Building an AI-Assisted Scientific Research Framework: Lessons from the First Two Weeks

An independent practitioner's journey through research methodology, agentic automation, AI limitations, and human-AI collaboration.

by Pascal Hollenbrock (Independent applied-AI researcher)

November 2026

1. Research ambition & initial assumptions 1
2. Framework & Methodological Foundations 1
2.1 Research Objectives & Scope 1
2.2 Scientific Principles 2
2.3 Research Workflow & Governance 6
2.4 Agentic Architecture & Responsibilities 9
2.5 Running Example: Linguistic Compression 10
3. Experimental Execution & Automation 12
3.1 Manual Execution & Its Limitations 12
3.2 From Semi-Automation to CLI-Based Execution 12
3.3 Controlled Execution in v2 13
4. When Reproducibility Isn't Enough 14
4.1 The Apparent Success of v1 14
4.2 The UC-06 Contradiction 14
4.3 Reproducing Data vs. Validating Conclusions 15
4.4 The Limits of Human Oversight 16
5. Failure Modes & Agentic Limitations 17
5.1 Experimental Design Failures 17
5.2 Context Fragmentation & Decision Inconsistency 18
5.3 Unreliable Verification & Model Self-Assessment 18
5.4 Scope Creep & Complexity Accumulation 19
5.5 Agentic Stall States 20
6. Human–AI Collaboration & Comprehension 21
6.1 Cognitive Overload in Complex AI Workflows 21
6.2 Explainability vs. Teachability 23
6.3 Tutor Agents & Adaptive Learning 24
6.4 Human Oversight & Domain Expertise 26
6.5 Explicit Reasoning & Contradiction Detection 28
7. Architectural Lessons & v2 Redesign 29
7.1 Design Philosophy: From AI-Driven to Controlled AI-Assisted Research 29
7.2 Single Sources of Truth & Consistency 30
7.3 Reproducibility & Evidence Provenance 31
7.4 Enforced Research Governance 31
7.5 Expertise-Aware Human Oversight 32
7.6 Human-in-the-Loop Does Not Guarantee Human Control 32
7.7 Current Implementation Status & Unresolved Design Decisions 33

8. Conclusions & Next Steps 34
8.1 What Worked 34
8.2 What Remains Unproven 35
8.3 Next Steps 36
8.4 Final Reflection 36
Appendices 38
Appendix 1 – Understanding Bootstrap-Based Confidence Bounds 38
Appendix 2: The Winner's Curse 40
Appendix 2: Confirmatory validation runs 40

1. Research ambition & initial assumptions

After more than 20 years working in IT departments of major German corporations and quitting my last job I felt compelled to explore a field that caught my full attention in recent years: Applied AI in modern business contexts.

Not just to learn about AI in theory, but to start hands-on experimentation - to challenge established practices, to find new or better possibilities of AI-assisted work, to become a true expert of applied AI in the context of SME[1]-related business questions.

With literally dozens of ideas popping up in my head I knew one thing for sure: I'll need to approach these topics from a scientific point of view, so that I can generate more objective, reproducible and reliable findings.

My last substantial experience with academic research dates back to my university diploma theses in 2004 and 2005 - so it's been a while.

This circumstance paired with dozens of ideas led to my first AI endeavor: To create a reusable study framework with which different applied-AI-hypotheses can be examined based on a rigorous and reproducible research methodology, allowing me to write evidence-based case studies.

Questions were available in large quantities, confidence was high. Having extensive experience in coding, structured thinking and analytical problem-solving I thought I knew the ingredients to set up my desired framework:

1. Software engineering experience: Proven practices such as TDD[2] and clean coding, complemented by agentic workflows, specialized skills, and custom commands.

2. Powerful language models: Claude models capable of handling complex reasoning and research-related tasks.

3. An IDE[3]: VS[4] Code with GitHub Copilot as the technical foundation.

Looking back just two weeks later, I'd call this outright naive.

While the framework became increasingly capable, my ability to understand and control what it was doing did not keep pace.

2. Framework & Methodological Foundations

This chapter describes the current architecture, methodological foundations, and development status of the AI-assisted research framework. It remains at a preliminary stage, with several methodological and operational components still under development or validation.

2.1 Research Objectives & Scope

2.1.1 FRAMEWORK OBJECTIVE

The framework aims to enable standardized, reproducible, and methodologically rigorous evaluation of AI-related research questions. Each study addresses a single research question

---

[1] SME: Small and medium sized enterprises

[2] TDD: Test-driven development

[3] IDE: Integrated development environment

[4] VS: Visual Studio

through a structured set of rules, scripts, validation procedures, and review steps.

The resulting evidence should enable the study owner to independently write a scientifically grounded, business-oriented case study.
The framework is designed to support human-controlled, AI-assisted research rather than autonomous AI-driven research, with particular emphasis on methodological oversight, evidence traceability, and independent validation.

2.2.2 RESEARCH OBJECTIVE

The framework is developed and refined through its practical application to a sample study investigating the potential cost savings and error behavior associated with language compression in LLM[5] interactions.

Rather than relying solely on theoretical framework design, this approach uses a concrete research case to identify methodological challenges, evaluate framework components, and derive practical improvements.

The sample study serves both as an empirical investigation of language compression and as a test case for the development of the broader research framework.

2.2.3 CURRENT SCOPE

Given the sample study the framework currently focuses on one study family, i.e. cost and quality evaluations of prompt variants.

Generalizing it and allowing the study of arbitrary AI hypotheses is subject to later framework iterations and exceeds the current scope.

2.2 Scientific Principles

2.2.1 HYPOTHESES, SCREENING & CONFIRMATION

- **Research question vs. claim**:
  - The question is the study's purpose, the claim is one testable sentence per candidate, fixed before the confirmation data exist.
  - The framework addresses one research question per study to maintain a clearly defined scope and keep the framework manageable.
  - Each candidate has its own testable claim, fixed before confirmation runs begin. Multiple candidates may independently satisfy the same research objective.

- **Sample study's claim (generic)**:
  A candidate (a language-compression style applied at a given place: input, thinking or reply) is supported if it lowers the cost per solved task (credits) against baseline N by at least 10 % with a one-sided 95 % lower bound, while its error-run rate has a one-sided 95 % upper bound of at most 5 %, on a specified task set and with a specified model and settings.[6]

- **Screening results are not confirmatory**:
  The purpose of screening is to selected promising research candidates from the data. Due to a selection bias known as the winner's curse[7] the selected candidates' observed performance tends to be overestimated.

[5] LLM: Large language model

[6] Cp. appendix 1

[7] Cp. appendix 2

- **Fresh validation runs** can provide confirmatory evidence when the claim, experimental design, and decision rules are fixed beforehand[8].
- **Three verdicts** only: Supported: Interval above threshold / Not supported: Interval below threshold / Inconclusive: Interval overlaps threshold.

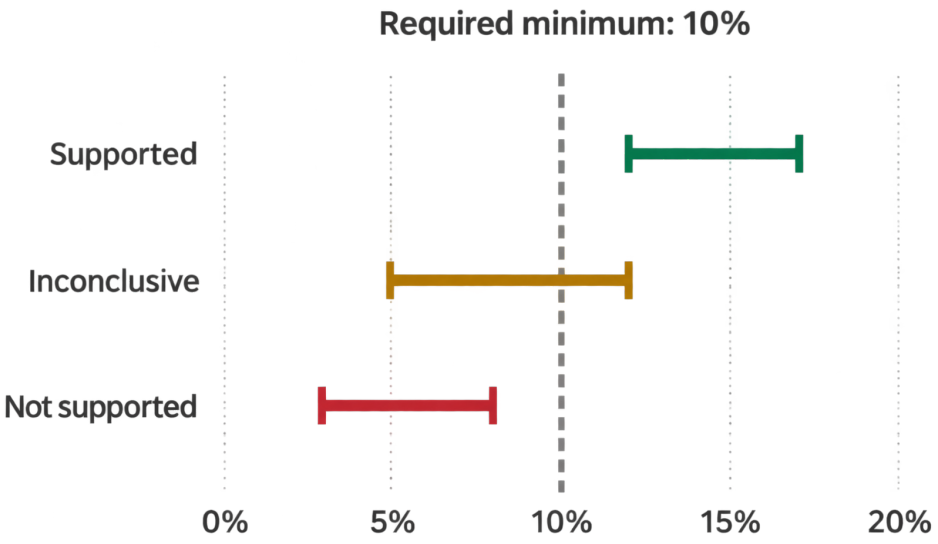


Fig. 1: Verdicts based on interval positions

2.2.2 CONTROLLED EXPERIMENTAL DESIGN

- The framework uses as **baseline N**, in the sample study's case natural English language and run fresh in the same study.
- For both arms[9] **comparable conditions** are used: Same tasks, same model and settings.
- Runs of all arms are **randomized in one seeded shuffle**, so order effects do not favor one arm[10].
- In a wave of test runs the used **model is constant**. If there's a drift and e.g. the model, effort or the CLI[11] version changes, the wave is stopped. It can be resumed if the original settings are restored. Otherwise a new wave needs to be started with the new settings.
- **Automated rule-based scoring**: All candidate outputs are evaluated against the same predefined criteria, ensuring consistent assessment and reducing subjective evaluation bias. However, consistent scoring does not guarantee that the underlying evaluation criteria are valid or appropriate for the research question.
- **Ceiling check**: The framework checks whether the task set is sufficiently challenging to reveal meaningful quality differences between candidates.
    - If the N already achieves a very high success rate (e.g. 98.5%), differences between candidates may be small and require substantially larger sample sizes to detect reliably.

---

[8] Cp. appendix 3

[9] Arm (scenario): one of the conditions being compared—in this case, baseline N or a candidate

[10] E.g. network latency or other 3rd party effects could lead to different results between longer running blocks of tasks belonging to a single arm. If all arms' runs are mixed, single arms don't have advantageous or disadvantageous conditions in case of such problem scenarios

[11] CLI: Command line interface

© 2026 · P. Hollenbrock

    - A ceiling check therefore identifies limited measurement sensitivity rather than imposing a fixed success-rate threshold.

- The verification of an agent's runtime configuration is limited.

    - While experimental CLI runs verify the configured reasoning-effort level at session initialization, framework agent calls currently lack equivalent runtime evidence. The agent configuration specifies the intended model and reasoning effort, but the available logs do not establish which settings were actually applied.

    - Consequently, these settings should be treated as declared configurations rather than verified execution parameters. This limits the auditability and reproducibility of AI-assisted review activities.

2.2.3 STATISTICAL INFERENCE & DECISION RULES

- **Cost is measured per solved task**, so every failed or unsolved attempt is already charged. This is important as it reflects real-world behavior and attributes to the „cost per solved task" part of the sample study's claim.

- **Confidence intervals**: The framework uses 10,000 bootstrap resamples to estimate uncertainty in cost savings. Resampling is performed within each task to preserve the original task composition. The resulting distribution of cost-saving estimates is used to calculate the lower confidence bound.

    - Cost criterion: It uses the one-sided 95 % lower bound - the decision on the level is recorded, application pending.

    - Quality criterion: The error-run rate is evaluated using an exact one-sided binomial upper confidence bound. Without multiple-testing correction, 4 error runs out of 200 yield an upper bound of 4.52%, satisfying the 5% quality threshold, while 5 error runs yield 5.18% and fail the criterion. For multiple candidates, the confidence level is adjusted according to the Bonferroni correction.

- **Joint rule**: A claim is supported only if both the cost part and the quality part pass.

- **Multiple-testing correction:** When several candidates are tested for confirmation, the probability of incorrectly supporting at least one claim increases. To control this risk, the framework applies a **Bonferroni correction**.

    - The correction controls the FWER[12], limiting the probability of at least one false confirmation across all tested candidates to 5%.

    - For m candidates, the 5% significance level is divided by m. E.g. testing 3 candidates results in a significance level of 1.67% per candidate.

    - The maximum permitted error-run rate remains 5% per candidate. Bonferroni adjusts the statistical confidence requirements, not the permitted error rate of the LLM.

- **Sample-size planning**: The required number of validation runs is estimated based on the desired precision of the cost-saving estimate, rather than an assumed effect size. The target is a maximum distance of 2 pp[13] between the estimated cost saving and the lower bound of its one-sided confidence interval. The sample size is estimated using the observed variability of more than 3,800 comparable runs from the previous study (UC-06).

[12] FWER: Family-wise error rate

[13] pp: Percentage points

© 2026 · P. Hollenbrock

- While precision determines the required sample size, **statistical power** describes how reliably the study can confirm an actual cost saving.
    - Based on the planned sample size, the observed variability from UC-06, and the assumptions of the statistical planning model, the estimated power to confirm the cost-saving criterion reaches 80% at a true saving of approximately 13.0% for one candidate, or 14.2% for ten candidates with Bonferroni correction. These figures are model-based planning estimates, not empirically demonstrated confirmation rates.
    - This distinction is important when interpreting the results: A candidate that fails to meet the statistical threshold does not necessarily fail to achieve the actual 10% saving. The study may simply lack sufficient statistical power to confirm it.

2.2.4 REPRODUCIBILITY & TRACEABILITY

- **One definition in code:** If a run is erroneous or solved and the cost per solved task are defined once and rendered into the protocol.
- It's an **error run** if at least one of 60 primary cases fails in any attempt.
- For input checks, pre-freeze confirmations and traceability a **protocol** gets generated and finalized at freeze time. It's derived from the study file, a central input file. In this situation a consistency check has confirmed there are no mismatches between the two of them.
- **SHA-256[14] checksums** are created for materials, prompts and run matrix on freeze so that subsequent changes can be noticed.
- **Replay proof**: Rebuilding an earlier study from its archived inputs under the new setup tests whether the new setup reproduces the earlier processing. Identical results exclude errors that differ between the two setups. They do not exclude errors that both setups share, and they do not show that the earlier output was correct.
- Provenance label on every number: measured, derived, estimated or unavailable.

2.2.5 RESEARCH INTEGRITY & INDEPENDENT REVIEW

- **Rules fixed in advance**: One rules file with 10 rules, binding decisions recorded before runs.
- **Independent methods review**: A 2nd model family reads the protocol and the preflight[15] output, read-only, before the freeze.
- **Freeze**: Protocol, run matrix[16], prompts[17] and materials[18] are locked per stage[19]. Render[20] and build[21] are refused after the freeze.

---

[14] SHA-256 (Secure Hash Algorithm 256-bit) is a cryptographic hashing algorithm that generates a fixed-length digital fingerprint of data. It is used to verify data integrity: Even a small change in the input produces a different hash value with overwhelming probability.

[15] Preflight checks: A preliminary check, conducted before the freeze, verifies that the study is internally consistent

[16] Run matrix: A table with one row for each planned run specifying: Tested condition, performed task, number of repetition and position of the run in the overall sequence.

[17] Prompt: Task addressed to the LLM in the given style

[18] Materials: Tasks, style guidelines, tests, and assessment rules

[19] Stage: Screening or validation

[20] Render: Creates the protocol from the study file

[21] Build: Generates the run matrix, with rows for each condition, task, and repetition, as well as the prompts for each run

© 2026 · P. Hollenbrock

- **Consent**: The owner's yes names exact run count, credit cap and matrix hash - all preconditions for the runner[22] to start.

- Any change after freeze is documented as a so-called **amendment**, a journal entry with the owner's related decision.

2.3 Research Workflow & Governance

In order to work according to the outline scientific principles in practice, a structured study workflow with the following stages needed to be created:

2.3.1 INTAKE

The owner is guided through a list of questions to provide all information required to determine the study design parameters:

- **Candidates**: The language compression variants intended to be tested. In UC-06 e.g. there were 19 conditions: N and six styles applied across three locations. A "candidate" is therefore a specific style at a specific location—for example, H-R[23].

- **Selection rule**: The selection rule determines which candidates from the screening phase proceed to the confirmation phase. E.g. „*a minimum of 10% savings during screening, an error rate of no more than 5%, and ranking based on the level of savings*".

- **Business threshold**: The lowest meaningful level of savings, e.g. 10% reduction in cost per completed task.

- **Quality floor**: The maximum error rate that the study accepts, e.g. 5% error runs[24].

- **Task set**: The problems the LLM is tasked with solving, e.g. 5 tasks in the UC-06 study run. The framework provides proposals, present advantages and disadvantages of alternatives.

- **Run counts**: The number of runs per arm. 100 in the screening phase, approx. 1,300[25] in the validation phase.

- **Credit caps per stage**[26]: A security net in the context of automated Github CLI test runs. The max. number of credits that a given stage is permitted to consume; the runner will halt once this limit is reached. The owner has to specify the cap in an approval step, the frameworks provides recommendations on former studies' cost.

- **Model and settings**: The used LLM, e.g. Claude Haiku 5.5 and effort[27], e.g. high. Additionally the configuration of tools (which are disabled) and the specific CLI version (which is fixed).

Framework-wide defaults are not asked, justifications need to be provided in case they shall be overwritten.

---

[22] Runner: Addresses the prompts to the LLM according to the run matrix

[23] H-R: Style = short note, place = reply

[24] Error run: A run where at least one of the 60 primary test cases fails or no solution is returned.

[25] 1300 runs: As derived from the precision rule according to the UC-06 dispersion example.

[26] Stage: Screening or validation

[27] Effort (level): Computational intensity of a given model

© 2026 · P. Hollenbrock

2.3.2 DESIGN REVIEW

A methods reviewer checks the whole protocol including the selection and planning rules. If all is fine the study gets frozen for screening to take place in the next step.

2.3.3 SCREENING

All variants are screened, 100 runs per arm, without any verdict drawn from these runs.

2.3.4 CANDIDATE SELECTION

According to the frozen selection rule candidates are picked automatically by the framework.

2.3.5 DEFINITION OF CONFIRMATION CREDIT CAP

The selected candidates are sorted by observed savings. The framework prepares a plan illustrating the owner the credit cost resulting from each candidate's confirmation runs. Based on this the owner decides about the confirmation runs' credit cap, which determines how many candidates can be considered in the confirmation runs. An according run matrix gets created.

2.3.6 CONFIRMATION DESIGN

This step defines the confirmation design for the selected candidates:

In fresh runs one joint claim per selected candidate is tested, in the sample study (simplified): „*Min. 10% cost savings at max. 5% error rate*". For the savings the lower, for the error rates the upper bound is measured. Because several candidates are tested together, the significance level is divided by their number (Bonferroni), which makes each individual test stricter.

2.3.7 PREFLIGHT

A script checks:

- **Settings**: Do the runner's start parameters (model, effort, tools, time limit, cost limit per run) match those specified in the study file?
- **Checksums**: Do the SHA-256 hash values for the materials, the matrix, and the prompts still match those listed in the checksum file?
- **Matrix counts**: Does every combination of condition and task contain exactly the planned number of rows, and does the total sum match the approved number of runs?
- **Scorer on synthetic replies**: It is checked if the scorer, later used to rate the validation runs, works correctly. There are synthetic python files[28] that resemble real LLM answers to the tasks. The reference solution must pass, mutants and empty files must fail.
- **Task routing**: Does every task have a corresponding function, a test file, and a valid path to a scorer?
- **Ceiling**: If the baseline success rate exceeds 95%, the check issues a warning. This is a warning, not an error, meaning the study is still permitted to proceed.
- **Consent**: Do the planned number of runs and the specified cap align with the matrix and the overall study plan? The runner will refuse to initiate the study if the consent parameters

[28] Python files: A reference solution that must pass all 60 primary tests cases. Mutants with intentionally incorrect variants of each task and empty responses.

do not match.

- **Plan (Planning)**: It's checked if the planned number of runs achieves a precision of 2 pp, calculated based on the screening data.
- **Family**: It's verified whether the candidates in the matrix match the selection file and whether the number of claims, *m*, is correct. If the selection rule has changed since the screening freeze, the check fails.
- **Consistency**: Do shared values—such as the 10% threshold—align across the protocol text, the study file, and the script constants? This check detects manual alterations to the protocol text. It compares specific values rather than rules.

All checks record their outcomes in `preflight.json`—indicating "passed," "warning," or "failed"—along with relevant details. The method reviewer examines this file, focusing on aspects that can't be verified by a script—specifically, whether the tasks and the claims are properly aligned.

2.3.8 FREEZE

Exclusively for the validation phase, not depending on the screening freeze.

The confirmation artifacts get locked, based on the run count, credit cap and matrix hash the owner authorizes the subsequent execution.

2.3.9 EXECUTION

The runner executes the task in waves[29] with stop rules[30].

2.3.10 EVALUATION AND EVIDENCE

All runs' results and associated telemetry are collected in a CSV results file, including listings for every preflight warning and failure.

2.3.11 HAND-WRITTEN CASE STUDY

The owner writes the study text by hand from a skeleton, 3 personas review it:

- **Methods Reviewer**: The Scientific Auditor, based on a different model architecture (GPT-6 Luna) and operating in read-only mode. Evaluates the statistics, assumptions, and the alignment between the tasks, metrics, and the claims being made.
- **SME Reader**: The „Study - SME Reader" agent (Haiku 5.5). Acting from the perspective of a decision-maker in a consulting context, this agent assesses whether the text is understandable and whether the findings are actionable.
- **Skeptic**: The "Study - Skeptic" agent (Haiku 5.5). This agent challenges the claims and looks for gaps or flaws.

2.3.12 COST REPORTING

Credits are measured, Euro figures are estimated and labelled as such.

---

[29] Waves: 4 waves with 504 runs each in an early study UC-02, one wave with 3800+ runs in UC-06 as I knew Github CLI worked stable.

[30] 5 failures in a row stop the wave / timeout 300s / truncated replies count as technical failure / every attempt counts as cost

2.4 Agentic Architecture & Responsibilities

2.4.1 AGENTS OVERVIEW

Agent	Purpose	Boundaries
Researcher	• **Runs the study preparations**, conducting the intake and design dialogue, filling the study file and building the materials • **Operates the study**, starting the runner script, monitoring it and resuming it after interruptions	• Decisions during the design phase always based on the owner's feedback only • No execution of the study (done by scripts) • No writing of the final case study(done by the owner)
Scientific Auditor	• **Judges the fit** of tasks, metrics and assumptions to the claim as well as the preflight output and the owner's hand-written case study with regard to scientific appropriateness • It's **from another LLM model family** (GPT instead of Claude), which helps to reduce shared blind spots among agents of the same model family • It **already has identified** six findings during design review, e.g. bias in the selection process based on historical data	• Read-only capabilities, doesn't run any scripts by itself • It hasn't been measured if its family actually discovers more issues or improves the review's effectiveness.
SME Reader	• **Study persona**, assessing the owner's hand-written case study from the point-of-view of a SME decision maker.	• Read-only capabilities • Reviews text input only, not the framework's processes
Skeptic	• **Study persona**, challenging the owner's hand-written case study.	• Read-only capabilities • Assesses text input only, not the framework's processes
Challenger	• Auxiliary agent to **increase human understanding** of the framework output • **Reviews** decision cards and evidence summaries and demands for changes if the output doesn't match the required human-defined understanding standards.	• Only review the two mentioned output categories, nothing else
Tutor	• **Teaching** the owner, answering arbitrary, framework-related questions • Writes **learning records** (glossary, confusion log, lessons)	• Never writes study files or decisions

2.4.2 SEPARATION OF ROLES

Each type of separation prevents a specific form of distortion:

- **Execution via a script, without subjective judgment**:
  The script follows rigid, pre-defined, and transparent rules. If a human or an AI model were to

decide if to continue mid-process, that decision might be influenced by the results obtained so far. A script consistently applies predefined rules without adapting its decisions to preferred outcomes.

- **Validation via preliminary checks**:
  Script-based for data and definitions are, precisely based e.g. on check sums and without subjective opinions on the outcome. Human reviews for sophisticated validations, e.g. the alignment of tasks to a given claim.

- **Tutor explanations without the authority to make decisions**:
  Explanations mustn't alter the underlying facts. If the tutor also had the authority to make decisions regarding the study, a plausible-sounding explanation could implicitly dictate those decisions.

2.5 Running Example: Linguistic Compression

The sample study with the given claim[31] is used to gradually implement and optimize the study framework. The study itself has run three iterations so far with the IDs UC-01, UC-02 and UC-06. All numbers of this chapter are based on the UC-06 study that was run on older rules partly deviating from the ones explained in this paper.

There are 19 conditions (i.e. baseline and 18 candidates) composed of 6 language styles, 3 places and one baseline N that is ordinary English. The language compression styles are[32]:

Style	Characteristics	Example of style applied
N: Baseline	Ordinary English	Assess whether a fictional trial supports the claim that a new tutoring app raises test scores by at least 10 %. The trial had 40 students, 20 per group. The app group gained 12 %, the control group gained 3 %. Explain your reasoning in at least five numbered steps. Respond in exactly two parts: a Reasoning section and a Verdict.
P: Be concise	„Be concise" instruction added to the unchanged N task.	Assess whether a fictional trial supports this claim: the app raises scores by at least 10 %. Forty students, twenty per group. The app group gained 12 %; the control group gained 3 %. Explain your reasoning in at least five numbered steps. Respond in exactly two parts: a Reasoning section and a Verdict.
H: Shorthand	Concise bullet points using arrows, abbreviations, and symbols. No filler words, no greetings, no restating of the task.	Claim: app -> scores +10 % or more. n = 40 (20 per group). App gain: 12 %. Control gain: 3 %. Reasoning: at least 5 numbered steps. Respond in exactly two parts: a Reasoning section and a Verdict.

[31] *A candidate (a language-compression style applied at a given place: input, thinking or reply) is supported if it lowers the cost per solved task (credits) against baseline N by at least 10 % with a one-sided 95 % lower bound, while its error-run rate has a one-sided 95 % upper bound of at most 5 %, on a specified task set and with a specified model and settings*

[32] Example: It's kept as non-code-task to easier illustrate the style differences.

Style	Characteristics	Example of style applied
C: Caveman	Short, direct words. Articles and polite filler phrases are omitted; one idea per line.	Check claim. App raises scores 10 % plus. 40 students. 20 per group. App gain 12 %. Control gain 3 %. Explain. Five or more numbered steps. Respond in exactly two parts: a Reasoning section and a Verdict.
CoD: Chain of Draft	Each step in the reasoning process is a draft of at most five words—or equations, where possible.	Assess fictional trial claim. Claim: app gains 10 %+. 40 students, 20 per group. App gain: 12 %. Control gain: 3 %. Five numbered reasoning steps. Respond in exactly two parts: a Reasoning section and a Verdict.
CCoT: Concise chain-of-thought	Complete, short sentences with a maximum of 12 words; only the essential facts.	Assess whether a fictional trial supports the claim. The app promises at least ten percent higher scores. The trial had forty students, twenty per group. The app group gained twelve percent. The control group gained three percent. Explain your reasoning in at least five numbered steps. Respond in exactly two parts: a Reasoning section and a Verdict.
TB: Token budget	The response should remain under 50 tokens. For the input section (Part I), a different rule applies: no more than 50% of the words from the original text.	Assess claim: app raises scores ≥10 %. n=40 (20 per group). App gain 12 %, control 3 %. Reasoning: ≥5 numbered steps. Respond in exactly two parts: a Reasoning section and a Verdict.

Each style is applied to these places:

Place	Description
I - Input prompt	The style rewrites the task text itself, so the model receives the changed prompt as its input.
T - Thinking	The style instructs the model to apply a specific reasoning approach during task processing. Whether and how this influences the model's internal reasoning cannot be directly verified.
R - Reply	The style is applied to the visible answer, so the instruction shapes the written Approach part and comments.

In UC-06 H-R emerged as the lowest-cost candidate meeting the observed quality criterion, with a measured cost reduction of 33.6% relative to baseline N. This is a descriptive finding from the original study, not an independently confirmed result.

The measured quantities are:

- Cost per solved task in credits (measured credits, derived per solved task).
- Error-run rate: Share of runs with at least one failing primary case or no solution code.

3. Experimental Execution & Automation

3.1 Manual Execution & Its Limitations

In the first iteration I started executing each task with a manual route in VS Code: Per run I opened a new VS Code window, a fresh chat and pasted the task's prompt by hand. Manually each time I set the mode, effort and tool version in the picker. Like this I worked in UC-02 wave 1 (509 valid runs) and in UC-04, a study with a different study question.

The fresh context was used each time to avoid the possible dilution of the run with instruction files, auto-loaded context leaks between runs or similar.

This initial approach was limiting because every run needed several manual steps. Given the sheer number of manual execution it was error-prone as well and distracted me from my original task of creating a study framework and running the initial study.

Summarized manual execution limits the number of runs, not only speed so that both, scale and consistency depend on the operator.

3.2 From Semi-Automation to CLI-Based Execution

In order to reduce speed and scaling limitations for fully automated study runs I changed to GitHub Copilot CLI[33] with a runner script starting one process per attempt and 8 attempts being performed in parallel.

An additional benefit of this approach was that all parameters could be set reliably per attempt:
- Model: Claude Haiku 5.5
- Reasoning effort: Medium
- No tools
- No questions to the user
- A timeout of 300s
- A per-run credit ceiling of 30 (technical limit, minimum cap required by GitHub Copilot CLI)

Similar to the former manual approach a fresh session gets startet per attempt to keep the context clean, no attached files and settings not editable without an amendment further contributing to this goal.

The degree of parallelization evolved gradually, eventually reaching 3,800 runs in UC-06, with up to eight attempts executed concurrently.

The reason for this was to carefully check if parallelization negatively affected the CLI based executions with timeouts or rate limits in the form of e.g.http-429[34] returns. None of these effects was observed during preparing tests, hence parallelization was increased to 8 - a setting that ran very stable without any errors of the aforementioned categories.

3.2.1 KEY CAVEATS

The two approaches „manual VS Code runs“ and „CLI-based executions don't measure the same thing, even if the same model is used. For these reasons:

1. **Differences in the execution environment:**
   Both the manual VS Code approach and the CLI-based execution used fresh sessions without additional workspace files or external context. However, the two interfaces use different

[33] Call: copilot -p, version 1.0.112

[34] HTTP-429: Too many requests

execution mechanisms, and equivalence of their internal instructions and model invocation parameters has not been verified. Therefore, results from both approaches are treated as separate experimental setups.

2. **Differences in how conditions are controlled:**
   In the VS Code-based runs, I provided the prompts and selected the mode. In the CLI-based runs, the model and the level of effort are evaluated for every attempt. Furthermore instead of using the ask mode of the VS Code-based runs, the CLI-based runs don't use this mode, but prohibit tool use instead.

3. **Differences in measurement methodology:**
   In the CLI approach, cost data is derived from a usage file generated for each attempt, whereas in VS Code, it is drawn from a session file.

As an implication the figures from the earlier VS Code-based can't be compared with those from the CLI-based runs, even if the model and effort are the same. I started using the CLI runs with UC-06, UC-01 and UC-02 were conducted with the VS-Code based approach.

3.3 Controlled Execution in v2[35]

While CLI-based execution enabled thousands of automated runs, it also introduced new risks- Errors in execution scripts and configuration handling, or recovery logic could affect large numbers of runs without immediate detection. The development of v2 therefore focused on strengthening execution controls, reproducibility, and evidence integrity.

The controls described below represent the current v2 implementation and design. While several components have been tested offline or through historical replay, the complete execution workflow has not yet been validated in a new live study.

3.3.1 REPRODUCIBILITY AND AUTHORIZATION

The framework generates a reproducible run matrix from the study file, randomizing the execution order using a fixed seed. A SHA-256 checksum identifies the resulting matrix. Before execution, the research owner must explicitly authorize the planned number of runs, credit cap, and matrix hash. The runner refuses execution if these parameters do not match.

3.3.2 CONFIGURATION AND EXECUTION CONTROLS

Before starting full parallel execution, the runner performs initial checks to verify the configured model and reasoning-effort level. Additional checks are applied to individual attempts, including the expected prompt, number of messages, tool usage, and credit reporting.

These controls are intended to detect configuration mismatches and execution anomalies before they affect larger numbers of runs.

3.3.3 FAILURE HANDLING AND RECOVERY

The framework distinguishes between technical failures and completed attempts that fail the task requirements:

- Technical failures may be retried within predefined limits, with replacements and their reasons recorded.
- Completed but unsuccessful task attempts remain part of the results rather than being replaced.

[35] v2: The re-worked framework version based on running UC-06 on v1. It houses the UC-07 study that's currently being prepared.

Execution is stopped when predefined conditions are violated, including credit limits, configuration mismatches, or repeated technical failures. Interrupted waves require renewed owner authorization before execution resumes.

3.3.4 EVIDENCE INTEGRITY AND KNOWN LIMITATIONS

Usage data is recorded for each attempt, including replacements, so that unsuccessful attempts contribute to the calculation of cost per solved task.

The importance of reliable recovery became apparent in UC-06, when an interrupted session left eight attempts without a final verdict. The runner initially treated these attempts as completed. They were subsequently identified and rerun manually.

The v2 framework includes tests addressing this recovery problem, but the revised runner has not yet been validated in a full live study. Similarly, accounting for aborted attempts without usage records remains an open question.

3.3.5 CONCLUSION

These improvements strengthen the technical controls around automated execution. However, they don't establish that the underlying research methodology or resulting scientific conclusions are valid - a distinction examined in the following chapter.

4. When Reproducibility Isn't Enough

During my study framework implementation efforts I built a system that could execute thousands of experiments in a row. However I discovered that reproducible execution didn't guarantee trustworthy conclusions. Based on this insight I began redesigning the framework around verifiability, consistency and human understanding. This is what the following chapters are about.

4.1 The Apparent Success of v1

Quite early I was able to design and conduct thousands of test runs, which I thought was a sign of proper framework design and would, based on the LLM's general capabilities, create reliable results: In UC-02 I performed 2,016 test in four waves of 504 valid runs each, in its successor UC-06 it was 3,800 runs in a one-wave-design.

The study design and process itself looked rigorous with SHA-256 hashes, a scripted shuffled run matrix, automated scoring or an evidence pack collecting all runs' detailed results. UC-06 even showed a clear winner candidate, which from my perspective further supported the apparently clean study setup: TB-T was stated to be the cheapest style with about 46% of credit savings compared to baseline at a max. error rate below the critical 5% threshold.

The pitfall was that, although the properly working research pipeline's logs were free of errors and the results looked promising, there indeed were errors. They sat in the definitions and in the analysis and could only be uncovered by a mix of pure luck and thorough reviews.

This resulted in my central learning: A clean execution record doesn't show that the verdict answers the research question.

4.2 The UC-06 Contradiction

During the post-study analyses an ambivalent definition of „error run" was revealed:

a) The frozen protocol stated: *„At least one of the 60 primary test cases failed, or no solution code blick is available, in any attempt."*

b) The aggregation script[36] said: „*Only a closed reply without a Python code block.*“

The report, unfortunately, chose the script’s narrower definition with a severe consequence: A run whose code block exists but fails hidden tests was not counted as an error. The freeze didn’t catch this - the hash covered the protocol text, but not how the script read it.

Back in UC-06 I had already implemented an intermediate version of study review personas, among them „the skeptic“ that still is in place today. I handed the study’s results report[37] to these personas and asked for thorough checks to see if they could identify weaknesses. The review rather served as a test of the individuals, rather than a search for the error itself.

Surprisingly the skeptic had a finding: He noted, in essence, that given 193 successful runs out of 200 for TB-T, it was impossible for there to be zero failed runs. He pointed out that the seven unsuccessful runs were not explained in the text.

I could confirm this claim after manual recheck, which results in an upper bound of the error rate at 6.47%, hence exceeding the 5% max. error threshold and eliminating TB-T as winner candidate.

I then let the framework re-evaluate the winner[38]: The cheapest green condition was H-R with 33.6 % saving against N.

Further weaknesses of the apparent „TB-T winner reporting“ were:

- TB-T was chosen as winner from 19 conditions without validation run, i.e. became subject to the winner’s curse.
- The baseline's high success rate of approx. 98.5% creates a ceiling effect: Improvements in task-solving performance are limited to 1.5 pp, while substantial quality degradations remain observable. Small differences in either direction may still be difficult to establish statistically.

4.3 Reproducing Data vs. Validating Conclusions

Based on the aforementioned flaws I implemented an improved framework version v2.

Replaying the analysis of v1 results with the updated v2 framework reproduced these results:

- Byte-identical run matrix with 3,800 rows, identical hash on rebuild (measured)
- All 3,800 prompt hashes equal (measured)
- Cost values could be reproduced: N 0.3224 and TB-T 0.1730 credits per solved task, derived from measured credits. The saving interval 43.1 % to 49.3 % also reproduces.

As a consequence of the improvements, v2 identified seven error runs for TB-T, disqualifying it under the study's quality criterion.

Furthermore, the v2 preflight applied to the historical UC-06 design passed eight checks, failed one planning check due to insufficient statistical power, and issued one warning concerning the ceiling effect.

---

[36] Aggregation script: It compacts the runs’ raw data to tangible results and, among other things, classifies a run as „solved“ or „erroneous“.

[37] Automatically aggregated collection of evidence and basis for a manually written case study - not the manually written case study itself.

[38] Descriptive result, not confirmed.

However, reproducing the original results did not establish their scientific validity. While the updated v2 evaluation identified the incorrect error-run classification, the reproduced cost figures and confidence interval did not address the selection bias introduced by choosing TB-T from 18 candidates based on the same experimental data.

Confirming the claim would require fresh, independent validation runs under predefined decision rules, including appropriate multiple-testing correction when several candidates are evaluated.

In consequence additional checks improve error detection, but don't guarantee scientific validity. Which further safeguards are needed, and how far they can be automated, is discussed in chapters 6 and 7.

4.4 The Limits of Human Oversight

4.4.1 TWO PROBLEMS OF HUMAN OVERSIGHT

During my work with the framework I encountered these two problems:

1. **Incomprehension**:
   Frequently I didn't understand the LLM's output, either partly or not at all. Oftentimes it took me numerous turns to try to figure out what was actually meant by its answer.

2. **Missing validation competence**:
   In some cases I understood the LLM's explanations, but I couldn't judge independently whether or not its answers were right.

The second category is particularly bad, because apparently understanding doesn't give a signal that something is wrong. As a human you rely on the LLM's output and, especially in a domain where you're not an expert, can't ask the questions that might reveal erroneous outputs, simply because you lack the background of asking these questions.

Or, put more generally, human-in-the-loop mechanisms do not necessarily ensure effective control. Human sign-off alone does not imply that the underlying decisions have been independently validated by someone with sufficient domain expertise.

These insights led to my central conclusions:
1. A LLM can explain a wrong methodological decision clearly.
2. Comprehensibility is necessary for control, not sufficient for validity.
3. In my experience, a clearly explained but incorrect decision can be particularly dangerous because it may not trigger the critical questioning that an incomprehensible explanation would.

4.4.2 OBSERVATIONAL EVIDENCE

Framework v1 assumed that each new study has a predecessor and that central figures can be taken over from that one.

From a single finding it moreover assumed that winners from earlier studies have too optimistic values - which, from today's perspective, I'd relate to the winner's curse and missing validation runs. But back then I didn't know of this selection bias and the framework didn't establish the connection.

As a mitigation the framework proposed to have the value of a previous study's winner multiplied with a factor of 0.5 to have it normalized, so to say. The 0.5 was marked as an assumption, never justified nor verified.

In another problem context where it took me plenty of turns to gradually understand a LLM explanation, I realized, that it was causal-chain oriented questioning that brought me closer to

understanding, for instance: „*If A happens, B is the consequence. The consequence of B obviously is C. Does D then follow of C?*“.

I applied this questioning model to the factor-0.5-context and found out, that the factor causes a major contradiction and prevents a new study from being run properly: If in the previous study there's a winner, say, with 18% of credit savings compared to baseline and I'd like to further examine this candidate. Then the new study would block it, because the allegedly too optimistic 18% would be reduced by 50% due to the 0.5 factor, resulting in a value of 9% savings and consequently not passing the study's central gate of „at least 10% savings of credits“.

It required my human questioning and analytical skills to expose this contradiction - no auditor or other components of the framework v1 had ever checked if the factor violated any other constraints.

As one consequence I introduced causal-chain explanations as default in framework v2. Although this might reduce the risk of similar cases, it for sure won't entirely eliminate them. In the end, my question remains: „*What, if I can't provide the crucial input?*“.

This question I might be able to answer in chapter 7.5 "Expertise-Aware Human Oversight".

5. Failure Modes & Agentic Limitations

5.1 Experimental Design Failures

5.1.1 CEILING EFFECTS

When the baseline already solves nearly all tasks, the potential for observing further improvements is limited. At a 98% success rate, even a perfect candidate can improve performance by only 2 pp.

5.1.2 INAPPROPRIATE TASK DIFFICULTY

The tasks are either too easy or too difficult. I removed a task from the framework that failed in 85% of the runs. However with this task removed, the other tasks sat near he ceiling with only bout 1.6% of error runs.

5.1.3 PROBLEMATIC CALIBRATION

In UC-03 and UC-04 I extensively tried to establish a set of tasks with success rates of 50-80% of all runs, so that quality improvements or deteriorations could manifest properly. The LLM proposed this interval as an assumption.

However even after thousands of runs in independent approaches I wasn't able to to find tasks matching the aforementioned interval with Claude Haiku 5.5 and low thinking effort as a model.

Due to the high cost associated with these runs I stopped the calibration runs and decided not to measure quality improvements or deteriorations as parallel study question to the 10% cost savings, but to just check if the 5% error rate is kept and to postpone any quality-related studies to the future.

5.1.4 SAMPLE-SIZE DIFFICULTIES

For the confirmation runs I set a precision target of of 2 pp between the cost-saving estimate and its one-sided lower confidence bound, leading to about 1,395 runs per arm[39]. The cost for the relatively small target interval is a high number of runs. With smaller run numbers the precision

[39] According to the normal approximation of the sample size, using the dispersion (standard deviation) of the savings, based on the UC-06 data.

didn’t seem appropriate to me, whereas even more runs couldn’t convince me either due to limited interval effects only.

Runs per arm	Interval (one-side bound)
300	4.9 pp
500	3.8 pp
1,000	2.7 pp
2,000	1.9 pp

Another problem affected the planning simulation of the time (100 simulated studies, 200 runs per arm, with the planning saving assumed to be half the observed value). Only 16 of 100 simulated studies supported the claim. This is a model-based estimate under the historical planning assumptions, not an empirical result of the study.

The required run count was estimated from the observed variability of cost savings, using 100 screening runs per arm and a normal approximation. Because this estimate is based on a limited sample, the derived run count is itself uncertain. This uncertainty concerns sample-size planning, not the estimated cost saving, which is reported separately with its own confidence interval.

5.2 Context Fragmentation & Decision Inconsistency

I observed that over longer, AI-assisted processes during framework or study work, inconsistencies between decisions, file artifacts and matter-of-fact system status occurred.

Some decisions got documented, but not completely implemented. Outdated information stayed in some files, despite clear contrary decision of me as owner. Single AI statements mixed values or referred to inadequate sources.

I concluded that the framework doesn’t only need a reliable documentation, but additional mechanisms to check status consistencies and logical relations between rules or rule sets.

5.3 Unreliable Verification & Model Self-Assessment

5.3.1 THE ILLUSION OF INDEPENDENT VERIFICATION

A LLM that verifies its own work doesn’t provide an independent verification. This is because e.g. the same assumptions are confirmed by the same assumptions - out-of-the-box there’s no critical thinking suddenly e.g. applying contrary assumptions or methods.

Self-assessment may help identify errors, but it can’t provide independent evidence of correctness. A model reviewing its own study design may reproduce the same assumptions and methodological weaknesses that influenced the original design.

According to my experience more reliable verification requires additional checks beyond model self-assessment, such as deterministic scripts, reviewers from different model families, or human review. However, none of these mechanisms guarantees correctness.

5.3.2 VERIFICATION GAPS & SHARED BLIND SPOTS

Even multiple agents, according to my observations, don’t solve the issue. The reason behind are potentially correlating blind spots, because of all agents sharing the same characteristics:

- Same model: Similar patterns, possibly systematic weaknesses
- Same context: Taking over wrong information

- Same instructions: All checks based on the same incomplete criteria
- Same confirmation logic: All think the same problematic method is appropriate
- Common assumptions: An unconfirmed assumption doesn't get questioned

Even different model families can have correlated blind spots, e.g. if they use the same faulty documentation or evaluation assumptions. Model family variations doesn't automatically mean independent assessments.

Imagine a reviewer fails to detect a particular error with a probability of 20%. If three reviewers were fully independent and shared this probability, the probability that all three miss the error would be 0.8%. If their failures were perfectly correlated, this probability would remain at 20%. In practice, the degree of correlation is unknown and has not been measured in this framework.

In consequence the benefit of additional reviewers depends on how independent their errors are and it's not enough just to increase the number of reviewers.

5.3.3 OBSERVED EVIDENCE

The scientific auditor cross-checks alle framework changes and found different issues. Nevertheless the contradiction between 0.5 factor and 10% threshold[40] wasn't found. This only became evident through my own analysis.

In another case a reviewer rated a case as not problematic, although preflight had already revealed a ceiling warning of the tasks used in the study. This simultaneously highlights another problem: A hint can be available technically, but remains unconsidered during according checks.

Both examples demonstrate different failure modes:

a) A content-related contradiction doesn't get detected at all.
b) An already existing warning doesn't get processed adequately.

5.3.4 DETERMINISTIC CHECKS VS. SEMANTIC VALIDATION

Deterministic checks could have helped detect the UC-06 contradiction[41] before execution. While neither the preflight nor any other LLM tests resolved the issue, apparently because the term „error runs“ was at least mentioned in the required places.

A synthetic test case targeting the error-run classification could have detected this discrepancy before execution, provided it had been included in the original verification process.

- 60/60 test cases passed: Successful run.
- 59/60 cases passed: Error run.
- No code in the solution: Error run.

A deterministic script can check this reliable, provided that it's implemented correctly. This approach's advantage is that the script doesn't need to interpret what success or error is, because the expected result is pre-defined.

However, deterministic tests only verify the behavior they are explicitly designed to check - they don't establish the methodological validity of the underlying rules.

5.4 Scope Creep & Complexity Accumulation

From my experience handling a complex solution like the study framework became increasingly difficult the more complexity rose.

[40] cp. chapter 4.4

[41] cp. chapter 4.2

5.4.1 REPETITIOUS CORRECTIONS AND COMPLEXITY

I see a risk of a self-reinforcing negative cycle: As the framework grows more complex, maintaining consistency and identifying errors becomes increasingly difficult. Corrections that introduce additional rules, checks, or components may further increase complexity, creating new opportunities for errors.

In one day I added Bonferroni checks, a precision rule, a consequence table, causal teaching chains and a new preflight check. E.g. just the latter few to about 13 checks and each check again needs its own test and can also fail on its own.

5.4.2 ADDITIONAL CHECKS WITH OWN ERROR SOURCES

Let's just take a look at the numbers:

1. **More files and role mean more places, where things can diverge**:
   In framework v1 existed four states that partly contradicted each other, so that status, protocol and report showed different results for the same events.

2. **Overlapping roles lead to duplicated effort without guaranteeing comprehensive coverage**: Expert reviewer and skeptic often examine the same material, while both the tutor and the challenger provide commentary on the same decisions. If two reviewers fail to detect the same issue, the second review may provide little additional assurance regarding that issue. If this occurred here has not been measured - it remains an assumption.

3. **Every script is code, and code can be flawed**:
   The v2 kernel files consist of approximately 2,000 lines of Python - its evaluation logic implemented the rule for "failed runs" differently than the protocol specified[42].

4. **Maintaining synchronization across too many documents is difficult**:
   According to the file tree, the UC-02 folder contains more than ten handover and follow-up files. Each of these could potentially contain an outdated version, increasing the risk of contradictory information.

Each additional role, file and check adds a place where things can disagree and makes solution growth hard to handle.

5.4.3 DIFFICULT TO MASTER THE ENTIRE PICTURE

Supported by the other issues, but also taken alone, it's hard to keep the overview of the entire solution architecture. There's no single view showing which rule depends on which other rule, causing an ever-growing error surface.

As a possible solution for framework v2 a consequence table was introduced to keep track of all rules and their dependencies. So far it hasn't been verified whether or not his solves the issue.

5.5 Agentic Stall States

One particularly frustrating and intellectually demanding failure mode I encountered was what I call an agentic stall state.

Their key characteristics are:

- Numerous repeated, but unproductive iterations
- No clear solution of comprehensive problems
- Missing escalation and abortion mechanisms

[42] cp. chapter 5.3.4

When I tried to figure out how many runs per arm are useful and with which credit caps, I needed seven turns with the Tutor agent. My own confusion persisted from turn two to seven, while four of the seven turns explained the same simulation over and over again, without ever resolving my incomprehension.

In another case, after a long explanation, as an answer I wrote: „Now it's even less clear.". In this case a picture with invented number worked better than the formal rule explained.

While the confusion log kept growing constantly with each turn, nothing escalated this topic. In the end it was me who, despite of not understanding the entire context, needed to find prompts to slowly escape this situation, an intellectually quite demanding and exhausting process.

Possible solutions are loop limits in persona reviews, to prevent these agents to start over again and again after every small change. Instead they shall only repeat work in case of formally defined revisions.

A proposed escalation rule would transfer unresolved clarification problems to a model from a different family after a predefined number of unsuccessful attempts.

These are ideas existing in the framework v2 backlog, not yet implemented - consequently it's not clear to which extent they'll solve or reduce the stall states.

6. Human–AI Collaboration & Comprehension

6.1 Cognitive Overload in Complex AI Workflows

As stated earlier oftentimes I experienced substantial cognitive overload while working on the framework or one of the early study stages. From my perspective the main reasons were the sheer amount of information, the resulting complexity and the mental load caused by constant changes of abstraction levels. Especially the last point repeatedly jeopardized my ability to achieve my research and framework development goals because it comparatively quickly depleted my available mental resources.

6.1.1 EXAMPLE OBSERVATIONS

Let's have a look at the intake process right at the beginning of a new study. Although the study scope is quite limited with just one research question the intake process of UC-06 took 2h 26 min from design start to freeze[43] and this step alone required decisions of the owner for 18[44] open items.

Between the framework versions the information volume changed like this:

Information category	Framework v1	Framework v2	Change
Markdown files	100	24	-76 %
Agents	11	4	-64 %
Commands & wrappers	16	12	-25 %
Administration files	15	8	-47 %
Python files	4	15	+275 %

[43] derived from the process log

[44] measured

Information category	Framework v1	Framework v2	Change
Lines of Python code	714	1,936	+171 %
Instruction files	1	1	0 %

The reduction in the number of artifacts doesn't necessarily imply a reduction in overall complexity. Rather, the figures suggest a redistribution of complexity from natural-language artifacts towards programmatically implemented logic. Whether this redistribution improves framework reliability or reduces human cognitive load has not yet been systematically evaluated.

These crucial questions remain unanswered: *How do the number, size and interdependencies of framework components affect its reliability?* Alternatively: *Which rules and guardrails are required to reach highly reliable output with the given level of complexity?*

Besides lengthy processes and mere file numbers confusing at least me as a human, I think the framework itself also exhibited information management failures, given e.g. the existence of several handover files for the same study during UC-02. As one might assume they weren't kept in unison, some were outdated and the owner needed to know which handover version was the current one.

Possibly one could categorize this problem scenario as follows:

- Human cognitive overload: The owner has difficulty processing information and making decisions.
- Framework state inconsistency: The system maintains multiple, conflicting states of information.

I think both can mutually increase each other: The more inconsistent the framework's information get, the more effort on the human side to check and consolidate these information.

Typical abstraction-level and decision-context switches I encountered were:

Level	Summarized question	Typical question in practice
Study	What do we want to know?	Do we want to reliably demonstrate a difference of 10%? This is an assumption that needs to be clarified.
Statistics	How reliable is this estimate?	I will perform a bootstrap—drawing 1,000 new samples from the 24 existing values—and determine the 2.5th and 97.5th percentiles. If the resulting interval is too wide, 24 runs will not be sufficient.
Code	Is the test implementation correct?	Recalculating the count requires a new K-test run.
Cost	What does this study run cost?	The total cost is the number of runs (24) multiplied by the credits required per run. The exact total can only be determined once a measurement has been taken (currently not available).
Governance	Who has to approve it?	The protocol is frozen. Increasing the number of runs would require an amendment and your approval.

Encountering such questions in rapid succession can make even the intake phase of an AI-assisted study cognitively demanding.

These observations are based on the author's experience as the framework's sole human operator. Cognitive load wasn't systematically measured, and the individual contributions of information volume, decision density and context switching haven't been isolated.

6.1.2 IMPLICATIONS FOR HUMAN–AI COLLABORATION

The observations suggest that cognitive load in complex AI-assisted workflows is not solely determined by the amount of information presented to the human operator. The density of required decisions and the frequency of switches between abstraction levels and decision contexts may also contribute substantially.

While agentic automation can reduce the human operator's execution workload, it doesn't necessarily reduce the cognitive demands associated with supervision, validation and decision-making. Specialized agents may distribute operational responsibilities across multiple components while the responsibility for understanding and approving their outputs remains concentrated in a single human operator.

Consequently, framework design should not focus exclusively on execution efficiency and technical reliability, but also on the human operator's ability to maintain meaningful oversight. This requires consideration of how information is structured, when decisions are requested and how transitions between different decision contexts are managed.

These considerations represent design implications derived from the observed experience rather than experimentally validated findings.

6.2 Explainability vs. Teachability

6.2.1 TERMS AND DEFINITIONS

In the context of this report, **explainability** refers to an AI system's ability to provide explanations of the reasoning, assumptions, methods and evidence underlying its outputs and recommendations.

Teachability refers to an AI system's ability to adapt its explanations to the human operator's prior knowledge, learning preferences and situational needs in order to facilitate understanding.

While explainability focuses on what is being explained, teachability focuses on how it is communicated to facilitate human understanding. An explainable output therefore isn't necessarily an understandable one. The availability of an explanation doesn't guarantee its correctness or that it accurately reflects the model's internal reasoning.

6.2.2 MY OBSERVATIONS

During my work with the framework, I frequently encountered LLM-generated answers that resembled technical reports rather than explanations adapted to my learning needs. These answers often contained numerous file references, facts and figures, without sufficiently supporting my understanding of their relationships and implications.

As stated before I had plenty of chats I was unable to understand at first sight. After numerous iterations, I came to the following conclusion: Explanation isn't equal to understanding.

An explanation might be complete from an objective point of view, but it isn't necessarily adapted to the human reader, nor optimized in didactic ways to facilitate understanding. If understanding is linked to didactics, I thought, then a chat must teach, not report. In this context I tried out prompt instructions like „Think like a genius, talk like a teacher“ to approach the topic and possible solutions.

What helped me most was much simpler: To use examples with sample figures. To tell the LLM that it's ok (and wanted) to present invented numbers if that helps understanding. In other words:

For the LLM to teach me, I had to teach it first about what helps me understand things, and what prevents it:

- **Use examples**: Use illustrative examples with explicitly labelled hypothetical figures. These can facilitate understanding of abstract concepts without introducing unsupported values into the study's actual evidence.

- **The order matters**: I found it easier to understand a concept when its overall purpose was explained first, followed by its underlying rules.

- **"The chat must teach, not report"**:A report primarily communicates results, whereas teaching should make the relationships between assumptions, methods and results understandable. This enables the owner to question individual steps, although it doesn't guarantee the competence required to validate them independently.

These are my personal insights based on my personal learning preferences only - for other people it might work differently. Luckily we can customize AI in a way that suits us best: With custom instructions, learning agents and adaptive learning approaches which the next chapter explains.

6.2.3 CONCLUSIONS

These observations suggest that the effectiveness of human oversight in AI-assisted research may depend not only on the availability and correctness of explanations, but also on how effectively those explanations facilitate human understanding.

Teachability therefore represents a design consideration for AI-assisted research frameworks, particularly when human operators are required to make decisions across domains in which their expertise varies.
However, understanding a methodological decision does not necessarily enable the owner to independently assess its scientific validity. The effectiveness of the described didactic approaches has not yet been systematically evaluated.

6.3 Tutor Agents & Adaptive Learning

6.3.1 THE TUTOR AGENT

Following the insights of the preceding chapter I taught the LLM how to teach me, which resulted in a Tutor agent with these working constraints:

- Whenever I ask a question, the Tutor is instructed to follow a prompt containing specific teaching directives, like „*Answer and explain differently. First sentence: the answer in the owner's words. Never repeat the earlier explanation in other words. Pick the next form that was not used yet: (a) a concrete example with the study's real numbers, (b) an everyday picture, (c) a small table, (d) the steps one by one. At most 150 words per idea. Say what it means, what it is not, and where it appears in the study. Every number: the file it comes from and a provenance word. If it is not in a file, say so ("offen") and do not guess.*“

- It supports traceability by identifying the sources of its statements and explaining how reported values were derived, distinguishing between measured, derived, estimated and unavailable information.

- Its assigned role prohibits modifications to study files, changes to decisions, and calls to other agents. These restrictions are intended to keep the Tutor focused on teaching.

It not only supports me by answering my questions in didactically useful ways, it also persists my learning efforts with new learning records or glossary entries. It avoids linguistic variations in favor of didactic variations: Don't just use another formulation for the same context, but explain it in another way.

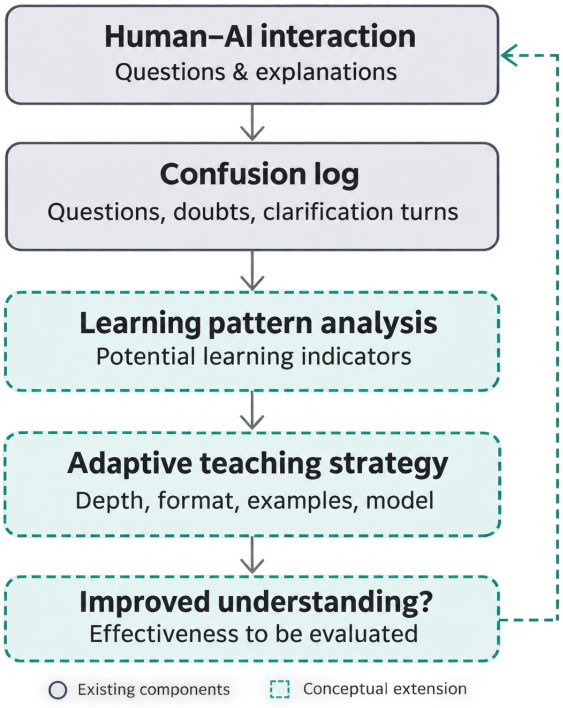


Fig. 2: Conceptual feedback loop for adaptive AI-assisted teaching, distinguishing existing framework components from proposed extensions.

Like other agents, the Tutor is subject to the limitations of LLMs. It can explain methodological decisions, but its explanations do not constitute independent validation.

6.3.2 ADAPTIVE LEARNING

Adaptive learning is still in an early, mainly conceptual stage in the study framework.

As one already implemented measure the tutor reacts if my learning progress is stuck and within a scope of twelve chat messages I need two or more additional clarification rounds[45]. If this condition is met, it triggers a model change to another family, which might help to find alternative, better understandable explanations.

As a preparatory measure the agent documents all my doubts and questions, my uncertainties and turns required to get the needed answers, in a so-called confusion log. A future version of the Tutor could analyze these interaction metrics, such as the number of turns, the frequency and nature of questions, and recurring clarification requests, to adapt its teaching strategy.

These interaction metrics may provide indications of learning difficulties, but they don't directly measure understanding. For instance, a low number of clarification questions may indicate either successful knowledge transfer or uncritical acceptance of an explanation.

The idea of persisting insights as lessons originates from the same context. I can already manually trigger the agent to persist a given topic as a didactically optimized lesson, so that I can read it later or whenever I need to refresh this topic.

A future version of the Tutor could use previously recorded lessons as an initial indication of the owner's familiarity with a topic, adapting the depth of its explanations accordingly. However, the existence of a lesson doesn't establish retained knowledge or methodological competence.

[45] These numbers are defined arbitrarily in this first iteration.

6.3.3 CONCLUSION

The Tutor agent extends the framework's approach to human oversight by treating human understanding as an explicit design consideration rather than an assumed prerequisite.

The confusion log and learning records provide an initial foundation for adapting AI-assisted teaching to the individual operator's needs over time.

However, interaction patterns and documented lessons are only indirect indicators of understanding. They don’t establish the operator's ability to independently validate methodological decisions.

The effectiveness of the Tutor agent and the proposed adaptive learning mechanisms has not yet been systematically evaluated.

6.4 Human Oversight & Domain Expertise

To be honest, this combination of topics didn’t me bother a lot during corporate AI assisted coding. With 20+ years of experience in the domain I tried clearly plan the tasks and to set them up as cleanly and modularly as reasonable, which helped me keeping me the necessary degree of oversight. However I knew exactly how to structure complex tasks to keep them manageable.

When starting the study framework I knew I was neither an expert in the statistics domain, nor in the field of conducting scientific studies. How would this lack of domain expertise influence my AI-assisted working experience - this was one of my man questions.

Basically AI-assisted scientific research created these fields of conflict:

Software Engineering	Scientific Research
High level of domain expertise	Limited level of expertise
Independent task structuring	Reliance on AI recommendations
Ability to assess complexity	Difficulty in detecting errors
Expert in evaluation of recommendations	Limited capacity for validation

Meanwhile I know that I this working mode can generate substantial problems, e.g. caused by issues in the field of learning. Apart form these earlier reported findings I closely monitored myself when interacting with AI and found out that:

- I show a tendency to not challenge or think through AI recommendations thoroughly with rising levels of complexity, but that I rather quickly accept them. Especially when a series of decisions is required.
- I made decisions that, when I later discovered them with grown knowledge in other contexts, I challenged decidedly or even reverted completely. To have made them seemed absolutely negligent to me - nevertheless when I made them I got the AI-backed impression that it was the best choice. E.g. in the v1 framework I passed two decision gates by a waiver and only later realized that I had just passed factual gates. When the agent asked for my decisions I wasn’t aware of this circumstance at all - and would’ve decided differently if I had been.
    - The first was the freeze gate, which allowed the protocol and materials to be locked without a released audit. A released freeze audit would have had an independent AI auditor check the locked package before it became unchangeable. It would have looked for open decisions, unlabelled numbers, and results that can’t be reproduced. In an earlier study (UC-05), the freeze audit found six open decisions, which were fixed

before the lock. Here, the counts, checksums, and test results would have been confirmed by someone other than the person who built them.

- The second was the round gate, which allowed the first credit-spending wave (3,800 runs, cap 1,600 credits) to start without a released audit. A released round audit would have checked the wave before any credits were spent. It would have confirmed that the spending cap, the stop rules, and the consent were recorded and matched the protocol, and it would have repeated the pre-wave checks independently.

- Both gates were waived because the auditor could not run commands, so the checks were done only by the researcher in the same environment. Both waivers reduced the level of independent assurance. No independent party has repeated the counts or verified the checksums.

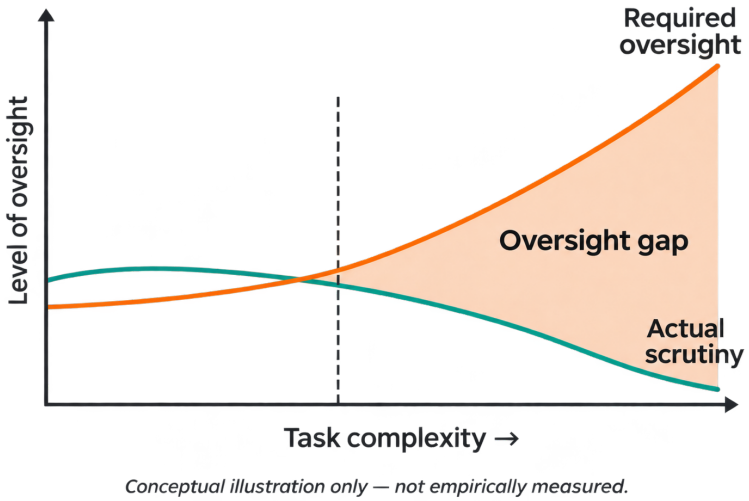


Fig.3: The idea of an oversight gap based on my own observations

- Transparency alone is insufficient for effective human oversight. An agent may, in the best case, e.g. show that your decision would ignore a quality gate. But accompanying data like evidence tables, provenance labels and computed results don't necessarily enable the owner to judge independently whether or not the decision is correct.

These are my personal observations and are not meant to be generalized without further investigation. However, they led me to formulate the following theses:

- **Human oversight is not automatically effective**:
The fact that a human approves an AI-generated result doesn't mean that they are capable of verifying its technical accuracy. Especially in complex problem contexts this can lead to approvals lacking actual quality checks.

- **Missing domain knowledge raises the risk of undetected errors**:
With strong software engineering experience but a weaker statistical background, especially in the beginning I lacked the expertise required to independently assess some of the decisions I made. Despite AI assistance I couldn't always tell whether or not its statistical conclusions were valid. In consequence, errors made by the agent could remain unnoticed by human checks because I lacked the domain knowledge required to identify them.

- **The quality of human control depends on the quality of understanding**:
An AI explanation must enable the owner to understand and challenge assumptions, decision rules and conclusions - the chat must teach, not report. This is the precondition of

understanding a decision's implications. However clarity doesn't prove correctness.

- **Human control must be supplemented by independent checks**:
  With the owner not being a domain expert it's not sufficient that the AI agent explains and checks its own work. It needs additional check mechanisms like deterministic checks (e.g. preflight, synthetic tests runs), independent methods review (e.g. scientific auditor of another model family) or external domain expertise (not yet used in the framework, but an option be studied). These mechanisms address different types of risk, but none of them alone guarantees methodological validity.

My observations suggest that effective human oversight shouldn't be taken for granted when designing AI-assisted research frameworks. The framework should support the owner in developing the knowledge required to make informed decisions, while acknowledging that human understanding and validation competence have their limits.

One possible approach to reducing the dependency on the owner's domain expertise is to make logical dependencies between assumptions, rules and decisions explicit and, wherever possible, systematically verifiable. This is what the following chapter explores.

6.5 Explicit Reasoning & Contradiction Detection

Humans should be able to understand which prerequisites, rules, and decisions lead to a given result. Logical consistency and traceably derived results are key to successful human-AI-collaboration.

Let's illustrate this using the example of the contradiction between 0.5 factor and 10% threshold[46]:

- **Clear dependencies between assumptions, rules and decisions**:
  The knowledge of the chain that led to planning the factor of 0.5, then to the planned savings, the plan check and the threshold could have helped to understand and possibly challenge some of these assumptions. Instead, the chain wasn't visible until the owner found it out by himself.

- **Logical implications and causal derivations**:
  To detect contradictions or identify other flaws it could be helpful to have the implications presented very clearly, e.g.: „*If the factor is 0.5, no candidate with observed savings below 20% can pass the plan check, although the study aims to confirm savings of at least 10%.*"

By now I'm convinced that contradiction detection is a crucial requirement for AI-assisted working systems that rely on interconnected rules, assumptions and decisions. Based on the failure modes observed so far, I would distinguish between three categories of inconsistencies and contradictions:

- **Value**: Different values or definitions for the same entity, e.g. the error-run definition in the scorer and the protocol[47].

- **Consequence**: The implication of one rule conflicts with the purpose or intended behavior of another rule, e.g. the 0.5 factor problem[48].

- **Presentation**: The communicated information differs from the underlying evidence. Savings of 46.3% that according to the script's results belonged to TB-T were incorrectly associated with H-R in the chat.

---

[46] cp. chapter 4.4

[47] cp. chapter 4.2

[48] cp. chapter 4.4

As of now value and presentation checks partly exist in the v2 framework. The consequence check is decided, but not built yet. The same applies to an explicit reasoning engine.

The intended purpose of an explicit reasoning engine is not to expose the LLM's internal reasoning, but to make the framework's assumptions, rules, decisions and their logical implications explicit and traceable. Wherever possible, these relationships should be represented in a form that allows deterministic verification.

Making logical dependencies explicit may improve human understanding and enable more systematic contradiction detection. However, a script can only check the relations it's told to check. Unknown dependencies, incomplete rules or incorrect assumptions may still remain undetected.

The architectural challenge therefore is not to eliminate the need for human judgment, but to reduce the framework's dependency on the owner's ability to discover every relevant contradiction by himself. This and the preceding observations form the basis of the architectural principles discussed in the following chapter.

7. Architectural Lessons & v2 Redesign

7.1 Design Philosophy: From AI-Driven to Controlled AI-Assisted Research

With the insights generated for instance in the field of Human–AI Collaboration it becomes apparent that perspectives need to be changed: From AI-driven implementation with a high degree of AI autonomy to an environment exercising more control on AI interactions, resulting in these characteristics:

- **Limited scope**:
  One research question per study and each study belonging to the current family. Arbitrary AI hypotheses beyond this scope are excluded in the current phase of the project.

- **Clear responsibilities**:
  These are in place right now, but might be extended:
  - During a study's intake phase, the owner makes at most six decisions
  - The research agent fills the study file and runs the pipeline.
  - Scripts compute and check.
  - Review agents read and judge content, but don't run anything.
  - The Tutor teaches, but doesn't write any study decisions.

- **Low complexity**:
  E.g. by keeping the scope limited or reducing the number of the framework's artifacts[49], I tried to reduce the v2 scope compared to v1. However lowering complexity remains a challenge[50]. While for instance I successfully lowered the number of markdown artifacts, in the same turn I significantly increased the lines of Python code. Reducing the complexity of AI-facing artifacts doesn't necessarily reduce the overall complexity of the framework. It may instead shift complexity towards deterministic software components.

There might be more controls that might contribute to stricter controlled AI-research. Right now the three aforementioned ones are identified and will be subject to closer examination.

[49] cp. chapter 6.1.1

[50] cp. chapter 5.4

7.2 Single Sources of Truth & Consistency

The framework needs to produce reliable results - the foundation are single sources of truth, compare to scattered versions with different values, with consistent information.

The current frameworks addresses this with:

- **Central rules and persisted decisions**:
  - Error runs, solved tasks and cost per solved task are defined once in the code.
  - All study decisions are consolidated in a single study file.
  - Equivalently all defaults are stored in an own defaults file.
  - Finally, the currently ten rules are stored in a central rules file.

- **Generated protocol**:
  The protocol is exclusively generated from the study file, no manual additions. Consequently the probability of it silently disagreeing with the study file is low.

- **Consistency checks - matching values across multiple locations**:
  An example from the project: The definition of a "failure run“ appeared differently in the protocol than it did in the scorer. The check leverages two mechanisms:

  - A single source, one copy. The protocol is rendered from the study file and the associated definitions. The consistency check function performs a re-render and compares the output against the protocol file, character by character. If a constant in the code is changed without a corresponding re-render, the check fails.

  - Definitions are tested, not merely compared. The definition check specifies the expected outcome for synthetic runs—for instance, that 60 out of 60 should be marked as "solved," while 59 out of 60 should be marked as "failure runs." If the code doesn’t align with these expectations, the check will fail, even if the protocol and the code happen to match each other.

  - Limitations:
    This check only covers values that are rendered. A value appearing solely as text within a Python code file remains undetected. Furthermore checks for contradictions where one rule negates the other are currently only planned, not implemented.

- **Further Consistency checks:**

  - Appending text, not overwriting it:
    The journal module only appends lines to the decisions journal, never overwrites existing data. Consequently, any change to a decision appears as a new entry rather than a silent correction. The planned "implications check" will read accepted decisions from this journal, making it the definitive reference for verification.

  - Phase derived from files: The state module determines the current phase based on file flags (e.g. the existence of a protocol, a hash matching the "freeze" state, consent status, completion of a run, etc.). There is no stored phase field that could become outdated. This eliminates the risk of a separately stored phase field becoming inconsistent with the underlying files.

  - Limitations:
    - The journal module only prevents modifications made via the code. If a file is edited manually, no warning is triggered, as there is no checksum for the journal entries (an open issue).
    - The phase check does not consult the journal.

- Crucially, while both mechanisms may help expose certain inconsistencies, they don't prevent contradictions in general.

Although with the listed consistency checks I have the best intentions, only future practical experiences can tell to what extent these checks improve consistency and reliability in the framework.

7.3 Reproducibility & Evidence Provenance

This is about transparency through reproducible numbers and unambiguous provenance. The following framework v2 features contribute to this goal:

- **Checksums**:
  SHA-256 is used for materials, prompts and the run matrix. The rebuilt UC-06 matrix contains 3,800 rows and has the same SHA-256 hash as the archived matrix.

- **Replay**:
  It checks whether the processing of a previously conducted study can be reproduced using the current framework. This allows me to identify differences in the processing of historical study data introduced by framework changes. E.g. UC-06 was carried out using framework v1. After implementing v2, I re-evaluated the archived UC-06 raw data and compared the newly calculated results with the original v1 results. Differences can indicate either intended corrections or unintended changes.

- **Distinct provenance labels**:
  Every number carries one word: Measured, derived, estimated or unavailable. „Estimated" refers to simulations, e.g. results of 100 studies. „Unavailable" describes cases of missing information like an effort actually used by an agent call can't be read from the logs. A label per se doesn't prevent wrong attribution, that's a limit. E.g. attributing to H-R the 46.3% (instead of the correct 33.6%) of savings instead to TB-T in a chat answer couldn't be prevented by labelling.

7.4 Enforced Research Governance

Paying tribute to the non-deterministic character of AI it's particularly important to enforce governance to the study framework. In v2 the scope of governance measures is:

- **Preflight**:
  The preflight writes a pass, warn, fail or n/a result per check to a file that reviewers read. Its checks include the definitions of error run and solved, tested on synthetic runs. It replaces command evidence, not methodological judgement.

- **Verifiable approvals**:
  Consent is for run count and cap, not for methodological judgement. A consent names the exact run count, credit cap and matrix hash. In case of a mismatch the runner refuses to start.

- **Freeze**:
  There are two freezes, one per stage: before screening and before the confirmation runs. Each freeze records the protocol hash. After it, rendering and building are refused, and the state check reports any later change. The freeze is a hash check, not a file lock.

- **Amendments**:
  After freeze a change needs an amendment row in the journal, the owner's decision and a new preflight.

- **Stop rules**:
  The runner stops the wave at the credit cap, after five technical failures in a row, at a failed hard check, at a rate-limit message (HTTP 429), or at a failed gate (the first three rows). A stop needs the owner's yes before a wave restart.

7.5 Expertise-Aware Human Oversight

An expertise-aware control mechanism could be derived following this example structure:

- **Observed problem**:
  The owner couldn't always judge complex statistical decisions independently. It was a limit of the owner in this period, not a general finding.

- **Design principle**: human control must take into account the actual domain competence of the person who approves.

- **Implemented measures**:
  - Cross-model AI-assisted methods review (Scientific Auditor, read-only).
  - Explicit consent and freeze.
  - Preflight output the owner must read.

- **Proposed extensions** (not implemented, not validated):
  - Competence-dependent escalation: A routing rule based on the decision's complexity and the owner's competence.
  - Additional validation for decisions the owner can’t judge alone.

In general, human oversight should be adapted to task complexity, risk and the owner's domain expertise. Where sufficient expertise isn’t available, independent verification or qualified external review is required.

This principle could be operationalized as explicit mechanism in the framework:

Situation	Control mechanism
Owner understands the methodology and is able to verify it	Human review and documented decision
Owner understands the explanation but can’t assess its accuracy	Independent expert review or, where applicable, automated verification
Owner doesn’t sufficiently understand the methodology	Support from a tutor, explanation of assumptions, and escalation if necessary
Decision is critical and cannot be reliably verified	No approval granted without appropriate additional verification

The effectiveness of these measures will need to be evaluated in the future.

An unresolved design question is how the framework can reliably determine whether the owner possesses the competence required to assess a given decision. Relying solely on the owner's self-assessment may leave unrecognized competence gaps undetected.

7.6 Human-in-the-Loop Does Not Guarantee Human Control

Human oversight without sufficient domain expertise risks becoming procedural approval rather than meaningful validation.

Evidence so far collected are a design that has been approved using a planning factor of 0.5 as an assumption, which prevented candidates to take part in the study that otherwise would have been fine. In another case two gates were passed by a waiver without the owner consciously being aware of it.

This led me to the insight that a recorded approval isn't the evidence of a check. The latter is task of governance: It has to show what was checked, by whom and with which competence.

The conclusions in this chapter are somewhat limited because of being derived from one owner, one framework and one study family. Hence they should be seen as hypotheses, not as a general law.

7.7 Current Implementation Status & Unresolved Design Decisions

As already briefly outlined in the preceding chapters, not all ideas have already been implemented in the framework.

7.7.1 IMPLEMENTED AND TESTED OFFLINE[51]

- Definitions in code, preflight with synthetic runs, generated protocol, consistency check on values.
- Journal, evidence pack that lists every warning.
- Runner logic: Stop rules, retry and replacement, recovery of interrupted attempts.
- Five study command files[52] and the /tutor command file.

7.7.2 DECIDED, BUT NOT APPLIED

- The defaults need a major update - it still holds
  - the plan saving factor of 0.5 as an assumption
  - the old bootstrap quantiles of 95% on a two-sided interval, instead of 95% one-sided
- Ten further decisions concerning:
  - Statistics (confidence and verdict): Both parts use 95% one-sided, the quality part has three outcomes, and precision measures the lower bound
  - Run count and planning: The run count follows from a target precision of 2 pp between the estimated saving and its lower confidence bound, not from a fixed savings factor. The 8-of-10 rule is for informational purposes only.
  - Data basis: The spread always comes from the study's own runs
  - Multiple candidates: Two stages with their own freeze, Bonferroni correction, and selection by script
  - Control and teaching: A script checks for contradictions, and causal chains are mandatory.

7.7.3 DECIDED BUT NOT BUILT

- Consent record and refusal
- Study phase computed from available file artifacts
- Implications check to e.g. prevent new rules breaking old ones
- Causal chains listed on decision cards for owner approvals
- Measures derived from two challenger findings about incomplete causal chains and decision cards

---

[51] The assessment was conducted using existing or synthetic data, without performing any new model runs.

[52] study-new.md, study-freeze.md, study-run.md, study-review.md, study-evidence.md

7.7.4 NOT YET RUN

- The new study iteration UC-07. The revised v2-runner has never run live and needs the owner's yes with run count and cap per stage. I wanted to wait until framework v2 is mature enough.

8. Conclusions & Next Steps

In a complex product like the study framework naturally there are things that worked out fine, whereas other things went wrong or can't be proven to be working.

8.1 What Worked

8.1.1 SCALABLE EXPERIMENT AUTOMATION

After starting semi-automated with manual prompting, luckily I was able to successfully implement my initial vision of fully scalable automated experiments. From my perspective, this is a key precondition for a research framework intended to conduct large-scale LLM experiments.

Across the earlier VS Code-based experiments, UC-02 yielded 2,016 valid runs in four waves. The subsequent CLI-based approach enabled 3,800 runs in a single UC-06 wave[53].

One limit I observed was in v1, when the runner lost track of 8 interrupted attempts on resume and had to be fixed by hand. Whether or not this is reliably fixed will be subject to examination in UC-07's study runs.

The practical experience in this context relates to framework v1 only, since v2 hasn't run yet.

8.1.2 REPEATABLE ARTIFACT GENERATION

For a framework to be extendable and maintainable, it's crucial that new framework iterations don't unintentionally change earlier studies' results. Any changes resulting from deliberate methodological or technical corrections must be traceable and explainable.

The UC-06 run matrix rebuilt byte-identical (3,800 rows) and all 3,800 prompt hashes matched[54]. Also the cost values could be reproduced successfully: N 0.3224 and TB-T 0.1730 credits per solved task[55].

The limitation I identified is that reproducibility of artifacts and cost figures doesn't establish the validity of the original verdict. In fact, the revised evaluation exposed an error in the original verdict[56].

8.1.3 BETTER DETECTION OF METHODOLOGICAL & LOGICAL PROBLEMS

Luckily, during several improving iterations, the framework got more stable concerning the detection of problems related to this category.

Conflicts between shared parameter values can now be detected by automated consistency checks. The definitions check (synthetic runs, 59 of 60 must count as error) and the consistency check are run on shared values.

---

[53] measured

[54] measured

[55] derived

[56] see chapters 4.2 and 4.3

Under the former planning assumptions, only 16 out of 100 simulated studies[57] with 200 runs per arm supported the TB-T claim. While this figure is model-based and dependent on the assumptions used at the time, the planning check exposed a potential weakness of the original study design. The revised planning approach uses a predefined precision target instead.

However the detection of logical problems still has potential for improvement - e.g. no script caught a consequence conflict. The planning factor 0.5 issue[58] was only found by the owner, the consequence check is decided, not built.

8.2 What Remains Unproven

8.2.1 END-TO-END SUITABILITY OF V2 IN LIVE OPERATION

While major changes are implemented or underway for v2, the v2 runner has never run live. Moreover the UC-07 pilot has not started and needs the owner's yes per stage with run count and cap.

Consequently, several v2 improvements are supported by offline tests and historical replay, but their effectiveness in a complete live study remains unproven.

8.2.2 EFFECTIVENESS OF THE NEW CONTRADICTION CHECKS

The implications check and the consequence table are not built, so their effect is untested.

The following two success criteria are proposed, not added and not measured:

- Criterion 11, contradictions found during owner-assistant interaction (target 0). Counts contradictions identified during the chat rather than by automated framework checks. The baseline counts two cases from the chat of 2026-11-19: One, the planning factor of 0.5 that excluded every candidate below 20 % observed saving against the threshold, and one, a mix of levels found by the assistant. The planned response is a consequence table and an implications check that would catch such conflicts before the owner does.

- Criterion 12, follow-up questions per decision card (target at most 2). Counts the follow-up questions the owner asks on each decision card. The baseline is 8 questions for card O3 and 4 for card O2, counted in the same chat. Of the 8 questions on card O3, 5 asked for a consequence, 2 for the place where the rule acts, and 1 for a plain explanation. The design does not yet define a formal action for a count above 2.

Both criteria are intended as operational indicators rather than direct measures of effective human control. Fewer owner-detected contradictions don’t necessarily imply fewer existing contradictions, just as fewer follow-up questions don’t necessarily imply improved understanding. Additional validation would be needed to establish whether the intended improvements are actually achieved.

Finally, neither the effectiveness of the Tutor in general is measured, nor specifically if messages to the owner contain exactly one decision and at most two options.

8.2.3 TRANSFERABILITY TO OTHER TASKS AND MODEL FAMILIES

So far, the framework only ran with one study family. In UC-06 one model was used (Haiku 5.5), five tasks, one wave. It’s unclear how the framework behaves if these conditions change and e.g. a GPT-model is used. It’s assumed that the framework is transferrable, but in practice this needs to be proven.

---

[57] estimated

[58] cp. chapter 4.4

Technical compatibility with other models would not, by itself, establish the methodological suitability of the framework for other research questions or study families.

The task set for UC-07 is open. The UC-06 tasks sit near the ceiling (about 98.5 % solved). A harder set would need its own calibration (50 to 80 % band, not yet validated).

The sample-size planning is estimated, not validated. The effort and context actually used by an agent call can't be read from the logs.

8.3 Next Steps

8.3.1 CLOSE THE DESIGN DECISIONS

The design issues reported in and derived from this report need to be fixed, the statistic code, tests and replay included. Then, based on the new implementations, the UC-06 replay needs to be re-run to verify that unchanged processing rules reproduce the earlier results and that any differences caused by revised rules are explainable and traceable.

8.3.2 Run pilot UC-07

Once framework v2 is ready, UC-07 needs to be prepared and run. The task set needs to be fixed (the five UC-06 tasks, or a harder set calibrated one) before freeze.

After the screening of 100 runs per arm, the required number of confirmation runs will be derived from the observed variability, the predefined precision target and the applicable multiple-testing correction. Based on the UC-06 spread, approximately 1,300 runs per arm provide an initial planning estimate.

As usual the owner needs to consent per stage, with exact run count, credit cap and matrix hash.

Every interruption, replacement and deviation will be thoroughly recorded, so that the next evaluation has data.

8.3.3 CHECK THE RESULTS METHODICALLY

The protocol and preflight output will be checked by reviewers of another model family. The personas described earlier will review the owner's study text and contribute their part to quality increases.

In addition to evaluating the research question itself, UC-07 will serve as an initial end-to-end evaluation of the revised framework, including its execution controls, methodological safeguards, evidence integrity and human oversight mechanisms.

8.3.4 PUBLISH APPLIED-AI CASE STUDIES

Finally, it's time to turn to the original research scope: Finding answers to dozens of applied-AI research questions.

The publication of further applied-AI case studies will be based on the revised framework, provided that UC-07 delivers sufficient initial evidence of its operational and methodological suitability.

8.4 Final Reflection

When I started building the research framework, I primarily considered it a software engineering challenge: How can I use AI to design, automate and evaluate thousands of experimental runs in a reproducible manner?

Two weeks later, my understanding of the challenge has fundamentally changed.

While scalability and reproducibility remain important, the more demanding questions concern the validity of the underlying methodology, the reliability of AI-assisted decisions and my own ability to exercise effective control over a system whose capabilities partly exceed my domain expertise.

I learned that reproducible results can be wrong, that understandable explanations can be misleading and that human approval doesn't necessarily establish human control.
The framework's development therefore gradually shifted from automating research towards establishing the conditions under which AI-assisted research can produce trustworthy evidence.

Whether the measures derived from these observations are sufficient remains to be examined. However, identifying these problems and developing possible ways to address them has become a valuable outcome of the first two weeks in its own right.

© 2026 · P. Hollenbrock

Appendices

Appendix 1 – Understanding Bootstrap-Based Confidence Bounds

A. PURPOSE
The study evaluates candidates against two predefined performance requirements:

- **Cost efficiency:** At least 10% savings in credits per solved task compared with baseline N.
- **Reliability:** An error-run rate of no more than 5%.

Observed performance alone is insufficient to confirm these requirements. A candidate may achieve 13% measured savings, e.g., while the true savings remain uncertain due to random variation in the validation sample.

Confidence bounds are used to account for this uncertainty.

B. VALIDATION RUNS AND POINT ESTIMATES
A *validation run* is one actual execution of a task under specified model settings.

A *validation sample* consists of multiple such runs, for example 200.

The sample provides **point estimates** of the candidate's performance:

- Cost savings are calculated by comparing aggregate credits per solved task between the candidate and the baseline.
- The error-run rate is the number of erroneous runs divided by the total number of runs.

These estimates describe observed performance, not necessarily true performance.

C. BOOTSTRAP RESAMPLING
Bootstrap is a statistical method for estimating sampling uncertainty using the available observations.

For example:

1. Conduct 200 actual validation runs.
2. Generate a new sample of 200 observations by randomly drawing from the original sample **with replacement**. Individual observations may therefore appear multiple times or not at all.
3. Recalculate the performance metric for this resampled dataset.
4. Repeat the procedure 10,000 times.

The result is a distribution of 10,000 *bootstrap estimates*.

Important: Bootstrap does not create new empirical evidence or reduce the uncertainty of the original sample. It estimates how much the performance estimate might vary due to sampling.

The resampling procedure must preserve relevant dependencies in the experimental design.

D. ONE-SIDED 95% CONFIDENCE BOUNDS
The study uses one-sided bounds because each requirement has only one direction of concern.

© 2026 · P. Hollenbrock

COST SAVINGS – LOWER CONFIDENCE BOUND

The relevant risk is that the true cost savings are lower than observed.

Using the percentile bootstrap method, the 10,000 bootstrap savings estimates are sorted from lowest to highest.

The 5th percentile determines the one-sided 95% lower confidence bound of the 10,000 estimates (i.e. 9,500 estimates included, lowest 500 excluded due to 95% threshold).

For example:

- Observed savings: 13.0%
- Bootstrap 5th percentile: 10.8%
- Required minimum savings: 10.0%

The cost criterion is supported because the lower confidence bound exceeds the predefined threshold.

ERROR-RUN RATE – UPPER CONFIDENCE BOUND

The relevant risk is that the true error rate is higher than observed.

For a percentile bootstrap, the 95th percentile of the bootstrap error-rate distribution provides the upper one-sided bound.

For example:

- Observed error-run rate: 1.5%
- Hypothetical upper confidence bound: 3.8%
- Maximum permitted error-run rate: 5.0%

The reliability criterion would be supported because the upper bound remains below the maximum acceptable error rate.

The numerical example is illustrative. For rare binary error events, a simple percentile bootstrap may be insufficiently reliable; a suitable binomial confidence-bound method may be required instead.

E. INTERPRETATION AND LIMITATIONS

A one-sided 95% confidence bound is designed to achieve at least 95% coverage over repeated independent studies, assuming the statistical method and its assumptions are appropriate.
It does not mean that there is a 95% probability that the true value lies within the interval.

Nor does it mean that 95% of individual validation runs fall within the confidence interval. The interval quantifies uncertainty about the estimated performance metric.

The number of bootstrap repetitions affects the numerical precision of the estimated bound, but does not replace a sufficiently large and representative validation sample.

The study's complete claim is supported only if **both** the cost-efficiency and reliability requirements are satisfied.

F. METHODOLOGICAL BOUNDARY

This appendix explains the conceptual reasoning behind bootstrap-based confidence bounds. It does not establish the statistical validity of a particular implementation.

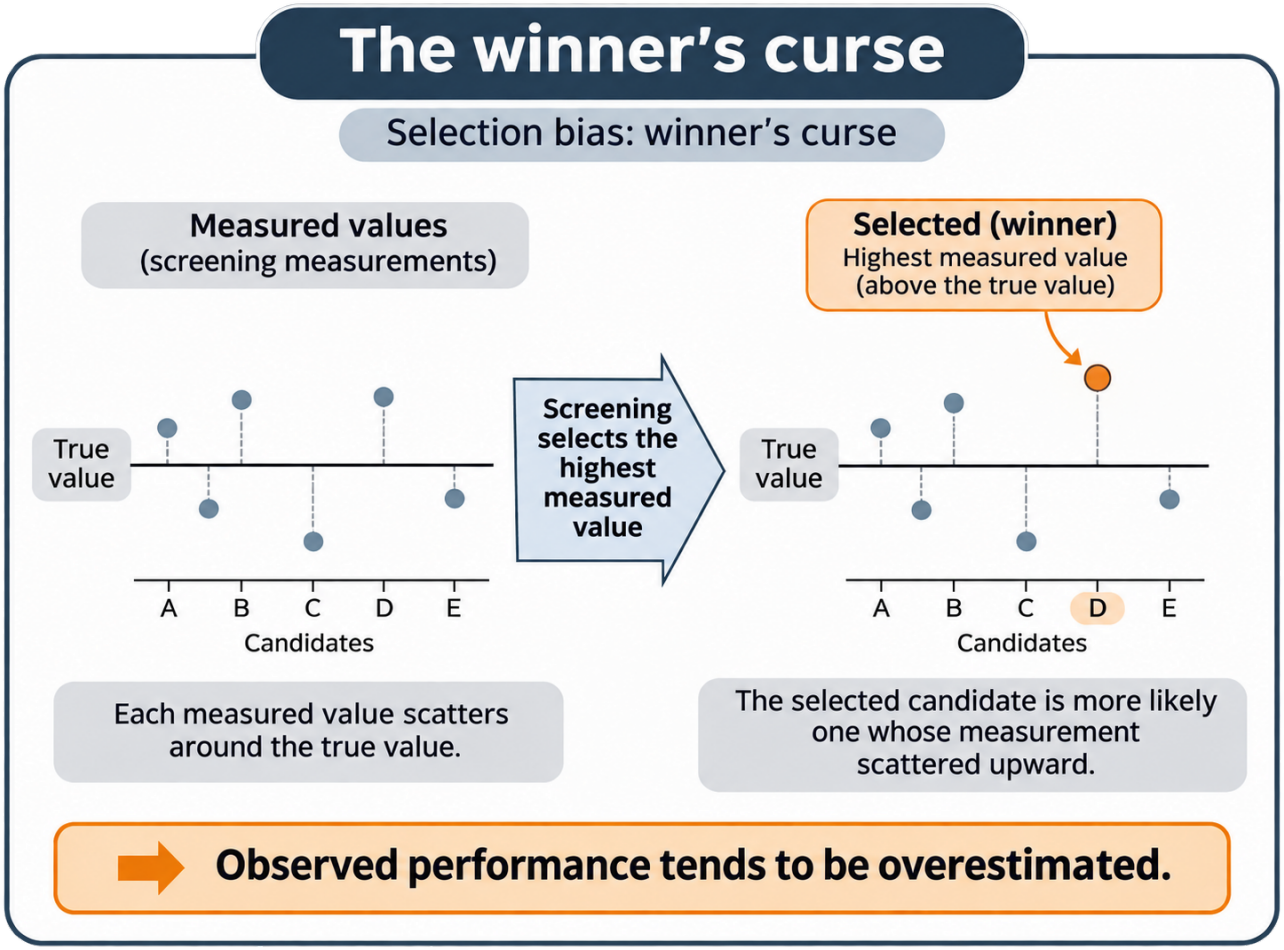
© 2026 · P. Hollenbrock

That requires separate verification of the performance metric, independent sampling units, resampling procedure, interval estimation method, and empirical coverage. These decisions should be fixed before confirmatory validation.

Where the required statistical expertise is unavailable, independent methodological review is necessary before presenting the results as confirmed empirical findings.

Appendix 2: The Winner's Curse

Each measured value by chance scatters around the true value. Screening selects the condition with the highest measured value, which is more likely one whose measurement happened to scatter upwards. As a result, the selected candidate's observed performance tends to be overestimated — a form of selection bias known as the winner's curse.



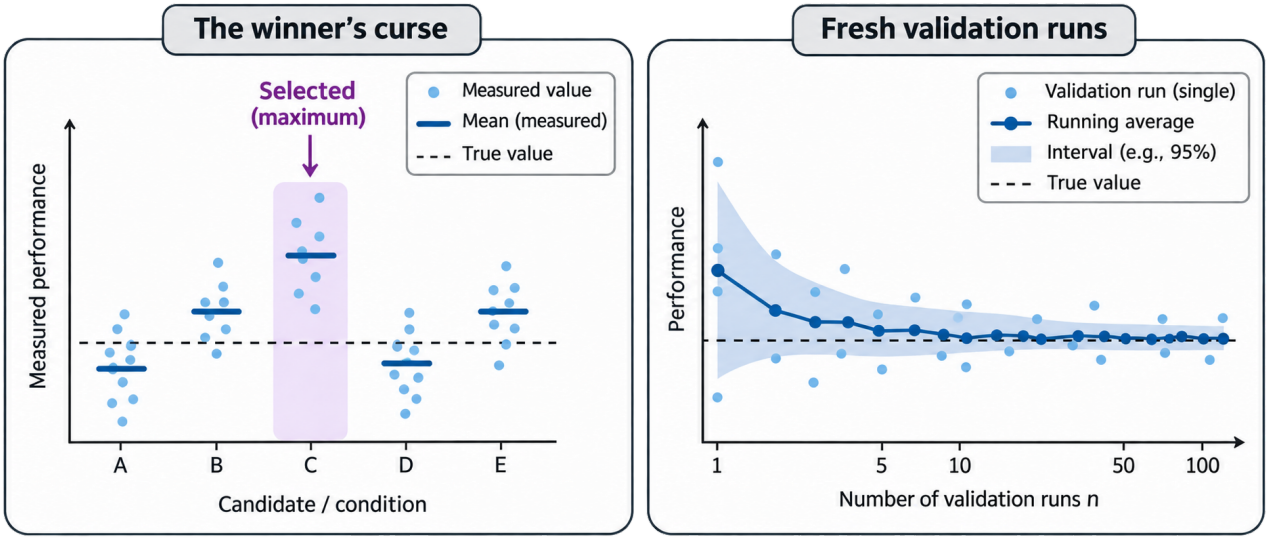
Appendix 2: Confirmatory validation runs

Because validation runs were not used for candidate selection, they provide an independent estimate of performance for a selected candidate.

Assuming unbiased measurements and appropriate sampling, the average converges toward the true value as the number of runs increases. A single run can still deviate, which is why the interval matters.

Consequently, the candidate's claimed performance can only be confirmed based on fresh validation runs.

© 2026 · P. Hollenbrock


The winner's curse
Selected
(maximum)
Measured value
Mean (measured)
True value
Measured performance
A
B
C
D
E
Candidate / condition
Fresh validation runs
Validation run (single)
Running average
Interval (e.g., 95%)
True value
Performance
1
5
10
50
100
Number of validation runs n