[Title to come]

Holden Richard
University of Vermont
hgrichar@uvm.edu

***Abstract*—[Abstract to come.]**
***Index Terms*—[Index terms to come]**

I. Introduction

The central difficulty in trusting the agentically engineered work lay in the subject's inability to comprehensively review the code due to a knowledge gap (as well as trusting that secrets and sensitive information, etc., were established and contained correctly within the environment and the code). Delegating implementation came with a burden of verification [1]. Although this knowledge gap can provide potential failure modes which could prove harmful, a well-enough-equipped system can mitigate these risks through different mechanisms (skills, hooks, checks, gates, etc.), though their presence alone does not establish that the system is safe.

The iOS app at the center of the case, Kabu, attempts to remedy the general lack of financial literacy, specifically among young to middle-aged adults, especially focusing on market actions and knowledge.

This paper attempts to conceptually replicate Davis et al. [2] from the unanswered non-expert position [3].

The original analysis states that "whether comparable outcomes could be achieved by non-experts, domain specialists, or larger teams remains an open question" [2, Sec. VIII-C].

This case analysis follows a computer science undergraduate with no prior Swift or iOS experience through his development process as well as a logging regime recording the governance processes and their evolution, some of them quantified. The case dives into how judgment and authority were exercised throughout the study, how failures were converted, and whether each conversion was seen to fire on the failure it came from.

- This study holds that a developer without prior Swift experience was able to build governance that successfully constrained agentic throughput while utilizing agents to help construct, test, and further produce the governance architecture. It should not be believed that this was without failures, as failures are actually the very foundation for conversion. The logged record shows accumulation and transfer, covering several projects where governance was established due to the original system's (Kabu) failures that were exposed and built upon.
- This study traces how the subject's role changed as implementation and verification were delegated to agents. One of the key things the subject did as the human was to steer and watch the process: not always needing to have a human in the loop is okay, but having a human overview the general process of what's happening can prove to be very important, especially on long-horizon tasks. Recorded interventions and decisions to expand or withdraw delegation show where human direction remained consequential.
- This study adds that the reach of governance, when acceptance depends on an outside gatekeeper, is bounded, as governance cannot control the outside gatekeeper's decision or know what it is grading based on: App Review publishes general rules, but case-specific knowledge is applied to each submission, and the comparison behind the 4.3(a) ruling was never named. The subject could govern rules, checks, gates, and other structures and mechanisms internally without settling the external ruling; an exploratory reading through MAGE [4] interprets this boundary through meaning, evidence and authority.

II. Background and related work

A. Neighboring work

Fawzy et al.'s survey of 162 participants reports experience-related differences in verification practices, which the authors interpret as a potential perception–action gap [5]. Deng et al.'s audit of 200 deployed applications found persistent security risks that improved harnesses and prompting did not eliminate [6]. This study follows verification in practice through evolving processes and task-specific instructions. The subject's workflow came to include agentic review and verification, with some checks delegated while device testing remained with the subject. The record traces how particular failures prompted controls and whether those controls later caught the failures they originated from. It does not establish that greater delegation produced sustained velocity gains or improved overall application security. Ferris and Bonilla [7] report a neighboring non-expert case, a healthcare platform governed by an operator without formal software-engineering training who found its tests, monitors and reviewing agents fallible.

B. The original study and governance conversion

The original study [2] looks at the process through which the engineer delegated work to agents, interpreted their outputs, and adapted the development environments over time. The study's analysis team uses two sources of evidence:

1) The case subject's notes kept over the 12-week study.
2) The engineering artifacts from the study obtained through the project's repository.

The following summary adapts the original study's process model [2, Fig. 4a]:

1) Velocity exposes failure
2) Human classifies failure
3) There is a governance conversion (e.g., agent harness is updated/made)
4) Future work is then constrained within the harness
5) Governability compounds as this process recurs more and more

Governance conversion responds to a failure class with a durable change to the architecture or controls governing later work [2, Fig. 4a]. These controls may guide agents through non-deterministic instructions, or constrain their actions through deterministic checks. This study's counting condition, C1, is defined in III-C.

A local repair fixes a specific instance of failure without necessarily changing how later development may encounter the same failure. Structural conversion addresses the foundational failure class by changing the environment or controls, which later work automatically inherits. The discernment between the two does not depend on the size or cost output of the change.

Persistent conversion is expected to compound when later work benefits from the controls previously established, though the original authors found compounding "a tendency, not strict monotonicity" [2, Sec. VI-C]. An adaptive governing structure does not ensure removal of the failure class.

C. MAGE: concepts used for interpretation

MAGE (Model-Based Agentic Software Engineering) [4] argues that as AI code generation increases implementation capacity, engineering judgment becomes relatively scarce. MAGE's core claim is that "durable engineering progress from agentic work depends not only on agentic capacity, but also on the quality of the engineering environment in which agents operate" [4, Sec. 5].

Engineering capital: MAGE defines this achieved construct as durable reusable structure that reduces the cost or uncertainty of future development processes [4, Sec. 5.2]. In MAGE, recurring judgment "becomes engineering capital when it is converted into a model, procedure, or mechanism whose expected future value exceeds the cost of constructing and maintaining it" [4, Sec. 5.2].

Semantic Gap: MAGE defines this construct as "the distance between where an engineering obligation arises and where the environment has sufficient meaning, evidence, and authority to determine whether it is satisfied" [4, Sec. 5.2].

III. Method

A. Research questions

RQ1: "Through what process can high-velocity agentic implementation be converted into governable progress?" [2, Sec. III-A]

RQ2: For a developer with no prior experience in the system's main language, how were judgment and authority at each step of governance conversion divided between developer and agent, and how, if at all, did that division change over the logged period?

RQ2a: When the platform gatekeeper holds ruling authority, how far does governance conversion reach: which of its rulings, and which failures those rulings occasioned, came under standing controls, and which received only local patches, or none?

RQ3: For each obligation that arose at the platform gatekeeper, which of MAGE's semantic-gap components (meaning, evidence, authority) were inside the developer's environment, and which were not?

B. Study design, boundary and methodological comparison

We examine the questions through a first-person case study [8]. The case in question is the subject's process across all agent-built work throughout the case window of 30 weeks and a logged case study window over 14 weeks (see Table I).

This paper approaches the conceptual replication of the original study from the unanswered non-expert software engineering perspective [2, Sec. VIII-C], as the original Subject "has sixteen years of professional experience" [2, Sec. III-D], while this case's subject is a fourth-year undergraduate university student. The original paper studies a solo expert subject who records contemporaneous notes which have been interpreted retrospectively by the co-authoring team along with the subject [2, Secs. III-B, VIII-B]. In this examination, the subject acts as the sole human interpreter, leveraging agents to help draft session logs and ledger rows, as well as to synthesize a large amount of the research regime's data, while the subject approved, corrected, and interpreted their output. This allows the subject to see how decisions are made in real time, as well as insight into the collaboratively produced engineering artifacts, which can be difficult to reconstruct after the fact.

C. Data, units, conversion condition and evidence gaps

This study draws on data from multiple different sources acquired and established through the subject's root environment research logging regime. All sources used to draw data from while synthesizing this study are as written:

- Session logs (171 indexed), including session goal, what occurred, decisions changed and why, rows written, rules created, register surfaces touched, verification lines, and human corrections;
- the ledger (628 rows, one per defect or near-miss, 18 columns, controlled vocabularies), each live row linked to its session log, and a Python script that checks integrity of each written row before every commit and push;
- the attestation register, which holds 7 promises at the freeze, also containing the gatekeeper log (13 rows: 9 events, 4 corrections or additions);
- the agent's transcripts: kept private, used for aggregate counts in the registered analysis (P4g), as well as for private-source checks;
- lastly, the subject's own dictated notes, kept verbatim.

This study anchors the ledger rows as the unit of analysis (the original study's unit of analysis is a recorded incident built into an analytic memo [2, Secs. III-C, V-A]). This offers a selection-by-rule process, compared to the original study's

TABLE I
COMPARISON OF THE ORIGINAL STUDY AND THIS STUDY

Dimension	The original study	This study
Case	One expert; 16 years professional experience; 12-week case	One senior CS undergraduate; no prior Swift; 30-week case / 14-week record
Record	88 contemporaneous field notes; repository history	628 ledger rows / 171 indexed sessions; 490 rows logged same-turn; 17 retro rows
Analysis	The original Subject + critical author team; LLM instruments; second author recoded 10 incidents	One human interpreter + agent instruments; blind P4a/P4c coding against sealed agent codes; 24 claim bins
Registration	Not reported	P4a–P4d 08-18, pre-analysis; P4e/g/h/i 09-30, pre-computation; P4f 10-09, with premature-computation disclosure DEF-593; four 08-02 entries: one to be scored at the essay freeze, one partial, two not scored
Reliability	10/10 incident class; 6/7 category; 5/7 third coding layer	P4a blind: 16/68, κ –0.07; P4c blind: 35/70, κ 0.20
Data	Mechanism catalog, Table II; field-note availability not stated	Protocol DOI; dataset release subject to redaction / withholding; agent transcripts private
Tools	Anthropic Claude; "Claude for VSCode"; model count unknown	Claude Code / desktop; claude.ai planner; nine Claude models by trailer; cross-model Codex/Gemini audits; Cursor/Grok

practiced selection by salience. A row is written in the ledger during the same turn the failure is caught. This is recorded across 18 regimed fields, including how it was caught, how far it got, how it was fixed, and what knowledge was needed to catch the failure class. There are three sub-units of these rows:

- the session during which it was found;
- the declared promise (the attestation surface) it touched;
- the platform event that exposed it.

This study registers a conversion once its control has been observed to fire on the failure it originated from (condition C1). C1 is also a condition on the original study's third step, governance conversion [2, Fig. 4a]. The original study's proposed conversion loop, shown in Section II-B, suggests how implementation can become sustainable progress; whether that progress occurred in this case must be measured against the record rather than presumed from the number of rules accumulated.

The logs began once the research regime did, which leaves the first 16 weeks without coinciding logs; their data is presented in Git, the private claude.ai data archive, and available Claude Code transcripts. There are at least 14 known unlogged sessions never reconstructed (sessions ruled no log needed; drafts never approved; transcripts never drafted). 17 ledger rows were written in later sessions, reconstructed from agent memory and documents, logging failures from before the research regime; these failures are marked "retro" and synthesized separately from live rows, since back-filled rows favor defects that were remembered or documented.

Reach is measured on a ranked scale: none, local patch, standing human control, machine gate. A level records what kind of control the ruling produced, not whether it works. This is shown only once the control catches the failure it originated from (C1).

D. Registered analyses and exploratory interpretation

This study provides deductive and pre-registered analysis, while the original study's analysis is inductive [2, Sec. III-C]. A registration is defined as an entry into the README, which defines the measure, its population, and the expected result before any data is calculated.

P4e, P4f, and P4g attempt to mirror MAGE's churn, model maintenance, and reconstruction cost measurements, respectively [4, Sec. 7.2]; P4h and P4i work to mirror the measures that the original authors propose [2, Sec. VI-C] for the original study's own propositions.

E. Agent instruments, researcher roles and reflexivity

The developer in the case is referred to as the subject: a computer science undergraduate (BS) at a U.S. university, beginning a master's in complex systems and data science, who is also the examiner of his own case study. The case covers all of his work under the regime, with examples identified as Kabu or research-record work where results are used. Any outside influence originated from the CRWT 700 course instructor's mentorship, along with Professor Davis's recommendation to frame the study as a replication.

Agents draft every rule and row, while approval remains with the subject. The subject reviews, edits, and approves them case by case, a show-before-write procedure, with recorded exceptions where drafts were written without prior subject approval or approval became a standing veto (often during long-horizon tasks). The subject specifically notes that he only skims through long drafts, rarely edits, and entrusts the governance architecture to uphold the integrity and accuracy of the regime and its contained data.

IV. CASE AND DEVELOPMENT CONTEXT

A. System, development experience and wider regime

The system analyzed in this research was initially developed as part of a Spring 2026 independent study around agentic engineering.

Kabu was the subject's main project during this period, with other projects and the research record drawing on its resources and the lessons learned during its development. In caselaw (subsequent project), the governance process was installed near the start; its rules continued to develop from incidents. Reprompt, a later project that improves dictated prompts, used mutation-tested regression tests and the logging regime.

Creating an iOS app means answering to the platform's gatekeeper, in this case, Apple App Review. An iOS application only reaches the App Store and users through App Review, which checks applications against Apple's App Review guidelines [9] before distribution to the App Store is allowed. Kabu's requirements can be separated into two parts: 1. What guidelines require of it (e.g., 4.3(a) on spam, 5.1.1 on data collection). 2. What it declares true about itself, with the promises thoroughly tracked within the attestation register. The platform was chosen to develop a longtime interest in the distribution system into a formidable deployment exercise.

B. Development period

The subject built Kabu part-time in the Spring of 2026, secondary to coursework, then continued full-time from summer through a fall semester abroad. A part-time course project became, by July 2026, the subject of its own research record. App commits fall on 62 of the 194 days between the first commit and the last, most densely in June (13 days) and July (15 days).

Velocity is defined in the original study as the "rate at which agents generate, modify, repair, test, and document software under human direction" [2, Fig. 4b]. This case also considers agent output relative to the subject's capacity to judge it. This is a qualitative interpretation of output and recorded review practices, not a rate inferred from observed failures.

C. Tools, orchestration and verification boundary

The subject worked with Anthropic's Claude throughout: Anthropic's Claude was the primary builder of the system of study (426 of 451 commits carry a Claude trailer). Commit trailers document nine Claude models used: Opus 4.6, 4.7, 4.8, 5, and 5.5; Sonnet 4.6 and 5; Fable 5 and 5.1. The first build was pasted from a claude.ai chat into Xcode on 03-20; by 03-25, the builder was Claude Code in the terminal. Planning continued in claude.ai, which could not inspect the repository directly. The subject pasted build prompts into the builder and returned each slice's output to the planner for audit. This resulted in a thorough but slow, copy-heavy iteration process, as the subject carried every handoff between agents. This did not last: by late June, planning, building, and any other necessary agentic roles had moved into their own optimized environment within Claude Code's desktop app, although the claude.ai chat stayed present as a research aid until mid-August.

Cross-model review with Gemini and Codex was recorded on 08-10; Cursor/Grok joined as a builder and reviewer from 09-25. No human read of application code is recorded in any log; relying on governance by rules, gates, and the register.

D. System state at the freeze

At the October 16 freeze, Kabu had not been accepted for App Store distribution. Build 5 had been rejected under Guideline 4.3(a) on September 30; version 1.1 remained held, and builds 6 and 7 had not been uploaded to TestFlight.

V. Findings

A. RQ1: governance conversion in this case

1) Velocity, review and failure exposure: Errors are often caught within agentic velocity, allowing the subjects to identify both the architectural gap and the knowledge gap within their own environment and/or governance and/or harness; as the original study proposes, agentic velocity creates a much larger throughput than conventional methods, where failure classes surface much slower [2, Sec. VII-B].

This case study tested this. A content pipeline agent caught a contradiction in Kabu's paper trading threshold while correcting a marketing claim. The threshold was typed in seven places, and only one was right. On 08-03, the same day it fixed those errors, the fixing agent typed a new copy. The recurrence was found four days later by pure chance.

The subject's raw output velocity is much slower, as he kept a human in the loop, matching speed to his ability to judge; this helped build a stronger foundation of understanding of both the codebase and the environment. While the record shows failures arriving close together, it cannot prove velocity caused them. Like the original study, this first-person case study supports the premise, though it is unable to quantify velocity models [2, Sec. VI-C].

The subject's Kabu app repository logged just 451 commits by the study's numbers freeze, compared to 18,662 commits in the original study's repository [2, Sec. III-C]. Across his roughly 30-week development window, compared to their 12 weeks, this raw weekly gap is even wider; this stems partly from his workflow of approving every output before committing, and partly from building the app part-time around his spring coursework.

2) Classification and proposed responses:

- During Kabu's pre-submission audit, the workflow was killed by a spend limit after 45 of 71 agents finished. The agent proposed to synthesize the found data and report the workflow as finished; the subject caught this and ruled otherwise, as only 63% of the workflow's agents had run. The resumed run went on to refute 16 findings.

The human absolutely classifies error, especially on human checks and passes, though in the subject's view, the same holds for the agent. The agent often caught many of its own errors, and the agent would also often correct the subject's errors (agents caught 490 of the 628 ledger rows; a human caught 44), as this represents a significant advantage of a non-expert using agentic engineering practices.

3) Conversion into standing controls: A strong environment's foundation relies on converting repeating, preventable failure classes into compounding governance. This paper shares the original study's view on the "failure as discovery" [2, Sec. VII-A] loop, but from a novice perspective; this study's subject applies this same step to different targets. While the original Subject converted some failures into his agent harness, his targets were mostly deterministic (lints, types, and models) built into the product. This study's conversions also generally reached a gate (54 of 78), but majority of

those gates landed outside the app: 24 are regression tests in Reprompt and 14 are checks on the study's own record. In the app's own repository, written rules outnumber gates, 10 to 3. Not everything in the regime came from failure, though: the logging regime came from the study itself, and the register was created on August 2 in anticipation of App Review, rather than in response to a failure or rejection.

Origin of these rules: the accumulated rules were mostly actioned into place through found failures throughout development; few rules were importable at all. Under the strictest test (a source published before the project began that states the rule itself, not a broader principle), 8 of 68 rules had a published source, five of which the subject was unaware of. Under the agent's broader reading, which also accepted general principles, 50 of 68 had one.

The first recorded conversion was the config-identity gate: a check that a new key is present in the config file before a swap occurs. This gate was implemented nine days into the logging regime, but 121 days after Kabu's first commit. Since nothing recorded conversions before the regime, part of that gap is a limit of the record. The rest can plausibly be attributed to factors consistent with the subject's non-expert background, such as general unawareness of governance and slower velocity and workflow.

4) Binding controls and observed catches: As the agentic governance increases and the environment becomes more well-equipped, later agents' work is increasingly caught by the checks that earlier failures left behind. Examples from this study include:

- In Kabu, a builder agent reported a change done without building it; the full-suite build failed before commit (DEF-526).

5) Compounding, transfer and recurrence: Like the original Subject, who applied the case's lessons "as ex-ante governance on two subsequent projects" [2, Sec. VII-B], this study finds that ex-ante lessons applied on later projects after establishing a foundation with Kabu can result in a stronger structural baseline in governance.

The logged data show failures being converted into controls that later constrained agent work, aligning with the replicated study's five-step loop [2, Fig. 4a]. In the subject's experience, this process helped him establish governance for later systems. Satisfying expectation P4h, the median binding lag for 65 of 75 verifiable rules was zero days and zero sessions. In caselaw, a CLAUDE.md appeared within one day of the first commit, compared with about eight weeks for Kabu; ex post was still needed, though: in another project called Reprompt, the subject's public privacy claim was found wrong the day after it was published. Despite these improvements, structural failures continued: a guardrail constructed after a ninth two-writer collision partially failed on the tenth, only being caught by a pre-commit integrity gate. Due to P4e, P4f, and P4i failing to meet expectations, while P4g was mixed, the study substantiates accelerated rule binding while leaving sustained velocity gain and economic return lacking validation.

B. RQ2: judgment, authorization, ratification and veto

What held differently? Since the subject is not an expert, the amateur perspective:

1) Replacing recognition with protocol: the human intervened and corrected the agent, and wrote little of the session logs himself. In fact, only 51 out of 700 were human verification lines.
2) The subject found himself trusting the agent's session logs (assuming the session was within reachable context) and not usually fully inspecting or amending the logs (the subject changed at least 30 of 156 drafts) unless he felt the session report had missed something it might have never been able to reach in the first place (e.g., the subject's or the system's goal, external pressure, etc.). Although the subject feels an expert may show similar trust in a competent model, he, as the amateur engineer, is completely unaware of the unknown unknowns. Therefore, the subject's general ability to declare which properties can be made answerable is much less than that of an expert.
3) Human checks shared the machines' blind spots. A copy check passed while reading zero files (DEF-056); likewise, 4 device passes by the subject cleared iPad, and then a systematic drive found the tab bar doubled on every screen, a major UI/UX issue (DEF-175, DEF-177). This was a coverage failure. An iOS developer may have known where to look.

The subject's original thought was that the human must always be the admin in complete control, directing and orchestrating agents; real governance is not possible with a purely agentic loop or no human. Although this is partly true, the subject's view on this has evolved. At first, yes, a human is required to implement governance conversion into the architecture, but once a strong base or regime is established, (rules, processes, or methodologies, etc.), the agent can often compound and implement/check and find governance conversion within an environment.

The division of work changed in both directions over the logged period. Agents took on more execution, checking, and day-to-day judgment, while the subject retained authority to ratify rules, approve restricted actions and veto recommendations. Simulator verification was delegated on 07-23, while device passes remained with the subject. Firebase reads were delegated on 08-08, while writes still required per-action approval. In the subject's account, delegated reads reduced the time he spent retrieving information for the agent.

On 08-18, the subject ratified the agent-proposed P1–P8 together. From 09-09 onward, recorded exceptions allowed logs to be written before being shown. The subject associates these exceptions with smaller tasks or long-horizon tasks where the subject was less present. Correspondence drafting returned to the subject under the 09-23 rule, reversing the established delegation. On 09-25, reversible work proceeded under standing authorization and the subject's veto, reducing

approval round trips; irreversible, external or legal steps still required his input.

These changes document particular delegations rather than establishing that accumulated rules caused them; the logged record cannot always distinguish independent deciding from accepting recommendations.

In the subject's account, the agent has suggested many of the governance rules that are established within the subject's regime. At the conclusion of this study, the subject can confidently say he would not have proposed or even noticed many of the resolutions, upgrades, and positional changes that have been implemented into this regime as it stands.

C. RQ2a: observed reach at the gatekeeper

1) Evidence inside: declarations and the false-signing episode: The subject's second submission was rejected, with reasoning alleging resemblance to other apps from a terminated developer. A reply in the review thread the same day said the subject could supply a 6-month signed Git commit history, a claim pulled from an agent's contextual memory ("all recent commits report verified=true") and reused without re-deriving it. This app review troubleshooting often felt like something the agent would handle more competently than the non-expert subject. On 08-16, an agent run check found 345 of 347 unsigned app commits; 9 minutes later the necessary correction was made.

The failure led to P3, the regime's assertion-discipline rule, ratified on 08-18. P3 requires claims to be re-derived from primary evidence when made. Its catches concern research-record claims; operation on a Kabu declaration to Apple remains unestablished. The rule's existence alone does not establish engineering capital or a closed semantic gap.

A developer's declarations about an agentically engineered app are a governed surface of their own [10]: their failures break nothing inside of the app build. Therefore, the loop detects them only when a declaration is re-derived from the product.

When the developer did not write the code, statements or claims about it are carried rather than known to be true. These must be re-derived from the product each time they leave the project, causing a human-review bottleneck [2, Sec. VI-B].

It would weaken this argument if ordinary code checks also caught the declaration failures described here. The record does not separately count code and declaration failures, therefore it does not establish a difference in their rates or demonstrate what would happen without P3.

2) Public knowledge and case-specific meaning: Publicly stated requirements must be distinguished from information held only by the platform. Published guidance that the subject did not know would indicate a knowledge gap; case-specific meaning (unavailable inside the project) is interpreted through MAGE's semantic gap [4, Sec. 5.2].

3) Withheld comparison basis and local patches: On 09-02, App Review retained the 4.3(a) ruling but reframed it: the terminated-account reference was absent, and the matched apps remained unnamed. The six code repairs were included in the 09-17 resubmission; only the 09-30 platform ruling followed those repairs.

Governance conversion held for a gatekeeper's ruling only when the ruling made known a basis or requirement that the developer could act on inside his own system or project.

App Review gave the subject, a solo developer, both rulings that make a basis known and rulings that don't. Evidence includes age rating, EULA link, as well as the first 4.3(a) ruling claiming the subject's app resembles a terminated developer account's apps, without naming them. A ruling that names nothing but still yields a standing control that later catches its originating failure would contradict this argument.

This study categorizes how far governance conversion reached when App Review held ruling authority. The 08-13 rejection over the age-rating answer and missing Terms link (2.3.6 and 3.1.2) led to standing human checklist items. Whether those items later caught their originating failures is unknown. The later 4.3(a) ruling led to local patches, including six code repairs included in the 09-17 resubmission. App Review did not name the matched apps, leaving the subject without the comparison basis needed to evaluate the finding internally.

Failures exposed while responding to the rulings produced additional controls. Missing asset-generation records led to a retention rule, while the ledger's inability to classify an external gatekeeper led to an enforced gatekeeper vocabulary. These controls govern the developer's evidence and research record, rather than resolve the platform's rulings. The rulings therefore reached checklists or local patches, while failures arising in their handling led to standing rules and an internal machine gate. No ruling itself produced a machine gate.

D. RQ3: interpretation through the semantic-gap components

The final research question interprets meaning, evidence and authority through three selected cases using MAGE's semantic gap [4, Sec. 5.2]. In DEF-216, the obligation was to validate the signing-history claim. The repository evidence was available internally, and the subject could require its re-derivation. P3's demonstrated catches concern research-record claims; its operation on Kabu declarations to Apple remains open.

DEF-221, the age-rating/account-type restriction, shows that checking the app's content missed what the developer was able to declare. Guideline 5.1.1(ix) [9], published before the 08-13 rejection, asks that gambling apps come from a legal entity rather than an individual; its application to simulated gambling through the age-rating answer is not stated in Apple's Guidelines, so whether only the platform held that meaning stays unknown. In 4.3(a), the matched apps were unnamed, so the comparison could not be reconstructed internally and App Review retained authority over acceptance, while the subject could still govern his own evidence and responses.

This reading is exploratory: it uses the same evidence as RQ2a and does not independently confirm its findings. A comparable case with more relevant meaning, evidence or authority outside the environment nevertheless reaching

further would challenge the proposed ordering. Three selected cases cannot establish a causal gradient.

VI. Comparison with the original study

The original study describes a "deterministic control envelope" [2, Sec. VII-D]; this study holds it differently: the recorded evidence states that some deterministic checks were a failure, a failure source that looked like success (DEF-056, DEF-101).

Individually, neither study establishes the developer's expertise as a determining parameter.

MAGE's authors write that "differences among developers, e.g., in expertise and motivation, confounded experiments: human factors often dominate the observable effects of the engineering mechanisms under study" [4, Sec. 8].

Expertise may determine whether governance relies on recognition or failure, not whether it's needed.

This is the question that this case examines. In response to the non-expert angle, this study's scope does not reach teams, where the original analysis expects governance conversion "would likely require organizational processes" [2, Sec. VIII-C]. The original study holds that "as implementation becomes abundant, judgment becomes relatively scarce" [2, Sec. VI-B].

Neither study establishes an external gatekeeper as a determining parameter. The original study reports no outside ruling on the product in its 12 weeks. Here, one ruled 4 times.

Across the two cases, failures prompted changes that constrained later work, consistent with the original study's proposed loop [2, Fig. 4a]. In the subject's blind reading of 24 claims, 14 held the same, eight differently but consistently, one inconsistently, and one was not reached. These are interpretive judgments about the claims rather than independent tests of their effects. In the blind reading, the one inconsistent claim was the deterministic control envelope, and the one not reached was the marginal cost of quality.

VII. Discussion

The objective of this paper aims to illustrate a more clear idea of what writing code with an agent entails, what kind of responsibilities the engineer (should) take on, and what spectrum of authority and answerability the engineer gives away in turn for agentic velocity. The general practices goal can be simplified: responsibly and effectively onboarding an agent rather than simply offloading and eventually outsourcing to the agent. Accepting an agent's recommended default can become a risk when authorization to proceed is mistaken for evidence that the subject independently verified the work. This record does not measure the frequency or speed of such acceptance.

VIII. Threats to validity

A. Internal validity

Retrospectively evaluating one's own work can offer deep, direct access to rationales that external methods may miss, but in turn, this increases the risks of subjective overinterpretation as well as confirmation bias [2, Sec. VIII-B]. Due to this study lacking an independent team of reviewers, its counts are tied directly to verifiable repository data and notes recorded throughout the development process; most were contemporaneous (490 of the 628 frozen rows were written in the same turn), which bounds the threat without removing it.

The limit to this is that the record struggles to differentiate drafting from deciding.

Agents also drafted rows about their own failures. Of 700 verification lines, 51 named a human or human-plus-agent actor; no human reading of application code was recorded. The subject changed the agent-drafted record in at least 30 of 156 v2 sessions, but that is a floor on recorded intervention, not a measure of independent judgment.

The subject's logging regime only exists because of this very study itself. In his 10-12 notes, the subject wrote that the process "resulted in a regime that quite literally tests the theories it was set up to record"; under Wohlin and Rainer's case-study definition, that places the study nearer action research [11]. The regime's gates create some of the catches that the study counts. Changing logging intensity also limits comparisons over time.

Session logging ended on October 22, before the planned end; the ledger and register continue to the essay freeze, and no frozen number changes.

The models, tools, task mix and governance procedures changed together during the case. Nine Claude model versions appear in app commit trailers; context compaction and changing task mix were registered confounds. The record cannot isolate the effect of stronger governance from those changes, thus comparisons over time do not establish a causal gain in velocity or reliability.

B. External validity

This study relies on an n = 1 design with a single subject who assumes both roles as the sole human developer and evaluator within a specific agentic development environment. This creates a risk that the resulting observations are artifacts of the subject's development style and the chosen software ecosystem (Apple App Review as the platform's gatekeeper; Anthropic's Claude as the primary builder), rather than generalizable patterns [2, Sec. VIII-B]. The study's findings identify the "human-agent capability fit" [2, Sec. VIII-C] from the perspective of a senior undergraduate who has gone to school in the period of widespread AI flourishing and adoption, and who had used AI coding agents before 2026: his inexperience was with Swift and iOS, not with agents. This case study aims at theoretical rather than statistical generalization [8], [12], [13]; the limits above stand regardless.

IX. Ethics and availability

Like the original study, this study involved self-experimentation by its author. The subject self-sponsored and formed TISL LLC, taking ownership for the app studied as his product; there was no provided salary or grant for this study, nor an attempt made at one. The original study publishes a description of its governed engineering environment, including

its controls [2, Sec. X]. The study's protocol is archived under DOI 10.5281/zenodo.21987310. This study publishes its frozen analyses, scripts and count-level data under DOI [results DOI], and the dataset under a reserved DOI [dataset DOI] after a redaction check at publication, as well as keeping the agent transcripts private. The published dataset details the subject and agents' session logs, the ledger, the attestation register and its gatekeeper log, the pre-registrations, the coding sheets with both codings, and the analyses; the frozen analyses use the full private record, which may differ from the released subset. The agent transcripts stay out, along with used sources (cited, not redistributed), secrets, testers' emails, and any other sensitive or irrelevant information. People outside the study appear by role only; App Review's messages appear as guideline numbers and paraphrases; secrets appear only by their last four characters, if at all.

X. Conclusion

The original study's process held in a case toward the other end of the experience range [2, Fig. 4a], with three notable differences. Conversion came from failure more than published practice; the agents drafted the classifications, and the subject approved them; a conversion counted only once its control was seen to fire, and the record leaves sustained velocity gains and economic return unestablished. The agents proposed much of the governance once a foundation was established, and the subject ruled on these proposals, though the record cannot always tell his own decision from his acceptance of theirs. The extent to which conversion reached depended on how much of a ruling's basis was known: App Review gave the subject both rulings that make a basis known and rulings that don't, and no ruling itself produced a machine gate; the MAGE reading of this boundary is exploratory and rests on the same evidence. Generative AI makes code cheap, but this case suggests that new developers' enablement comes from building the boundaries which keep autonomous systems answerable.

Acknowledgment

Claude Code and OpenAI Codex assisted with evidence assembly, source checking, suggesting revised wording for the author's drafts, manuscript organization, proofreading, syntax refinement, and standardizing formatting; the author dictated, drafted, edited, and revised everything. Claude Code also drafted research records and produced the sealed P4a/P4c codings; agents assisted with computation. Section III-E describes these contributions.

References

[1] S. E. Farrag and Y. Saber, "Where accountability lives: Mapping human responsibility to workflow artifacts in agentic software development," arXiv:2608.15678v2 [cs.SE], Sep. 2026.

[2] J. C. Davis, P. C. Amusuo, T. Singla, B. Çakar, and K. A. Davis, "Cheap code, costly judgment: A case study on governable agentic software engineering," arXiv:2607.01087v2 [cs.SE], Jul. 2026.

[3] ACM SIGSOFT, "Replication," ACM SIGSOFT Empirical Standards for Software Engineering. Accessed: Oct. 28, 2026. [Online]. Available: https://www2.sigsoft.org/EmpiricalStandards/docs/standards?standard=Replication

[4] J. C. Davis, K. G. Kalu, H. Peng, and P. V. Patil, "Model-based agentic software engineering," arXiv:2608.25174v1 [cs.SE], Aug. 2026.

[5] A. Fawzy, A. Tahir, and K. Blincoe, "From prompting to verification: How experience shapes vibe coding practices," arXiv:2605.24521v1 [cs.SE], May 2026.

[6] J. Deng, Z. Fan, and R. Meng, "Understanding the (in)security of vibe-coded applications," arXiv:2606.23130v4 [cs.CR], Sep. 2026.

[7] L. Ferris and S. Bonilla, "A case study in assuring AI-written software," arXiv:2610.08651 [cs.SE], Oct. 2026.

[8] P. Runeson and M. Höst, "Guidelines for conducting and reporting case study research in software engineering," Empirical Softw. Eng., vol. 14, no. 2, pp. 131–164, 2009, doi: 10.1007/s10664-008-9102-8.

[9] Apple Inc., "App Review Guidelines," Apple Developer. Accessed: Oct. 14, 2026. [Online]. Available: https://developer.apple.com/app-store/review/guidelines/

[10] T. Li, K. Reiman, Y. Agarwal, L. F. Cranor, and J. I. Hong, "Understanding challenges for developers to create accurate privacy nutrition labels," in Proc. CHI Conf. Human Factors Comput. Syst. (CHI '22), New Orleans, LA, USA, 2022, doi: 10.1145/3491102.3502012.

[11] C. Wohlin and A. Rainer, "Is it a case study?—A critical analysis and guidance," J. Syst. Softw., vol. 192, Art. no. 111395, 2022.

[12] ACM SIGSOFT, "Case study and ethnography," ACM SIGSOFT Empirical Standards for Software Engineering. Accessed: Oct. 27, 2026. [Online]. Available: https://www2.sigsoft.org/EmpiricalStandards/docs/standards?standard=CaseStudy

[13] B. Flyvbjerg, "Five misunderstandings about case-study research," Qualitative Inquiry, vol. 12, no. 2, pp. 219–245, 2006.