed the engineering problem itself and proposed technical conclusions supported by available evidence.

The second layer emerged when participants evaluated alternative reasoning processes.

Attention shifted from solving the engineering problem toward understanding why different analyses reached different conclusions.

Assumptions, evidence selection, verification strategy, and reasoning structure became explicit objects of investigation.

A third layer gradually appeared through repeated recursive comparison.

Instead of analysing individual reasoning processes in isolation, participants began recognising recurring relationships between multiple reasoning strategies.

Patterns became identifiable across successive verification cycles.

These patterns frequently explained not only the engineering problem under consideration but also characteristic strengths, limitations, and interactions between different reasoning approaches.

Importantly, these higher-order observations rarely originated from a single participant.

Rather, they emerged from the structured interaction between multiple independent reasoning processes over successive verification cycles.

This distinction proved fundamental.

The layered observations were properties of the verification system as a whole rather than capabilities attributable to any individual participant.

Consequently, the concept of Layered Reasoning was introduced as a descriptive framework for understanding how engineering knowledge progressively evolved through recursive verification.

Within this framework, each successive layer analysed the outputs generated by the preceding layer while preserving the independence of the original reasoning processes.

The framework therefore describes an emergent property of the verification methodology rather than a characteristic of any particular artificial intelligence model.

This observation significantly altered the direction of the research.

The objective was no longer limited to improving engineering answers.

Instead, the investigation sought to understand how successive layers of independent reasoning could produce engineering knowledge that was inaccessible through isolated analysis alone.

Layered Reasoning therefore became not simply an observation, but the conceptual foundation upon which the subsequent Consensus Verification Framework was later constructed.

Chapter 4 — Results

4.1 Independent Reasoning

The first and most consistent observation throughout the investigation concerned the behaviour of independent reasoning processes.

When identical engineering problems were presented independently to different participating AI systems, the resulting analyses were rarely identical.

Differences appeared across multiple dimensions, including problem decomposition, engineering priorities, supporting arguments, proposed implementation strategies, identification of risks, and interpretation of available documentation.

These differences were observed consistently throughout the investigation and were independent of the engineering domain under consideration.

Importantly, variation between independent analyses did not imply variation in engineering quality.

Different participants frequently reached equivalent engineering conclusions while employing substantially different reasoning paths.

Conversely, apparently similar reasoning strategies occasionally resulted in different engineering recommendations.

This distinction proved significant.

The investigation therefore recorded reasoning diversity as an observable characteristic rather than treating it as experimental noise.

Repeated observations demonstrated that independent reasoning processes naturally generated complementary perspectives on the same engineering problem.

Some analyses focused primarily on implementation details.

Others prioritised architectural consistency, governance implications, operational maintainability, security considerations, or methodological correctness.

No single participant consistently produced the most comprehensive analysis across all engineering activities.

Instead, different participants repeatedly demonstrated complementary strengths depending upon the nature of the engineering task.

These observations remained stable despite changes in participating systems throughout the investigation.

While individual capabilities evolved over time, the existence of independent reasoning diversity remained consistently observable.

Accordingly, the study interpreted reasoning diversity as a recurring property within the documented project context rather than as a temporary characteristic of particular AI models.

At this stage of the investigation, no attempt was made to determine which participant produced the "best" engineering answer.

Instead, the primary result consisted of the repeated observation that multiple independent reasoning processes generated engineering information unavailable from any single analysis alone.

This observation provided the initial practical basis for examining convergence, disagreement, and higher-order reasoning patterns.

4.2 Convergence of Independent Analyses

Following the collection of independent engineering analyses, recurring patterns of convergence became observable across successive verification cycles.

Although participating reasoning systems frequently differed in terminology, structure, emphasis, and intermediate arguments, they often reached remarkably similar engineering conclusions.

This convergence rarely occurred through identical reasoning.

Instead, different participants frequently approached the same engineering problem from distinct conceptual perspectives before arriving at compatible recommendations.

The convergence therefore appeared primarily at the level of engineering judgement rather than at the level of reasoning structure.

This distinction remained consistent throughout the investigation.

Independent analyses often highlighted different risks, prioritised different implementation strategies, or organised their arguments differently while nevertheless identifying the same underlying engineering issue.

Accordingly, agreement was not interpreted as evidence of identical reasoning.

Instead, agreement indicated that multiple independent reasoning processes had reached compatible engineering conclusions despite following different analytical paths.

This observation significantly increased engineering confidence.

When several independently developed analyses identified the same architectural weakness, implementation defect, governance concern, or documentation inconsistency without prior exposure to one another's conclusions, confidence in the existence of the underlying issue increased substantially.

Importantly, convergence was not universal.

Some engineering questions produced immediate agreement.

Others generated partial convergence combined with significant disagreement regarding implementation details or engineering priorities.

This variation proved valuable because it revealed that convergence itself was dependent upon the nature of the engineering problem rather than representing a fixed property of the participating systems.

Repeated observations also demonstrated that convergence frequently became stronger after additional engineering evidence had been introduced.

Clarification of requirements, review of implementation details, consultation of documentation, or refinement of the engineering question often reduced disagreement without requiring participants to abandon their independent reasoning processes.

Consequently, convergence was observed as a dynamic characteristic of the verification process rather than as a predefined objective.

The investigation therefore identified convergence as an observable engineering phenomenon emerging from repeated independent analyses rather than from coordinated agreement between participants.

This observation provided the practical basis for examining the complementary role of disagreement within the overall verification methodology.

4.3 Divergence as a Source of Engineering Evidence

Not all independent analyses converged toward similar engineering conclusions.

Numerous verification cycles produced significant differences in interpretation, implementation strategy, architectural evaluation, or assessment of engineering risk.

Rather than treating these divergences as failures of the verification process, the investigation documented their practical consequences within the engineering workflow.

One consistent observation emerged.

When substantial disagreement occurred, the engineering investigation rarely concluded immediately.

Instead, divergence initiated additional verification activities.

These frequently included re-examination of project documentation, inspection of implementation details, clarification of engineering requirements, reformulation of the original question, or further independent analysis.

In practice, disagreement functioned as an indicator that the available engineering evidence was insufficient to support a confident decision.

The disagreement itself therefore became part of the engineering evidence.

Importantly, subsequent verification frequently demonstrated that no individual participant had been entirely correct or entirely incorrect.

Instead, different analyses often captured different aspects of the underlying engineering problem.

One participant identified an implementation constraint.

Another recognised an architectural implication.

A third highlighted governance or operational considerations.

Viewed independently, none of these analyses completely described the engineering situation.

Viewed collectively, however, they provided a substantially richer representation of the problem than any single analysis alone.

Repeated observations further demonstrated that disagreement frequently exposed hidden assumptions.

Engineering discussions that initially appeared to concern conflicting conclusions often proved instead to reflect different interpretations of incomplete requirements, undocumented constraints, or unstated design objectives.

Once these assumptions became explicit, many apparent disagreements were either resolved or reformulated into more precise engineering questions.

Consequently, divergence frequently improved the quality of the engineering investigation even when it did not immediately improve the engineering answer.

This distinction proved important throughout the study.

The practical value of disagreement did not arise from contradiction itself.

Its value arose from directing attention toward areas where engineering confidence remained insufficient.

Accordingly, the investigation identified divergence not merely as an unavoidable characteristic of independent reasoning, but as an observable mechanism for locating uncertainty within complex engineering problems.

This observation remained consistent across software implementation, architecture, documentation, governance, and methodological discussions.

4.4 Emergence of Higher-Order Patterns

As the number of completed verification cycles increased, a further observation gradually became apparent.

Certain engineering insights could no longer be attributed to any single independent analysis.

Instead, they emerged only after multiple analyses had been compared collectively.

Initially, these observations appeared as isolated coincidences.

Independent participants repeatedly highlighted different aspects of the same engineering problem, yet none explicitly described the broader relationship connecting those observations.

Only when the analyses were examined together did additional patterns become visible.

These higher-order patterns differed fundamentally from ordinary engineering conclusions.

Rather than describing the engineering problem itself, they described relationships between independent reasoning processes.

Examples included recurring complementary reasoning strategies, repeated sequences of verification, characteristic combinations of strengths and limitations, and consistent interactions between different analytical approaches.

Importantly, these observations were not produced directly by any individual participant.

They became visible only through structured comparison of multiple independent analyses collected across repeated verification cycles.

The investigation therefore distinguished between two categories of engineering knowledge.

The first consisted of direct engineering conclusions generated during individual analyses.

The second consisted of emergent observations describing relationships between independently generated engineering conclusions.

This distinction remained consistent throughout the study.

Repeated comparison demonstrated that higher-order patterns frequently explained engineering behaviour that appeared inconsistent when individual analyses were considered separately.

Viewed collectively, however, these apparent inconsistencies often formed coherent and reproducible structures.

Another recurring observation concerned explanatory completeness.

Engineering questions that initially appeared unresolved frequently became substantially clearer after the relationships between multiple independent analyses had been examined.

In several cases, the additional understanding did not originate from new engineering evidence.

Instead, it emerged through reinterpretation of evidence that had already been identified independently by different participants.

Accordingly, the investigation recorded higher-order pattern recognition as a distinct observable phenomenon within the verification process.

At this stage of the research, no theoretical explanation was proposed.

The phenomenon was documented solely as a repeated empirical observation whose interpretation is discussed in the following chapters.

4.5 Stabilisation of Meta-Patterns

As the investigation progressed, individual observations gradually evolved into recurring and recognisable engineering patterns.

Initially, each verification cycle appeared to represent an independent engineering event.

However, after a sufficiently large number of documented verification cycles, similarities between previously unrelated investigations became increasingly apparent.

Certain categories of disagreement repeatedly emerged under similar engineering conditions.

Likewise, specific forms of convergence, complementary reasoning, and recursive cross-review consistently produced comparable verification outcomes despite involving different engineering problems.

This repetition represented an important observation.

The investigation no longer relied upon isolated examples.

Instead, engineering behaviour began exhibiting stable characteristics across multiple independent verification cycles.

Repeated observations demonstrated that these characteristics were reproducible within the documented engineering workflow.

Importantly, stability did not imply identical outcomes.

Individual engineering problems continued to generate unique discussions, different implementation proposals, and varying analytical perspectives.

What became increasingly stable was not the engineering answer itself, but the structure of the verification process through which confidence was established.

The investigation therefore distinguished between variability of engineering content and stability of engineering behaviour.

Engineering conclusions remained problem-specific.

The verification dynamics became progressively predictable.

Several recurring characteristics were documented repeatedly.

Independent analyses consistently expanded the available engineering perspective.

Convergence consistently increased engineering confidence.

Divergence consistently initiated additional evidence collection.

Recursive cross-review consistently revealed relationships that were not visible during isolated analysis.

These observations remained stable despite changes in participating AI systems, engineering topics, project maturity, and available documentation.

Consequently, the research concluded that the observed verification behaviour could no longer be explained solely by characteristics of individual participants.

The stability appeared to arise from the structure of the verification process itself.

This marked the final empirical observation recorded during the investigation.

The interpretation of these recurring patterns and their implications for engineering methodology are presented in the following discussion.

Chapter 5 — Representative Verification Cases

This chapter reconstructs representative verification cycles from retained project records. The purpose is illustrative rather than statistical: the cases show how independent review, disagreement, evidence inspection, remediation, and re-verification operated in practice. They do not establish comparative model performance or a universal success rate.

Source-language note. Most primary project records were written in Hungarian, with some audit artefacts in English. The descriptions below are analytical English summaries prepared for this paper; they are not presented as verbatim transcript translations. Original timestamps, verdict labels, file identifiers, and technical distinctions have been preserved where material.

5.1 Security Remediation and the Split-Brain Repository

Field	Recorded value
Period	22-26 May 2026
Source completeness	Audit reports, structured findings, three time-stamped verification runs, and retained review conversation
Reasoning roles	Independent security audit; implementation and consolidation; subsequent verification
Initial state	CRITICAL / UNSAFE FOR PRODUCTION
Final recorded state	SECURE / PASS after consolidation and re-verification

The first audit identified critical defects in device binding, team-scope propagation, revocation, and trust in client-provided identifiers. A later remediation report claimed that several findings had been corrected. The 19:15 verification found partial improvement but retained four material failures.

At 19:45, a further verification cycle produced a different explanation: several reported fixes existed only in duplicate machine-suffixed files and were absent from the canonical execution path. The disagreement between the remediation narrative and the inspected code was therefore not treated as a matter of model preference. It became a new engineering finding: the repository itself was in a split-brain state.

After the duplicate implementations were consolidated into the canonical path, the 20:30 verification re-examined the same finding matrix and recorded a PASS. The decisive evidence was not consensus alone, but the traceable change in canonical code and the subsequent status transition of each finding.

Methodological significance.

- Divergence localized a hidden configuration-management failure that a summary-level review could have missed.
- A changed verdict was legitimate because the underlying evidence changed and the reasons were recorded.
- The case supports iterative verification; it does not prove that every independent audit is correct.

5.2 Layered Review of an Encryption Data Path

Field	Recorded value
Period	27 May 2026 and subsequent implementation cycles
Source completeness	Retained multi-participant discussion, design documents, gap lists, implementation summaries, code-focused reviews, and remediation checks
Reasoning roles	Conceptual architecture, primary audit, implementation analysis, and data-path-focused independent review
Initial question	How can a creator-operated platform protect business data while remaining maintainable?
Outcome	Progressive gap discovery followed by narrower implementation and remediation phases

The initial discussion concentrated on key ownership, service authority, and the limits of creator access. Independent reviews then followed different representations of the same data. One line of reasoning examined the formal key model; another examined implementation feasibility; a third traced copies and derivatives across PostgreSQL JSONB mirrors, PocketBase payloads, Loki records, audit rows, projection failures, local caches, exports, and snapshots.

This produced a layered sequence. The first layer asked whether the proposed encryption model was conceptually coherent. The second reviewed the reasoning and discovered that protecting only final domain columns would leave plaintext replicas elsewhere. The third converted the recurring observation into a broader rule: the security boundary must follow the entire data path, including diagnostic and historical copies.

Implementation review then exposed further operational consequences, including payload-encoding compatibility, migration requirements, audit sanitization, and the distinction between an encrypted mirror and a still-plaintext projected destination. Remediation was deliberately divided into small phases and rechecked against the earlier gap register.

Methodological significance.

- Different reasoning perspectives were complementary because they followed different artefacts and failure surfaces.
- Repeated reconsideration added information rather than merely restating the original premise.
- The case motivates Layered Reasoning and the provisional hypothesis of productive context drift.

5.3 Independent Review of a Workflow State Machine

Field	Recorded value
Date	19 July 2026
Source completeness	Decision documents, code, UI routes, migration and smoke artefacts, two audit reports, retained audit conversation, and same-day remediation record
Reasoning roles	Primary docs-to-code audit; independent defect-first reaudit; human-directed correction
Initial verdict	ALIGNED across all declared gates
Corrected verdict	PARTIAL because one hard gate contradicted the implementation; ALIGNED after correction

A primary dual audit reported that the default-team reference workflow was aligned across documentation and code. The independent review agreed with most of the analysis but tested the declared state-machine invariant separately. It found that the implementation permitted direct submitted-to-accepted and submitted-to-rejected transitions, while the normative gate required an intermediate under-review state.

The finding was narrow but decisive. Because one hard gate was contradicted, the overall result could not remain ALIGNED even though the remaining gates were satisfied. The transition logic was corrected, the audit was repeated, and the final report retained an addendum describing both the contradiction and its closure.

Methodological significance.

- High aggregate agreement did not erase a single material contradiction.
- The independent reviewer added value by isolating one invariant rather than reproducing the whole first audit.
- The retained correction history makes the final consensus more credible than an unqualified initial PASS.

5.4 Compact Counterexample: A Premature Zero-Finding Conclusion

Field	Recorded value
Date	23 July 2026
Source completeness	Previous audit, independent reaudit, targeted recheck, test result, and retained conversation
Subject	Canonical team UUID versus legacy slug in live UI and contract paths
Prior conclusion	No live MUST_FIX findings remained
Independent result	Ten live files remained; after remediation, targeted recheck recorded zero and 9/9 tests passed

This case is included as a caution against treating a previous zero-finding result as evidence of completeness. A targeted independent search found eight live user-interface files and two active contract or wire documents that still transmitted or illustrated a legacy team slug where a canonical UUID was required.

After correction, the same explicit target set was rechecked and the result changed to zero remaining live findings. The evidential value came from the reproducible search boundary and before-and-after record, not from the confidence of either auditor.

Methodological significance.

- Negative findings require a declared search boundary and are especially vulnerable to incomplete coverage.
- A reaudit can legitimately overturn consensus without implying that the earlier work had no value.

5.5 Cross-Case Interpretation

Across the four cases, the strongest contribution of multiple reasoning participants was not majority voting. It was the creation of inspectable transitions between claim, contradiction, evidence, correction, and re-verification. Agreement became useful only when its evidential basis was visible; disagreement became useful when it identified a testable difference.

The cases also distinguish productive reconsideration from unproductive repetition. A repeated premise was productive when the subsequent analysis inspected a different artefact, assumption, lifecycle stage, or authority boundary and therefore changed the engineering evidence. Repetition without a new inspection target, new evidence, or a changed decision state would not satisfy this criterion.

These records support the plausibility and practical usefulness of the proposed methodology within this project. They do not establish model independence, general effectiveness, or superiority over established review methods. Those questions require a prospective study with predefined tasks, preserved prompts, controlled information boundaries, and explicit outcome measures.

5.6 Evidence Availability and Audit Trail

The case reconstructions draw on locally retained project artefacts, including versioned Markdown reports, structured JSON findings, code and migration references, test outcomes, and archived AI conversation logs. These records contain project-specific and potentially sensitive operational detail and are therefore not published with this draft. A future research release should provide a redacted case corpus, a provenance manifest, and stable identifiers sufficient for independent inspection without exposing credentials, personal information, or protected implementation detail.

Chapter 6 — Discussion

6.1 From Observations to Interpretation

The preceding chapters documented the principal practical observations reconstructed from the engineering investigation.

These observations included the existence of independent reasoning processes, recurring convergence and divergence between analyses, the emergence of higher-order relationships through recursive cross-review, and the gradual stabilisation of verification behaviour across repeated engineering cycles.

At this point, the focus of the paper changes.

The Results chapter intentionally limited itself to documenting observable behaviour without proposing theoretical explanations or methodological conclusions.

The purpose of this discussion is different.

Here, the observed phenomena are interpreted within the broader context of engineering verification and AI-assisted software development.

The distinction between observation and interpretation is intentional.

Throughout the investigation, considerable effort was made to avoid drawing conclusions before sufficient empirical evidence had accumulated.

Only after recurring patterns had been observed across multiple verification cycles did it become appropriate to consider provisional explanations for those observations.

Accordingly, the interpretations presented in this chapter should be understood as engineering hypotheses grounded in reflective practice rather than as independently verified scientific facts.

Their purpose is to explain the recurring behaviours described in the Results chapter and to provide a conceptual framework capable of supporting future investigation.

One recurring observation proved particularly significant.

Engineering confidence appeared to increase not because individual analyses became more accurate, but because multiple independent reasoning processes repeatedly interacted through structured verification.

Likewise, disagreement consistently acted as a mechanism for identifying uncertainty rather than simply indicating analytical failure.

Furthermore, recursive comparison repeatedly revealed relationships that were not visible during isolated analysis.

Taken individually, each of these observations appeared limited in scope.

Considered together, however, they suggested that the verification process itself possessed characteristics extending beyond the capabilities of any individual participant.

The following sections examine this interpretation in greater detail.

To facilitate discussion, the observed phenomenon is described using the conceptual terminology introduced in this paper.

These terms do not represent established scientific classifications.

They are proposed solely as descriptive labels for recurring behaviours documented during the investigation and are intended to support precise discussion rather than to assert the existence of fundamentally new principles.

The first of these concepts is referred to as Layered Reasoning, which provides an interpretative model for understanding how higher-order engineering insights emerged from repeated interaction between independent reasoning processes.

6.2 Layered Reasoning

The observations presented in the Results chapter suggest that the engineering verification process cannot be fully explained by considering individual reasoning processes in isolation.

Throughout the investigation, engineering understanding evolved through repeated interaction between multiple independent analyses.

This evolution did not occur because individual participants continuously improved their own reasoning.

Instead, it emerged from the structured comparison of different reasoning processes across successive verification cycles.

To describe this observed phenomenon, this paper introduces the term Layered Reasoning.

The term is not intended to describe a new reasoning algorithm or a specific property of artificial intelligence.

Rather, it provides a conceptual description of how engineering understanding progressively developed during the documented verification process.

At the first layer, independent participants analysed the engineering problem directly.

Each analysis reflected its own assumptions, priorities, decomposition strategy, and interpretation of the available evidence.

This layer corresponds to conventional engineering reasoning and can be performed by either human engineers or AI-assisted systems.

The second layer emerged when these independent analyses became the subject of further examination.

Attention shifted from evaluating the engineering problem itself to evaluating how different participants had reasoned about that problem.

Differences in assumptions, missing constraints, complementary evidence, and alternative interpretations became explicit.

At this stage, the reasoning process itself became an engineering artefact subject to verification.

A third layer gradually appeared during repeated recursive cross-review.

Rather than comparing isolated arguments, the investigation began revealing recurring relationships between reasoning strategies themselves.

Certain analytical approaches consistently complemented one another.

Some combinations repeatedly exposed hidden assumptions.

Others consistently strengthened engineering confidence through independent convergence.

These recurring relationships were not attributable to any individual participant.

They emerged only after multiple verification cycles had accumulated sufficient comparative evidence.

An important characteristic of Layered Reasoning is that each successive layer operates on the outputs of the previous one rather than replacing it.

Engineering reasoning remains essential throughout the process.

However, the object of analysis progressively shifts.

The investigation moves from analysing engineering problems, to analysing reasoning about those problems, and finally to analysing recurring structures within the verification process itself.

This interpretation explains several observations documented in the Results chapter.

It explains why disagreement frequently generated valuable engineering evidence.

It explains why recursive cross-review repeatedly produced insights that were absent from individual analyses.

It also explains why stable verification behaviour gradually emerged despite substantial variation in engineering problems, participating AI systems, and implementation contexts.

Layered Reasoning should therefore be understood as an interpretative model describing the organisation of the verification process rather than the behaviour of any individual participant.

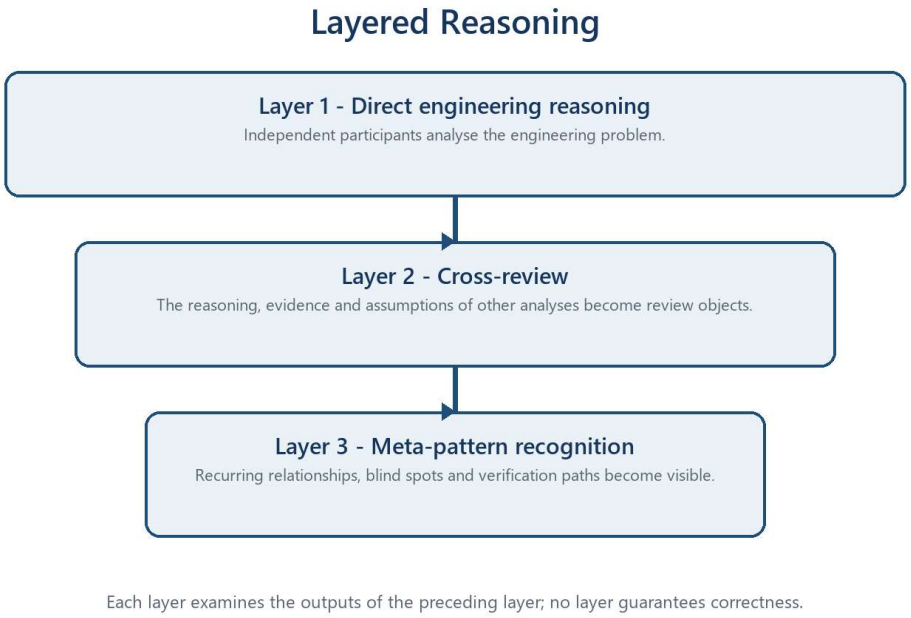

Figure 2. Layered Reasoning as successive analysis of problems, reasoning processes, and recurring relationships.

It provides a conceptual framework for understanding how engineering confidence can emerge from structured interaction between multiple independent reasoning processes without assuming that any single participant possesseses complete understanding of the engineering problem.

Within the context of this investigation, Layered Reasoning represents the first conceptual component of a broader engineering verification methodology.

The second component, discussed in the following section, concerns how these layered interactions can be organised into a repeatable verification framework suitable for practical engineering use.

6.3 Consensus Verification Framework

The observations presented throughout this investigation suggest that engineering confidence did not arise from individual reasoning performance alone.

Instead, confidence emerged from a structured verification process in which multiple independent reasoning processes interacted through repeated comparison, evidence evaluation, and recursive review.

Based on these observations, this paper proposes the Consensus Verification Framework (CVF) as an engineering methodology for organising AI-assisted verification.

The framework is not intended to identify the "best" reasoning system.

Nor does it assume that consensus necessarily represents objective correctness.

Instead, its primary objective is to increase engineering confidence by systematically exposing agreement, disagreement, uncertainty, and complementary evidence throughout the verification process.

Within the proposed framework, independent analyses constitute the initial stage of verification.

Participants examine the same engineering problem without prior exposure to one another's conclusions.

This independence reduces the likelihood that later agreement results merely from confirmation or adaptation to an existing solution.

The resulting analyses are then compared in a structured manner.

Areas of convergence increase confidence that independently developed reasoning processes have identified compatible engineering conclusions.

Areas of divergence indicate that additional investigation may be required before a reliable engineering decision can be reached.

Rather than treating either outcome as inherently desirable, the framework interprets both as informative signals within the verification process.

Recursive cross-review forms the next stage of the framework.

Participants evaluate not only the engineering problem but also the reasoning presented by other participants.

This process enables assumptions to become explicit, complementary evidence to emerge, and hidden relationships between independent analyses to be identified.

As documented in the Results chapter, repeated application of recursive cross-review may reveal higher-order patterns that are not visible during isolated analysis.

The framework therefore treats verification as an iterative process rather than a single evaluation event.

Each verification cycle may generate additional evidence, reformulate the engineering question, clarify requirements, or modify confidence in previously accepted conclusions.

The objective is not necessarily to eliminate disagreement, but to determine whether sufficient engineering evidence has been accumulated to support a responsible engineering decision.

An important characteristic of the Consensus Verification Framework is its independence from particular AI models, software platforms, or implementation technologies.

The methodology depends upon the structure of verification rather than upon the capabilities of any individual participant.

Human engineers, different AI systems, domain specialists, or combinations of these may participate within the same verification process, provided that independent reasoning is preserved during the initial stages of analysis.

The framework likewise does not require complete consensus.

Engineering decisions may legitimately be reached despite persistent disagreement, provided that the disagreement has been examined, documented, and understood.

In this respect, disagreement represents residual uncertainty rather than methodological failure.

Accordingly, the Consensus Verification Framework should be understood as an engineering verification methodology designed to organise independent reasoning into a repeatable process for improving engineering confidence.

It does not claim to guarantee correctness.

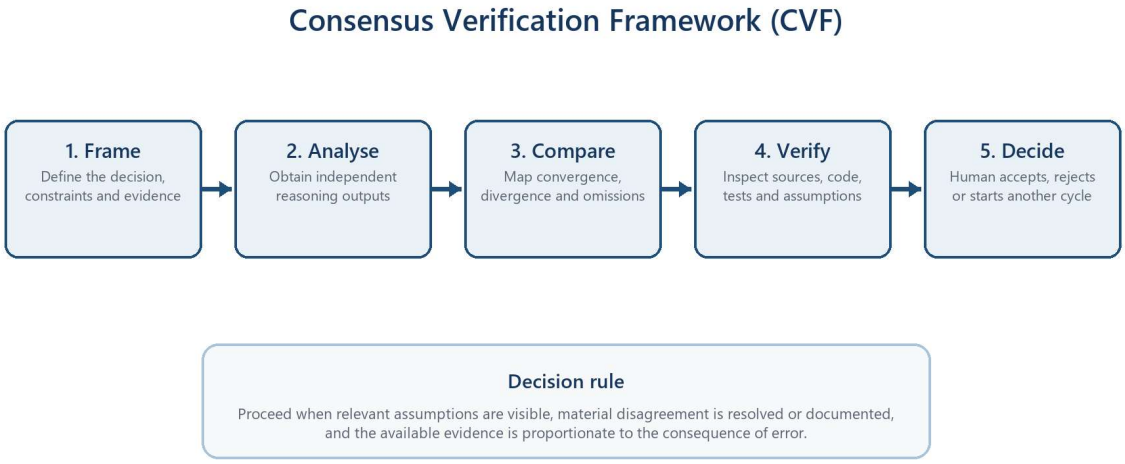

Figure 3. The Consensus Verification Framework as an iterative, human-governed verification cycle.

Instead, it provides a systematic approach for reducing the probability that important evidence, assumptions, or alternative interpretations remain undiscovered before engineering decisions are made.

6.4 Engineering Implications

The observations and interpretations presented in this investigation suggest that the primary contribution of AI-assisted verification may not lie in improving the reasoning capability of individual systems.

Instead, the principal engineering benefit appears to arise from organising independent reasoning into a structured verification process.

This distinction has practical implications for engineering practice.

Traditional software verification often relies upon independent code review, architectural review, testing, and domain expertise to reduce engineering risk.

The Consensus Verification Framework extends this established engineering philosophy by incorporating multiple independent reasoning processes into the verification workflow while preserving the fundamental principle that engineering responsibility remains with human decision-makers.

One implication concerns software architecture.

The investigation indicates that engineering verification should be considered an independent architectural capability rather than an activity performed only after implementation.

Verification can instead become a continuous engineering process operating throughout requirements analysis, system design, implementation, documentation, security assessment, and operational planning.

A second implication concerns AI integration.

Current discussions frequently evaluate AI systems according to benchmark performance or the capabilities of individual models.

The observations documented in this investigation suggest that equal or greater engineering value may arise from designing effective interaction between reasoning processes rather than from relying exclusively on increasingly capable individual systems.

From this perspective, the engineering challenge shifts from model selection toward verification design.

A further implication concerns engineering governance.

The structured documentation of agreement, disagreement, evidence collection, and verification history creates an auditable record of engineering decision-making.

Such documentation enables later review of not only the final engineering decision but also the reasoning process through which that decision was reached.

This characteristic may prove particularly valuable within safety-critical, regulated, or high-assurance engineering environments where traceability and accountability are essential.

The investigation also has implications for the role of human engineers.

Throughout the documented verification cycles, human participants remained responsible for defining engineering objectives, evaluating evidence, determining when verification was sufficient, and accepting responsibility for final decisions.

The proposed methodology therefore should not be interpreted as replacing engineering judgement.

Instead, it provides additional structured evidence intended to support that judgement.

Finally, the observations suggest a broader methodological implication.

Engineering verification may benefit from evaluating not only the quality of individual analyses but also the quality of interaction between independent analyses.

This represents a shift in emphasis.

Rather than asking whether a particular reasoning system produced the correct answer, engineering verification increasingly asks whether the verification process itself provided sufficient evidence to justify confidence in the engineering decision.

This distinction aligns closely with established engineering principles, where confidence is derived not from unquestioned authority but from independent verification, documented evidence, reproducible processes, and transparent decision-making.

Within this context, the Consensus Verification Framework should be viewed not as an alternative to established engineering practice, but as a structured extension of long-standing verification principles into AI-assisted engineering environments.

6.5 Limitations

No engineering methodology should be evaluated solely on the basis of successful observations.

The limitations of the investigation are therefore considered an essential component of the proposed methodology.

First, the investigation was conducted within a real engineering environment rather than under controlled laboratory conditions.

While this increased ecological validity, it reduced experimental control.

Engineering problems varied considerably in complexity, available documentation, participating AI systems, project maturity, and verification objectives.

Consequently, the observations cannot be interpreted as the results of a tightly controlled scientific experiment.

Second, the participating reasoning systems were not static throughout the investigation.

New AI models became available, existing models evolved, system prompts changed, software capabilities improved, and engineering workflows adapted continuously during the research period.

The observed verification behaviour therefore emerged despite changes in individual participants rather than under fixed experimental conditions.

Third, the investigation focused primarily on engineering verification within software architecture, implementation, documentation, security, governance, and operational engineering.

Whether the proposed methodology produces similar behaviour in other domains, such as medicine, law, scientific research, education, or creative disciplines, remains an open research question.

The present investigation makes no claim regarding such generalisation.

Another important limitation concerns the absence of objective correctness metrics.

The investigation deliberately evaluated engineering confidence rather than absolute correctness.

In many real engineering situations, definitive objective answers do not exist.

Instead, engineering decisions must be made under uncertainty using incomplete information.

The Consensus Verification Framework was therefore developed to improve confidence in engineering decisions rather than to guarantee their correctness.

The methodology likewise cannot eliminate systematic errors shared by multiple participants.

Independent reasoning processes may still rely upon similar assumptions, common training data, incomplete documentation, or shared misconceptions.

Consensus alone should therefore never be interpreted as proof that a conclusion is objectively correct.

Human engineering judgement, external evidence, testing, and practical validation remain indispensable components of responsible engineering practice.

Finally, the conceptual models introduced in this paper, including Layered Reasoning and the Consensus Verification Framework, represent interpretative constructs developed to explain the observations documented during this investigation.

Their usefulness ultimately depends upon independent replication by other researchers, engineering teams, and industrial environments.

Accordingly, the conclusions presented in this paper should be understood as a proposed engineering interpretation derived from documented reflective practice rather than as definitive scientific theory.

The value of the methodology will ultimately be determined through continued application, independent evaluation, and practical engineering experience beyond the scope of this investigation.

Chapter 7 — Conclusion

This investigation began as a practical engineering exercise.

The original objective was not to develop a new verification methodology, but to understand how different AI-assisted engineering systems behaved when solving real engineering problems under operational conditions.

As the investigation progressed, the focus gradually shifted.

The comparison of individual AI systems became less significant than the interaction between independent reasoning processes.

Repeated engineering verification demonstrated that confidence was established not through reliance on a single participant, but through structured comparison, recursive review, evidence collection, and continuous refinement of engineering understanding.

The observations documented throughout this paper led to two principal interpretative contributions.

The first is Layered Reasoning, introduced as a conceptual model describing how engineering understanding evolved through successive layers of analysis during repeated verification cycles.

The second is the Consensus Verification Framework (CVF), proposed as an engineering methodology for organising independent reasoning into a repeatable verification process.

Neither concept is presented as a guarantee of correctness.

Instead, both seek to improve the quality and transparency of engineering verification while preserving the central role of human engineering judgement.

Perhaps the most significant outcome of this investigation is a shift in perspective.

Rather than asking how artificial intelligence can produce better answers, this research suggests that engineering practice may benefit more from asking how verification itself can be organised more effectively.

Within this view, the quality of interaction between independent reasoning processes becomes at least as important as the capability of any individual participant.

An unexpected observation emerged during the preparation of this paper itself.

The writing process deliberately followed the same verification principles described throughout the investigation.

Successive drafts were repeatedly reviewed, challenged, revised, and compared against earlier observations.

Statements that could not be traced directly to documented evidence were removed or reformulated.

The resulting manuscript therefore became not only a description of the proposed methodology but also a practical demonstration of its application.

This experience reinforced one of the central observations of the investigation.

Engineering confidence increased not through producing immediate answers, but through allowing sufficient opportunity for independent review, self-criticism, comparison of alternative interpretations, and iterative refinement of conclusions.

The investigation therefore suggests that the pursuit of immediate responses may not always represent the optimal strategy for complex engineering decision-making.

Instead, structured verification appears to provide a more reliable foundation for responsible engineering practice.

Finally, the work presented here should be regarded as the beginning of a broader engineering research programme rather than its conclusion.

A natural continuation of this investigation is the development of engineering systems capable of incorporating these verification principles into their own operational behaviour.

Rather than treating verification as an external activity performed after reasoning has concluded, future engineering platforms may integrate independent reasoning, recursive self-review, evidence-based confidence assessment, and structured verification directly into their internal architecture.

The long-term objective is therefore not to reduce the need for engineering verification.

It is to develop engineering systems that continuously improve the quality of their own verification as an integral part of normal operation.

If successful, such systems would not replace engineering judgement.

They would extend one of engineering's oldest principles into the era of AI-assisted development: that confidence is earned not by trusting a single answer, but by systematically verifying how that answer was reached.

Note on the Role and Timing of the References

The external literature cited below did not provide the starting point for this work. The observations, terminology, and provisional methodology emerged first from spontaneous engineering practice, repeated audits, and reflection on the retained project record. The relevant literature was sought only after those ideas had taken a sufficiently coherent form to be compared with existing research.

The references therefore serve as retrospective contextualisation and corroboration rather than as sources from which Layered Reasoning or the Consensus Verification Framework were derived. They show that several principles encountered independently in this workshop—such as the value and limits of independent versions, repeated reasoning, structured disagreement, and risk-oriented governance—have also been examined through other theoretical and empirical approaches. This relationship should not be interpreted as proof of the present methodology, nor as a claim that the cited authors made precisely the same argument.

The existence of related prior work does not diminish the practical significance of the observations reported here. On the contrary, the independent convergence is informative: a practice-led investigation, without using those publications to formulate its initial concepts, arrived at conclusions that are compatible with parts of the established literature. The earlier research helps to interpret and delimit the present claims, while the present project provides a contemporary engineering instance in which related principles re-emerged under different conditions. In that limited sense, the relationship offers two-way contextual reinforcement.

This sequence differs from a conventional literature-led study in which a topic, hypothesis, and research design are established before the investigation begins. Here, the phenomenon was noticed during ongoing development, the provisional account was constructed from practice, and the literature review followed as a final verification and positioning step. Future papers in this series may follow the same order when their subject matter likewise emerges from workshop practice; in each case, the chronology and the evidential limits should be stated explicitly.

References

1. Knight, J. C., and Leveson, N. G. (1986). An Experimental Evaluation of the Assumption of Independence in Multi-Version Programming. IEEE Transactions on Software Engineering, SE-12(1), 96–109. https://doi.org/10.1109/TSE.1986.6312924
2. Wang, X., Wei, J., Schuurmans, D., Le, Q. V., Chi, E. H., Narang, S., Chowdhery, A., and Zhou, D. (2023). Self-Consistency Improves Chain of Thought Reasoning in Language Models. The Eleventh International Conference on Learning Representations (ICLR 2023). https://openreview.net/forum?id=1PL1NIMMrw
3. Du, Y., Li, S., Torralba, A., Tenenbaum, J. B., and Mordatch, I. (2024). Improving Factuality and Reasoning in Language Models through Multiagent Debate. Proceedings of the 41st International Conference on Machine Learning, PMLR 235, 11733–11763. https://proceedings.mlr.press/v235/du24e.html
4. Huang, J., Chen, X., Mishra, S., Zheng, H. S., Yu, A. W., Song, X., and Zhou, D. (2024). Large Language Models Cannot Self-Correct Reasoning Yet. The Twelfth International Conference on Learning Representations (ICLR 2024). https://openreview.net/forum?id=IkmD3fKBPQ
5. Tabassi, E. (2023). Artificial Intelligence Risk Management Framework (AI RMF 1.0). NIST AI 100-1. National Institute of Standards and Technology. https://doi.org/10.6028/NIST.AI.100-1
6. Project audit record (2026). Security audit and verification series: R1/R2 findings and verification runs at 19:15, 19:45, and 20:30 on 26 May 2026. Retained project artefacts; predominantly English.
7. Project conversation record (2026). Multi-participant encryption architecture review and phased remediation discussion, 27 May 2026. Retained local transcript; predominantly Hungarian.

8. Project audit record (2026). Default-team reference workflow: primary dual audit, independent Codex audit, correction, and closing addendum, 19 July 2026. Retained project artefacts; Hungarian.
9. Project audit record (2026). Team UUID versus legacy slug audit, independent reaudit, and targeted recheck, 23 July 2026. Retained project artefacts; Hungarian.

AI Contribution Statement

AI Research and Engineering Participants

The AI systems are acknowledged not merely as writing tools, but as participants in the engineering verification process examined by this paper.

Named AI participants documented within the scope of PS-ERS-001 included ChatGPT, Codex, Claude Code, Gemini, and MiMo. Additional reasoning processes were accessed through Cursor. During portions of the documented work, Cursor operated in automatic model-selection and multi-model modes; the identities of all underlying models cannot therefore be reconstructed reliably and are not attributed individually in this paper.

These systems contributed in distinct roles, including engineering analysis, independent review, methodological criticism, implementation support, manuscript development, and verification of successive drafts. Their disagreements, convergences, recursive reviews, and changing interpretations formed part of the observed process from which Layered Reasoning and the Consensus Verification Framework emerged.

AI-generated outputs were treated as engineering contributions requiring comparison, criticism, traceable evidence, and human evaluation rather than as authoritative answers. No AI system is identified as a formal author or copyright holder.

Gabor Zoltan Nyarfadi defined the engineering objectives, directed the investigation, selected and interpreted the evidence, resolved publication decisions, approved the final manuscript, and accepts responsibility for the accuracy, integrity, and claims of the work. The paper is issued in the name of Poplar Systems Ltd.

Copyright and Licence