FROM TOKEN-MAX TO OUTCOME-MAX: HOW YOU USE AI DETERMINES ITS PRODUCTIVITY

Anonymous authors
Paper under double-blind review

ABSTRACT

Generative artificial intelligence (AI) models can perform increasingly complex tasks, yet greater AI usage does not necessarily translate into proportional productivity gains. We identify token-max as one source of this inefficiency: when token consumption is treated as productive effort, agents are encouraged to over-exert and expend computation beyond what is necessary. We instead propose *outcome-max*, which rewards independently verified task completion per unit cost and induces a principled stopping rule. Then, to study these objectives, we develop a three-level simulation framework spanning immediate interaction, long-run behavioral adaptation, and organizational collaboration. Across all three levels, outcome-max improves the efficiency of AI-assisted production while largely preserving verified task performance. To further align these incentives with outcome-max, we introduce *OutcomeShare*, an incentive mechanism. Theory and simulation show that OutcomeShare can induce participation while generating shared gains for employees, firms, and LLM providers. Together, our results suggest that AI productivity not just depends on model capability, but also on the objectives governing AI use.

1 INTRODUCTION

Generative artificial intelligence (AI) can improve productivity in software development (Cui et al., 2026) and professional writing (Noy & Zhang, 2023). Yet realizing these gains requires balancing greater adoption with its costs. New reports of token-usage leaderboards (Roose, 2026) and Meta's plans to limit employee token spending (Mann, 2026) reveal a tension between encouraging use and controlling costs: *why can greater AI usage fail to improve productivity?*

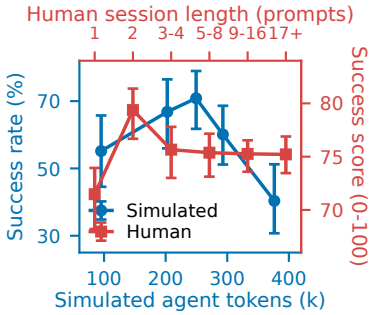


Figure 1: **More tokens need not improve success.** Blue: simulated token-max on SWE-bench. Red: real human coding sessions from SWE-chat.

We argue that one source of this inefficiency is the *token-max* policy, which treats greater token consumption as evidence of productive effort. Yet more tokens do not necessarily produce better outcomes. Figure 1 illustrates this pattern in both our simulations and real developer sessions: success initially increases but then plateaus or declines as interactions grow longer. The reason is due to *over-efforting* effect: excessive interaction expands the reasoning context, encourages unnecessary and redundant checking and revision.

These observations motivate an alternative objective. Inspired by economic theory for rewarding inputs versus rewarding outcomes Lazear (2000), we propose *outcome-max*, which rewards independently verified task completion per unit cost. Under this objective, further computation is justified when its expected marginal return exceeds the return achieved so far.

To evaluate the effectiveness of our proposed objective, we develop a three-level simulation framework spanning single-task execution, textual-memory adaptation, and employee selection. We simulate token-max and outcome-max behavior using LLM-based agents explicitly instructed to either maximize token use or achieve correct results at minimum cost. The results show that the outcome-max policy substantially reduces token costs while even slightly improving overall verified task performance.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

To further incentivize outcome-max behavior, we design *OutcomeShare*, a mechanism that combines verified-outcome payments with liability for defective delivery. Intuitively, it shares the value of verified outcomes to reward employees for avoiding wasteful effort, while imposing penalties for defective delivery to discourage them from misrepresenting unsuccessful work. Through theory and scenario accounts, we examine such a mechanism can align provider, employee, and firm incentives while creating shared gains. This motivates us not only to develop more capable AI models, but also to design mechanisms that guide how people use them effectively.

We make three contributions as follows. Code is available at `https://anonymous.4open.science/r/token-ecom-review-27FB/`.

- We identify the inefficiencies caused by rewarding token consumption and propose *outcome-max*, an alternative policy that optimizes verified production per unit cost.
- We develop a three-level simulation framework and demonstrate that outcome-max substantially reduces token costs while even slightly improving overall verified task performance.
- We introduce OutcomeShare, an incentive mechanism that aligns organizational stakeholders around the adoption of outcome-max.

2 Related Work

In this section, we aim to connect agent computation, economic incentives, and behavioral simulation as foundations for productive AI use.

Agent computation and adaptation. ReAct combines reasoning and actions (Yao et al., 2023); MetaGPT organizes collaboration (Hong et al., 2024). Executable benchmarks assess correctness (Jimenez et al., 2024; Liu et al., 2023). Scaling and cost-aware evaluation study accuracy–compute tradeoffs (Snell et al., 2025; Kapoor et al., 2025). Overthinking can waste computation and impair agent performance (Chen et al., 2024; Cuadron et al., 2025). BAGEN studies budget awareness and early stopping (Lin et al., 2026). Reflexion adapts verbal memory (Shinn et al., 2023); ACE evolves reusable playbooks (Zhang et al., 2026). GEPA evolves prompts using execution feedback (Agrawal et al., 2026). We evaluate these decisions under specified supervision policies.

Incentives and productivity. Principal–agent theory links measurement to effort (Holmström, 1979; Holmström & Milgrom, 1991). Imperfect performance measures can distort incentives (Baker, 1992), including in multi-agent systems (Rauba et al., 2026). Recent work studies contracts for model and effort choice (Sreenivas & Larson, 2026) and usage versus success-based pricing (Lobel et al., 2026). We combine outcome payments, defect liability, and three-party participation. Workplace studies find productivity gains (Brynjolfsson et al., 2025; Cui et al., 2026). Effects vary across settings (Vaccaro et al., 2024); early-2025 tools can also slow developers (Becker et al., 2025).

Behavioral simulation. Social simulation includes generative agents and Concordia (Park et al., 2023; Vezhnevets et al., 2023). CAMEL and SOTOPIA study role-based interaction (Li et al., 2023; Zhou et al., 2024). Gao et al. (2024) survey agent-based simulation. Simulated-subject studies compare human responses (Aher et al., 2023; Argyle et al., 2023). Economic simulations study decisions (Horton, 2023) and market institutions, including liability (Erlei & Meub, 2026). We evaluate costs and verified quality conditional on specified behavior; incentive-induced behavior requires a separate test.

3 Framework: From Token-Max to Outcome-Max

In this section, we aim to formalize AI supervision around interaction volume or verified output per cost, then simulate the corresponding supervisory behaviors across execution, adaptation, and selection (Figure 2).

3.1 NOTATION AND OBSERVABLE OUTCOMES

We distinguish externally verified success y, employee acceptance a, and agent token use T. Success and acceptance are binary; a shipped defect is $d = a(1 - y)$. The external evaluator scores the final artifact even when the employee does not accept it. Employees are persona-conditioned LLMs, not human participants (Appendices A and A.6).

Providers meter tokens in four categories $r \in \{\text{in}, \text{out}, \text{cr}, \text{cw}\}$: *input* tokens the agent reads (task text, files, tool results), *output* tokens it writes (reasoning, commands, patches), *cache reads* (input served from a prompt cache at a discount), and *cache writes* (tokens newly stored to that cache). With T^r the token count in category r and p_r its price per token, compute cost and the reported token quantity T are

$$C^{\text{AI}} = \sum_r p_r T^r, \qquad T = T^{\text{in}} + T^{\text{out}}. \tag{1}$$

The cost includes all four categories; the reported token metric counts uncached input and output. Supervision cost uses an assumed wage and time proxy, so dollar payoffs depend on the scenario in Table S2. These quantities separate verified output, resource use, and financial cost.

3.2 TOKEN-MAX: OPTIMIZE THE RESOURCE METRIC

Let π denote the interaction policy under the common resource limits. Token-max rewards expected token consumption over the task distribution:

$$J_{\text{tok}}(\pi) = \mathbb{E}_\pi[T]. \tag{2}$$

Additional tokens improve this objective even without a better verified result. This objective property does not predict behavior. Our simulation implements token-max through requests for continued work; token billing motivates the archetype without establishing that billing alone produces it.

3.3 OUTCOME-MAX: OPTIMIZE VERIFIED OUTPUT PER COST

Outcome-max rewards verified task completion per unit of compute cost:

$$J_{\text{out}}(\pi) = \frac{\mathbb{E}_\pi[y]}{\mathbb{E}_\pi[C^{\text{AI}}]}, \tag{3}$$

where expected cost is positive. Additional computation is worthwhile only when its expected resolution gain justifies its cost (Appendix B). The ratio uses workload totals, not the mean of task-level efficiencies. Success and cost together determine whether more computation is productive.

3.4 SIMULATION FRAMEWORK

We simulate token-max and outcome-max as two prescribed supervisory behaviors (Figure 2). Instructions specify how the employee continues or stops; the task, models, evaluator, and resource limits are held fixed within each paired comparison. Employee personas vary supervisory behavior; neither the employee nor the production agent sees the external gold outcome during an interaction.

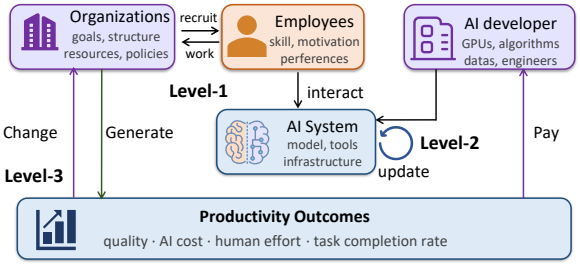


Figure 2: **Simulation framework.** L1: execution; L2: adaptation; L3: selection.

L1: execution. Employee i with objective o supervises the agent on task x_j: after each exchange t the employee accepts ($a_t = 1$) or asks for another step, until acceptance or a resource limit, $\tau = \min\{t \le \tau_{\max} : a_t = 1\}$. External evaluation checks the final artifact, $y_{ij} = E(x_j, H_{ij})$. L1 evaluates policy consequences without learning or workforce turnover.

L2: adaptation. Employee and agent retain separate textual tactic libraries L_g and K_g. After each training generation g, reflection sees only outcome feedback $F_g = \{(y, a, d, \tau, T)_{ij}\}$ and updates

$L_{g+1} = R_H(L_g, F_g; o)$ and $K_{g+1} = R_A(K_g, F_g)$, where employee reflection R_H revises tactics within the specified behavioral regime and agent reflection R_A targets verified resolution at lower cost. Feedback excludes task text, artifacts, and transcripts. Ablations freeze either library; joint adaptation updates both. Model weights stay fixed, and frozen libraries are tested on held-out tasks (Appendix A.5).

L3: selection. Each period, the firm ranks employees by average token use or verified solves per compute cost, with a bounded preference for smaller successful patches. It retains top-ranked employees and recruits their mutated descendants (Appendix A). This tests workforce composition and performance under the imposed retention rule. Together, L1–L3 evaluate policy consequences under specified update rules.

4 MECHANISM: OUTCOMESHARE

In this section, we aim to specify conditions for aligning provider, employee, and firm incentives through OutcomeShare: payments for verified work, liability for defects, and shared value.

4.1 FROM AN OBJECTIVE TO AN IMPLEMENTABLE CONTRACT

The intuition behind OutcomeShare is to share the value of verified outcomes among the provider, employee, and firm, thereby rewarding productive work rather than unnecessary token consumption. At the same time, liability for defective delivery discourages employees from approving or misrepresenting unsuccessful work as successful. The outcome-sharing and liability terms are fixed before production, and participation constraints ensure that each stakeholder benefits under the specified valuation and cost assumptions.

4.2 MECHANISM OUTCOMESHARE

Let V be value per verified output, $Q(\phi)$ weekly verified output under fee share $\phi \in [0, 1]$, $p > 0$ the effective audit probability, F the fine charged to an employee who approves a defect, and $\kappa V > 0$ the damage per defective delivery. The three payment quantities are

$$R_{\text{dev}}(\phi) = \phi V Q(\phi), \qquad \lambda = \frac{pF}{\kappa V}, \qquad pF = \kappa V \;\; (\lambda = 1). \tag{4}$$

Provider revenue is before compute costs; an employee bonus shares verified value. Normalized liability λ distinguishes no liability ($\lambda = 0$) from damage-matched liability ($\lambda = 1$), our alignment target; experiments also test other levels. Defects incur penalties without outcome payments.

Participation and fee cap. To preserve participation, compare each payoff $U_k(\phi, pF)$ with baseline U_k^0 under common valuation and cost assumptions:

$$U_k(\phi, \kappa V) \geq U_k^0, \qquad k \in \{\text{provider}, \text{firm}, \text{employee}\}. \tag{5}$$

The fee cap is the highest share satisfying all three constraints; fees only divide existing surplus.

4.3 THEORETICAL ANALYSIS

The results below characterize objective properties and verification losses (Appendix B), not LLM optimization or contract adoption.

Theorem 1 (Objective response to extra computation)**.** *Let π, π' be feasible policies on the same task distribution with positive expected cost. If π' increases expected tokens and cost while leaving positive expected verified output unchanged, then $J_{\text{tok}}(\pi') > J_{\text{tok}}(\pi)$ and $J_{\text{out}}(\pi') < J_{\text{out}}(\pi)$. More generally, for baseline output Y, cost $C > 0$, and changes ΔY and $\Delta C > 0$, outcome-max improves if and only if $\Delta Y/\Delta C > Y/C$.*

Matching expected defect costs aligns private and social verification effort.

Theorem 2 (Pigouvian verification alignment)**.** *Let verification effort $e \in \mathcal{E}$ cost $c(e)$ and induce shipped-defect probability $\delta(e)$, with external damage κV per defect. Assume a risk-neutral employee,*

a fixed effective audit probability $p > 0$, a collectible fine F, and all other private and social payoff terms constant in e. If $\mathcal{E}$ is nonempty and both losses attain their minima, then $pF = \kappa V$ implies that private loss $c(e) + pF\delta(e)$ and social loss $c(e) + \kappa V\delta(e)$ have identical minimizers.

Implications. Theorem 1 favors continuation when its marginal verified gain per dollar exceeds the current average. Hard-task losses therefore motivate better verification signals, rather than uniformly shorter work. Theorem 2 aligns verification by matching expected liability to defect damage.

4.4 From efficiency to shared gains

Efficient production creates surplus; fees and bonuses divide it. Shared gains require value sufficient to cover production, supervision, verification, and defects while satisfying participation constraints (Eq. (5)). Thus efficiency creates the opportunity, while quality control and allocation determine whether participants benefit.

5 Experiments

In this section, we aim to evaluate the consequences of token-max and outcome-max supervisory behaviors across L1–L3. RQ1–RQ3 compare their performance and sensitivity; RQ4–RQ5 evaluate delivery rules and scenario payoffs.

5.1 Experimental setting

Datasets. We evaluate externally verifiable work across software engineering, program synthesis, and mathematics. SWE-bench Verified uses repository issues with containerized gold tests (Jimenez et al., 2024; OpenAI, 2024). HumanEval+ uses programs with extended tests (Liu et al., 2023). GSM8K and MATH-500 use normalized answer comparison (Cobbe et al., 2021; Lightman et al., 2024). SWE-bench supplies the L1 execution, L2 adaptation, and L3 selection experiments; cross-task comparisons also include the other three benchmarks. We compare simulated conversation depth with WildChat-1M coding conversations (Zhao et al., 2024) (Figure S1). Appendix A gives task counts and selection rules; Appendix A.6 describes the external validation.

Metrics. For N attempted tasks, y_j is external correctness and T_j is the production agent's input-plus-output tokens. We report three complementary metrics, with efficiency measured in verified solutions per million agent tokens:

$$\text{Success} = 100\frac{\sum_j y_j}{N}, \qquad \overline{T} = \frac{\sum_j T_j}{N}, \qquad \eta = 10^6\frac{\sum_j y_j}{\sum_j T_j}. \tag{6}$$

External success is distinct from employee acceptance. *Efficiency improvement* is $100(\eta_{\text{out}}/\eta_{\text{tok}} - 1)$. This resource metric is distinct from monetary cost, which depends on model pricing and supervision. *Economic outcomes* comprise provider revenue, employee income, and firm net profit; Appendix A.5 defines their scenario accounting.

5.2 Comparing token-max and outcome-max

5.2.1 RQ1: Does outcome-max outperform token-max?

In this section, we aim to compare verified work per agent token under the two specified supervisory behaviors across the three levels (Table 1). Difficulty groups are defined independently of observed success; paired intervals appear in Appendix C.1.1.

At L1, outcome-max uses fewer tokens and improves efficiency across task families, while aggregate success remains similar or improves. The hard software and program-synthesis subsets instead show a quality cost. At L2, the efficiency advantage persists under employee-only, agent-only, and bilateral adaptation; bilateral adaptation improves success across difficulty groups, whereas agent-only adaptation trades some success for lower resource use. At L3, outcome-based selection retains more verified value at substantially lower cost, reversing the firm's average net sign.

Table 1: **Performance across three levels of human–AI simulation.** Bold marks the better point estimate within each pair; † denotes reconstructed math difficulty strata (Appendix A.1).

Setting	Objective	Success (%) ↑ Overall	Easy	Medium	Hard	Tokens (k) ↓	Efficiency gain (%) ↑
L1: single-round human–AI interaction							
SWE-bench	Token-max	**48.8**	62.2	43.9	**39.8**	21.1	0.0
	Outcome-max	48.2	**66.7**	**48.9**	25.0	**9.4**	**+120.7**
HumanEval+	Token-max	87.2	94.3	89.3	**73.8**	5.4	0.0
	Outcome-max	**87.5**	**95.9**	**91.4**	69.6	**4.7**	**+15.3**
GSM8K†	Token-max	89.8	93.9	86.5	88.5	2.4	0.0
	Outcome-max	**92.2**	**94.6**	**88.8**	**95.5**	**1.3**	**+86.5**
MATH-500†	Token-max	57.2	**80.4**	**63.0**	40.9	5.7	0.0
	Outcome-max	**57.4**	80.1	61.5	**42.1**	**4.6**	**+25.6**
L2: multi-round human–AI interaction							
Employee adapts	Token-max	57.6	72.9	**57.2**	24.2	281.0	0.0
	Outcome-max	**60.9**	**80.3**	56.7	**28.3**	**78.9**	**+276.8**
Agent adapts	Token-max	**62.4**	78.3	**59.8**	**32.8**	206.8	0.0
	Outcome-max	61.0	**78.9**	58.4	25.9	**96.0**	**+110.8**
Both adapt	Token-max	55.8	72.4	54.3	21.1	262.7	0.0
	Outcome-max	**63.0**	**84.8**	**58.1**	**26.3**	**91.5**	**+224.2**

L3: organizational human–AI collaboration				
Objective	Verified value ($) ↑	Total cost ($) ↓	Net profit ($) ↑	Normalized profit (%) ↑
Token-max	7,801.6	21,043.8	-13,242.2	-508.3
Outcome-max	**8,228.8**	**5,623.4**	**2,605.4**	**100.0**

Efficiency gains do not guarantee completion. The stopping audit links hard-task failures to premature acceptance and agent-turn limits (Table S9); matched comparisons narrow the agent-only success gap while retaining its efficiency advantage (Table S10). The neutral reference uses zero behavior propensities, no incentive directive, and no employee follow-up; it is unpaired with the adaptation arms. L1 and L2 differ in tasks and models, so their absolute rates do not isolate learning. Overall, the outcome-max policy is more efficient across levels; bilateral adaptation also supports quality, while hard tasks need safeguards.

5.2.2 RQ2: Is the objective gap robust?

In this section, we aim to assess policy sensitivity across backbones, personas, and prompt designs (Figure 3). Positive success gaps and efficiency gains favor outcome-max.



Figure 3: **Token-max versus outcome-max across policy implementations.** Blue: success gap in percentage points (left axis); gold: efficiency gain in % (right axis). "With rules" states both the objective and continuation instructions; "Only" states the objective alone. Positive gaps favor outcome-max.

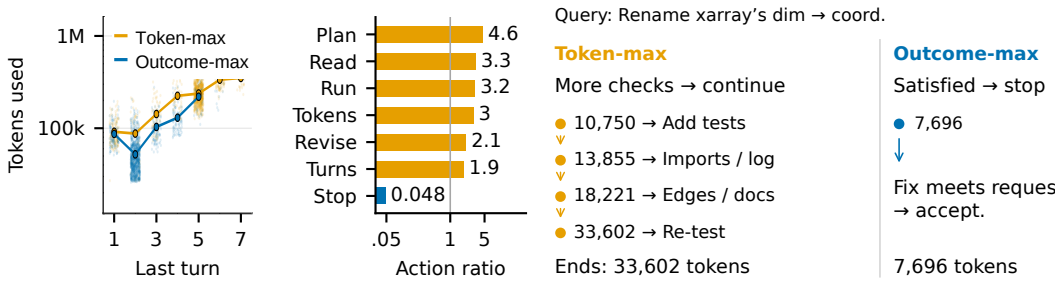
Figure 3(a,b) shows higher outcome-max efficiency across both backbones and all six personas, but success varies. Revise favors iterative correction, which may benefit from token-max's continuation rules; the uncertain success gap prevents a causal interpretation. Panel (c) shows smaller gaps when prompts state only the objective.

Objective-only prompts show a small directional efficiency gain, suggesting that objective framing may influence supervisory effort even without explicit continuation rules. Mean success is unchanged.

Appendix C.1.1 reports the intervals; model and persona differences prevent attributing the cross-setting gap solely to wording. The main comparison evaluates the richer token-max and outcome-max behaviors.

5.2.3 RQ3: WHY IS OUTCOME-MAX MORE EFFICIENT?

In this section, we aim to locate token costs in repeated actions, learned rules, and selection (Figure 4). Difficult tasks may require longer interactions, so duration alone does not identify the benefit of extra effort.



(a) Conversation cost (b) Repeated actions (c) Stopping case

Figure 4: **Where the efficiency gap arises.** (a) Mean agent tokens by the turn at which conversations end; (b) token-max/outcome-max ratio of action counts (above 1: token-max repeats more); (c) a paired xarray attempt: token-max keeps requesting checks after a passing patch; outcome-max accepts.

Token-max conversations extend to later turns and involve more planning, inspection, command execution, and revision. Outcome-max ends sooner when the employee judges the delivered work sufficient. Matched attempts also show more token-max continuation after the final correct patch is available (Appendix C.3.1). Thus repeated review contributes to the resource gap even when the final artifact is already correct.

Table 2: **Which follow-up work is wasteful, and which is useful?** Two paired xarray attempts. The first-patch tests below are external; employees cannot see them. Appendix C.3.2 gives the traces.

Objective	Employee behavior after the first patch	Verified result; total tokens
Case A: both agents' first patches already pass the task tests.		
Token-max	Requests extra tests, import/changelog checks, then edge-case and documentation work.	Four passing patches; 33,602.
Outcome-max	Accepts the first patch and ends the interaction.	One passing patch; 7,696.
Why wasteful: repeated checks and edits increase tokens without improving verified task correctness.		
Case B: both agents' first patches fail the task tests.		
Token-max	Questions backward compatibility of existing `dim=` calls, prompting a repair.	The next patch passes; 30,901.
Outcome-max	Says the fix looks good and stops without raising compatibility.	The patch still fails; 4,456.
Why useful: the additional compatibility check exposes a real defect; early acceptance misses it.		

What experience does self-evolution preserve? We examine whether adaptation preserves different supervisory priorities and relate them to human interactions (Figure 5). Panel (a) shows persistent emphasis on context, targeted fixes, and stopping under outcome-max, versus explanations, alternatives, and repeated review under token-max. Objective-conditioned reflection preserves these priorities; the libraries do not demonstrate spontaneous adoption of the objectives.

Panel (b) relates these priorities to SWE-chat user personas. Outcome-max and vague requesters tend toward shorter interactions with higher success, whereas token-max and mind changers tend toward longer interactions with lower success. The measures differ across populations, so this alignment is descriptive. Together, the panels suggest a broader implication for AI literacy: evaluate targeted problem solving and necessary verification instead of treating interaction volume as evidence of productive effort. This does not identify workplace incentive effects.



(a) Learned rules by topic (b) Human personas vs. objectives

Figure 5: **Learned experience and interaction depth.** (a) Employee playbook topics after early (light) and final (dark) learning; outcome-max emphasizes targeted work and stopping. (b) SWE-chat personas (Baumann et al., 2026) and simulated objectives compared by normalized interaction length and mean-centered success. Human ratings and simulated pass rates are distinct measures; bars show 95% cluster-bootstrap intervals.

Which employees survive, and how does profit change? We examine how selection connects supervisory styles to firm performance (Figure 6). Both firms replace the bottom three employees after each period. Token-based selection expands mode-setting and clarification lineages, while outcome-based selection preserves example-providing and missing-context lineages. Revenues remain comparable, but token-max's rising costs deepen losses while outcome-max stays profitable in the stated accounts. These results are conditional on the imposed retention rules, not voluntary policy adoption.

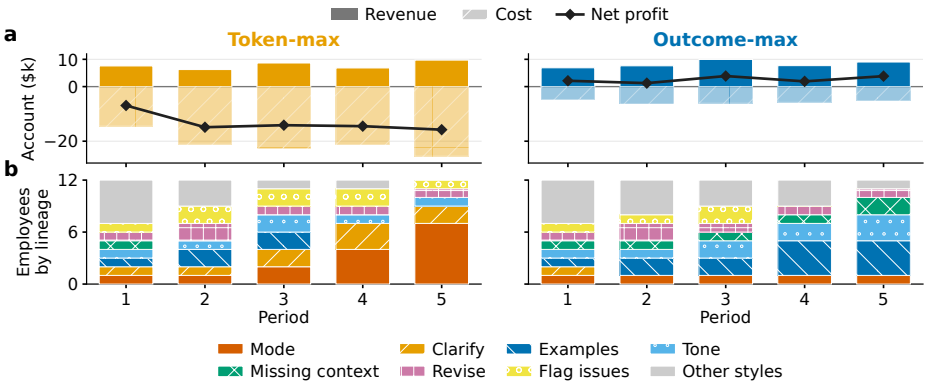


Figure 6: **Workforce selection and firm performance over five periods.** Left: token-max firm; right: outcome-max firm. (a) Revenue (bars above zero), cost (bars below zero), and net profit (line) under common scenario accounts. (b) Twelve working employees per period, colored by ancestral profile. Each period ends with three replacements from top-ranked lineages; expanding segments include descendants.

5.3 Evaluating OutcomeShare

5.3.1 RQ4: How do heuristic contracts affect delivery and payoffs?

In this section, we aim to evaluate delivery and payoffs when OutcomeShare is applied to fixed production under specified inspection assumptions. Table 3 compares token billing with outcome-based payments and damage-matched liability under a common task-value and salary scenario.

Under token billing, outcome-max increases correct production but also ships more defects, leaving the firm unprofitable. OutcomeShare makes delivery more selective: it sacrifices some correct outputs while removing a much larger share of defects. Relative to token-max, it delivers more correct work with fewer defects and improves provider revenue, employee income, and firm profit. The employee-heavy allocation gives the highest firm-profit estimate among the tested OutcomeShare rules (Appendix C.3.4).

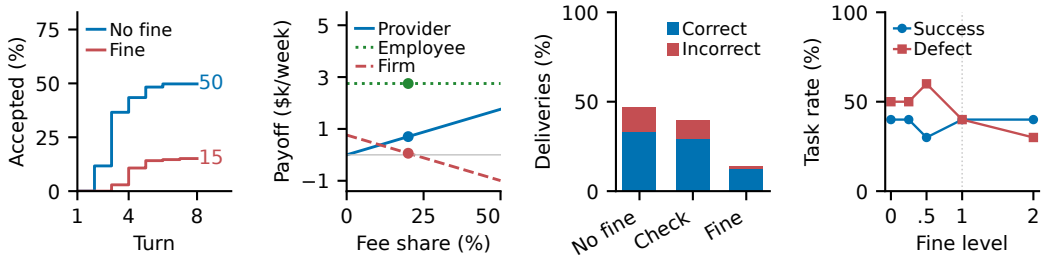
Overall, OutcomeShare combines efficient production with defect control and shared gains in this scenario. Its advantage depends on selective delivery and sufficient task value, rather than maximizing shipment volume alone.

Table 3: **Delivery and weekly accounts.** OutcomeShare reduces defective shipments while retaining more correct deliveries and improving all three participant accounts relative to token-max. Selective delivery trades some correct-output volume for defect control; settings are given in Appendix C.3.4.

	Deliveries / week		Weekly accounts (USD)		
Production / contract	Correct ↑	Wrong ↓	Provider ↑	Firm ↑	Employee ↑
Token-max / token billing	9.8	15.6	2	−5,429	2,400
Outcome-max / token billing	38.3	61.3	3	−14,746	2,400
Outcome-max / OutcomeShare	19.4	**2.9**	**535**	**275**	**3,443**

5.3.2 RQ5: How do acceptance, fees, and liability affect outcomes?

In this section, we aim to distinguish how acceptance, fees, and liability affect delivery and payoffs (Figure 7). Delivery rates use all attempted tasks as their denominator.



(a) Employee acceptance (b) Fee allocation (c) Delivered outputs (d) Fine level λ

Figure 7: **Acceptance, allocation, and defect control.** (a) Cumulative acceptance by turn with and without fine instructions; (b) weekly payoffs at fixed production versus fee share ϕ; (c) correct and defective deliveries on matched policies; (d) success and defects versus fine level λ.

Does liability change acceptance? Fine-conditioned policies show later, less frequent acceptance (Figure 7a), consistent with more selective approval under the combined liability and verification instructions.

Does the outcome fee affect shared gains? At fixed production, higher fees raise provider revenue and lower firm profit; employee income is unchanged (Figure 7b). Shared gains require satisfying all participation constraints (Eq. (5)).

Does verification improve delivered quality? On matched attempts, checking and liability reduce both defective and correct deliveries (Figure 7c; Appendix C.2.2). This is a quality–quantity tradeoff.

Does a larger fine improve outcomes? With learned policies fixed, fine and verification instructions yield non-monotonic success and defect rates (Figure 7d). These prompted responses involve no monetary deductions and do not identify an optimal fine (Appendix C.3.4).

6 Conclusion

We simulated token-max and outcome-max behaviors across execution, memory adaptation, and workforce selection. Outcome-max yields more verified work per agent token under these settings, with quality costs on some hard tasks. OutcomeShare identifies scenarios in which selective delivery and value sharing benefit all three participants.

The main comparison specifies two behaviors rather than testing whether objective optimization produces them. The small directional gain under objective-only prompts suggests a possible role for objective framing. Simulated participants and benchmark tasks limit external validity, while financial gains depend on verification, valuation, and collectible liability. Future experiments should separate reward from behavior instructions and test real-user participation and performance before drawing conclusions about workplace adoption.

AI USE STATEMENT

AI tools were used to assist with manuscript drafting and language polishing, literature retrieval and related-work discovery, research ideation and execution, and the development and checking of mathematical arguments. These uses supported the research process, with the specific applications described in the paper. The authors retain full responsibility for the originality and accuracy of the manuscript, the validity of its mathematical claims, the reliability of its references, and the interpretation of its results. The simulated employee and production agent roles, model configurations, and prompts are documented in Appendices A and D.

REPRODUCIBILITY STATEMENT

Appendix A specifies tasks, models, simulation settings, and accounting assumptions; Appendix B gives proofs. Appendix C reports supporting analyses, and Appendix D contains the prompts. The accompanying code and recorded outputs support the reported tables and figures. Code is available at `https://anonymous.4open.science/r/token-ecom-review-27FB/`.

REFERENCES

Lakshya A. Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J. Ryan, Meng Jiang, Christopher Potts, Koushik Sen, Alexandros G. Dimakis, Ion Stoica, Dan Klein, Matei Zaharia, and Omar Khattab. GEPA: Reflective prompt evolution can outperform reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2026.

Gati V. Aher, Rosa I. Arriaga, and Adam Tauman Kalai. Using large language models to simulate multiple humans and replicate human subject studies. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, volume 202 of *Proceedings of Machine Learning Research*, pp. 337–371. PMLR, 2023.

Lisa P. Argyle, Ethan C. Busby, Nancy Fulda, Joshua R. Gubler, Christopher Rytting, and David Wingate. Out of one, many: Using language models to simulate human samples. *Political Analysis*, 31(3):337–351, 2023.

George P. Baker. Incentive contracts and performance measurement. *Journal of Political Economy*, 100(3):598–614, 1992.

Joachim Baumann, Vishakh Padmakumar, Xiang Li, John Yang, Diyi Yang, and Sanmi Koyejo. SWE-chat: Coding agent interactions from real users in the wild. arXiv:2604.20779, 2026.

Joel Becker, Nate Rush, Elizabeth Barnes, and David Rein. Measuring the impact of early-2025 AI on experienced open-source developer productivity. *arXiv preprint arXiv:2507.09089*, 2025.

Erik Brynjolfsson, Danielle Li, and Lindsey R. Raymond. Generative AI at work. *Quarterly Journal of Economics*, 140(2):889–942, 2025.

Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, Zhuosheng Zhang, Rui Wang, Zhaopeng Tu, Haitao Mi, and Dong Yu. Do NOT think that much for 2+3=? On the overthinking of o1-like LLMs. arXiv:2412.21187, 2024.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. arXiv:2110.14168, 2021.

Alejandro Cuadron, Dacheng Li, Wenjie Ma, Xingyao Wang, Yichuan Wang, Siyuan Zhuang, Shu Liu, Luis Gaspar Schroeder, Tian Xia, Huanzhi Mao, Nicholas Thumiger, Aditya Desai, Ion Stoica, Ana Klimovic, Graham Neubig, and Joseph E. Gonzalez. The danger of overthinking: Examining the reasoning-action dilemma in agentic tasks. arXiv:2502.08235, 2025.

Kevin Zheyuan Cui, Mert Demirer, Sonia Jaffe, Leon Musolff, Sida Peng, and Tobias Salz. The effects of generative AI on high-skilled work: Evidence from three field experiments with software developers. *Management Science*, 2026. Articles in Advance, published online February 27, 2026.

Rick Dakan and Joseph Feller. AI fluency: Framework & foundations. Online course developed in collaboration with Anthropic, 2025. Introduces the 4D framework of Delegation, Description, Discernment, and Diligence. Accessed July 2026.

Alexander Erlei and Lukas Meub. LLM-agent interactions on markets with information asymmetries. arXiv:2603.08853, 2026.

Chen Gao, Xiaochong Lan, Nian Li, Yuan Yuan, Jingtao Ding, Zhilun Zhou, Fengli Xu, and Yong Li. Large language models empowered agent-based modeling and simulation: A survey and perspectives. *Humanities and Social Sciences Communications*, 11:1259, 2024.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *Advances in Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks Track*, 2021.

Bengt Holmström. Moral hazard and observability. *Bell Journal of Economics*, 10(1):74–91, 1979.

Bengt Holmström and Paul Milgrom. Multitask principal–agent analyses: Incentive contracts, asset ownership, and job design. *Journal of Law, Economics, & Organization*, 7:24–52, 1991.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Steven Yau, Zijuan Lin, Liyang Zhou, et al. Metagpt: Meta programming for a multi-agent collaborative framework. In *International Conference on Learning Representations*, volume 2024, pp. 23247–23275, 2024.

John J. Horton. Large language models as simulated economic agents: What can we learn from homo silicus? arXiv:2301.07543v1, 2023.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. SWE-bench: Can language models resolve real-world GitHub issues? In *International Conference on Learning Representations (ICLR)*, 2024.

Sayash Kapoor, Benedikt Stroebl, Zachary S. Siegel, Nitya Nadgir, and Arvind Narayanan. AI agents that matter. *Transactions on Machine Learning Research*, 2025.

Edward P Lazear. Performance pay and productivity. *American Economic Review*, 90(5):1346–1361, 2000.

Guohao Li, Hasan Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. CAMEL: Communicative agents for "mind" exploration of large language model society. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations (ICLR)*, 2024.

Yuxiang Lin, Zihan Wang, Mengyang Liu, Yuxuan Shan, Longju Bai, Junyao Zhang, Xing Jin, Boshan Chen, Jinyan Su, Xingyao Wang, Jiaxin Pei, and Manling Li. BAGEN: Are LLM agents budget-aware? arXiv:2606.00198, 2026.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is your code generated by ChatGPT really correct? Rigorous evaluation of large language models for code generation. In *Advances in Neural Information Processing Systems*, volume 36, pp. 21558–21572, 2023.

Ilan Lobel, Yanwei Sun, and Jiahua Wu. Pricing ai as a service. *Available at SSRN 6254459*, 2026.

Jyoti Mann. Tokenminimizing: Meta moves to curb employee AI usage as AI costs reach billions. The Information, June 2026. 12 June 2026.

Shakked Noy and Whitney Zhang. Experimental evidence on the productivity effects of generative artificial intelligence. *Science*, 381(6654):187–192, 2023.

OpenAI. Introducing SWE-bench Verified. OpenAI Blog, 2024.

Joon Sung Park, Joseph C. O'Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology (UIST)*, 2023.

Paulius Rauba, Simonas Cepenas, and Mihaela van der Schaar. Multi-agent systems should be treated as principal-agent problems. arXiv:2601.23211, 2026.

Kevin Roose. More! more! more! tech workers max out their A.I. use. The New York Times, March 2026. 20 March 2026.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling parameters for reasoning. In *International Conference on Learning Representations (ICLR)*, 2025.

Nanda Kishore Sreenivas and Kate Larson. Contracting for LLM delegation: Moral hazard in technology and effort choice. arXiv:2608.18232, 2026.

Michelle Vaccaro, Abdullah Almaatouq, and Thomas W. Malone. When combinations of humans and ai are useful: A systematic review and meta-analysis. *Nature Human Behaviour*, 8:2293–2303, 2024.

Alexander Sasha Vezhnevets, John P. Agapiou, Avia Aharon, Ron Ziv, Jayd Matyas, Edgar A. Duéñez-Guzmán, William A. Cunningham, Simon Osindero, Danny Karmon, and Joel Z. Leibo. Generative agent-based modeling with actions grounded in physical, social, or digital space using Concordia. *arXiv preprint arXiv:2312.03664*, 2023.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023.

Qizheng Zhang, Changran Hu, Shubhangi Upasani, Boyuan Ma, Fenglu Hong, Vamsidhar Kamanuru, Jay Rainton, Chen Wu, Mengmeng Ji, Hanchen Li, Urmish Thakker, James Zou, and Kunle Olukotun. Agentic context engineering: Evolving contexts for self-improving language models. In *International Conference on Learning Representations (ICLR)*, 2026.

Wenting Zhao, Xiang Ren, Jack Hessel, Claire Cardie, Yejin Choi, and Yuntian Deng. WildChat: 1m ChatGPT interaction logs in the wild. In *International Conference on Learning Representations (ICLR)*, 2024.

Xuhui Zhou, Hao Zhu, Leena Mathur, Ruohong Zhang, Haofei Yu, Zhengyang Qi, Louis-Philippe Morency, Yonatan Bisk, Daniel Fried, Graham Neubig, and Maarten Sap. SOTOPIA: Interactive evaluation for social intelligence in language agents. In *International Conference on Learning Representations (ICLR)*, 2024.

A Experimental Setup and Evaluation Protocols

Appendix guide. Each entry points to supporting evidence or implementation detail.

Question	Where to look
Tasks, simulation levels, validity, and accounting	Appendix A
Objective and verification incentives	Appendix B
Robustness: cohorts, models, personas, budgets	Appendix C.1
Ablation: adaptation and verification components	Appendix C.2
Analysis: stopping, selection, accounts, sensitivity	Appendix C.3
Exact behavioral prompts	Appendix D

In this section, we aim to define comparable tasks, participant roles, and resource accounts for the three simulation levels. A shared evaluation protocol compares policy packages under common task and model settings.

A.1 General setup

We define a common interaction and evaluation protocol so that objective comparisons use the same tasks, models, and resource limits.

Production interaction and evaluation. Let a task $x_j \in \mathcal{X}$ be assigned to employee i. A production agent proposes actions or candidate outputs, and the employee provides feedback, requests another production step, or stops; the completed trajectory H_{ij} records the interaction. The trajectory is scored by a task-specific evaluator $E : \mathcal{X} \times \mathcal{H} \to \{0, 1\}$ external to both participants, so $y_{ij} = E(x_j, H_{ij})$ is *verified* rather than perceived completion. The main comparison directly specifies token-max and outcome-max behaviors, holding task, evaluator, models, and resource limits fixed within each pair. It evaluates their consequences rather than whether optimizing an objective produces the behaviors. The instruction sets are: *token* (maximize interaction volume; hard to satisfy), *outcome* (a correct fix at minimum cost; stop once correct), and, for the audit-fine mechanism (Section 4), *audit-fine* (accept only after verification; shipped defects are audited and fined).

Tasks and evaluators. Repository software engineering uses SWE-bench Verified (500 human-validated GitHub issues; correctness requires the gold tests to pass in an isolated container (Jimenez et al., 2024; OpenAI, 2024)). Program synthesis uses HumanEval+ (164 problems, dataset-provided test programs (Liu et al., 2023)). Grade-school mathematics uses the first 250 tasks of the GSM8K test split (Cobbe et al., 2021). Competition mathematics uses the first 250 tasks of MATH-500 (Hendrycks et al., 2021; Lightman et al., 2024). Both mathematics benchmarks use boxed-answer extraction and normalized comparison. Behavioral coverage is checked against WildChat-1M (Zhao et al., 2024) coding conversations (Appendix A.6).

Difficulty groups. Table 1 fixes one grouping per benchmark before comparing objectives. SWE-bench uses estimated human repair time: below 15 minutes (Easy), 15 minutes to one hour (Medium), and at least one hour (Hard). The scheduled L1 grid contains five tasks per group. L2 uses its separate held-out task set. MATH uses official levels 1–2, 3, and 4–5. HumanEval+ and GSM8K have no corresponding labels in these records, so we use reference-solution complexity proxies. HumanEval+ counts Python AST statements in the canonical solution, excluding docstrings; the groups are at most 4, 5–8, and at least 9 statements. GSM8K counts annotated calculations in the reference answer; the groups are at most 2, 3–4, and at least 5. These thresholds are the one-third and two-thirds empirical quantiles of the full reference splits, with ties kept together. They are structural proxies, not validated difficulty scales, and need not order model success monotonically. All groups pool valid-run successes and counts directly.

Mathematics task identities are inferred from the task-major ordering of the first 250 benchmark items and the four-persona cycle because per-attempt IDs are unavailable. Difficulty results are conditional on this ordering; HumanEval+ and SWE-bench instead use explicit task IDs.

Models and roles. The fixed-policy SWE-bench grid uses Claude Haiku 4.5 production agents and Claude Sonnet 4.6 employees across 14 profiles, 15 tasks, and two repetitions. The mathematical and

program-synthesis runners specify DeepSeek v4 Pro agents and v4 Flash employees. L2 separately uses DeepSeek v4 Pro agents, Haiku 4.5 employees, and Sonnet 4.6 reflection. These identities come from experiment configurations; per-attempt model telemetry is unavailable. The static L1 grid and L2 adaptation runs use different task cohorts and model pairings, so comparing their absolute success rates does not isolate learning. L3 is evaluated on its population-selection task pool. Its trajectories are not included in the fixed-policy L1 row.

Employee personas. Table S1 defines the employee behaviors adapted from AI Fluency (Dakan & Feller, 2025); Appendix D gives their prompt encoding.

Table S1: **Employee behavior dimensions.** The first six dimensions control description; the remaining five control discernment. Low and high archetypes use propensities of 0.1 and 0.9; single-behavior profiles elevate one dimension.

Dimension	Low propensity	High propensity
Iterates and refines	Accepts initial output	Requests revisions
Clarifies goals	Leaves goals implicit	States goals and constraints
Provides examples	Gives no examples	Supplies examples
Specifies format	Leaves format open	Specifies output structure
Sets interaction mode	Uses no explicit role	Assigns a collaboration role
Communicates tone	Leaves style open	States style preferences
Identifies missing context	Omits missing information	Supplies needed context
Questions reasoning	Accepts explanations	Challenges reasoning
Fact-checks outputs	Trusts claims	Checks correctness
Flags issues	Leaves errors unchallenged	Points out errors
Requests alternatives	Accepts one approach	Requests other approaches

Measurement. With verified outcome $y_{ij} \in \{0, 1\}$, we price recorded model usage according to Eq. (1). Main-table efficiency consistently counts production-agent input and output tokens. Diagnostics that include employee tokens are labeled separately. L3 usage contains input and output counts without separate cache measurements; its accounts therefore use two-component input/output pricing. Human supervision cost is $C_{ij}^{\text{human}} = w\, h_{ij}$, with loaded wage w and a scenario-based supervision-time proxy h_{ij} (anchored at 15 min per attempt, scaled by persona-token burden); dollar levels are therefore scenario results, whereas token use and verified correctness are directly measured. Over the valid runs $\mathcal{V}$ we report the resolution rate $\bar{y}$, mean tokens $\bar{T}$, and the outcome efficiency of (6) (resolved tasks per million tokens). The two organizational key-performance indicators used for evaluation and selection, s_i^{tok} and s_i^{out}, are defined in (7), where $\rho_i \in [0, 1]$ is the mean parsimony of resolved patches (a 0.9–1.1 tie-breaker toward smaller patches). Cost-reduction ratios are computed within each domain and never averaged across domains.

We study three levels of increasing organizational complexity, sharing the setup above.

A.2 Level-1 simulation: supervised production

In this section, we aim to measure the consequences of specified supervision policies within a task. A single supervised interaction per task supports the token-versus-outcome comparison and the over-efforting analysis (Section 5.2.1). The cross-task comparison spans all four task families; the balanced *static persona grid* crosses 14 profiles $\times$ 15 tasks $\times$ 2 repetitions per objective (420 scheduled runs per arm before validity filtering), so every persona faces the same tasks in both arms. A separate Haiku validation set evaluates frozen generation-5 learned policies on 100 difficulty-stratified SWE-bench tasks with two repetitions; it is not an extension of the fixed-policy L1 grid (Table S7). A persona-only model grid crosses three agent models with two employee models over four profiles $\times$ 15 tasks. The two mathematics datasets each use 250 problems $\times$ 4 personas (2,000 runs across both datasets) and HumanEval+ 164 $\times$ 4 (656 runs), with at most 10 employee exchanges; the software runs allow at most 8 exchanges and 50 internal agent turns per exchange.

A.3 Level-2 simulation: human–agent co-evolution

In this section, we aim to test whether the policy gap persists as participants learn reusable policies. Both sides adapt across generations for the skill-learning results (Section 5.2.1). Curation uses 20 training and 15 disjoint held-out SWE-bench tasks; the protocol runs for three generations under

each objective and in three regimes that vary who adapts (employee-only, agent-only, or both), with per-profile runs over 11 starting profiles. The learning *protocol* itself is defined in Appendix A.5 (Algorithm 1); frozen generation-3 skills are evaluated on the held-out set and, for the efficiency panel, on the 100-task validation set.

A.4 Level-3 simulation: population selection (the firm)

In this section, we aim to evaluate workforce changes under specified retention scores and repeated selection. A firm selects its workforce on a leaderboard for the firm results (Section 5.2.1). Each firm begins with $n = 12$ employees (one all-low baseline plus one per elevated behavior dimension), assigns 20 backlog tasks per employee per period, and runs for 5 organizational periods (1,200 launched runs per arm). Each firm uses its assigned supervisory policy and retention score, s^{tok} versus s^{out} of (7): at the end of period t the bottom $K = 3$ employees are removed and replaced by mutated descendants of the top K (one randomly selected profile dimension flipped with probability 0.2), so the workforce updates as in (8). Period operating profit uses difficulty-weighted ground-truth solves (so a firm cannot profit from false acceptance) with the stated scenario of Table S2.

The two selection scores are

$$s_i^{\mathrm{tok}} = \frac{1}{n_i}\sum_j T_{ij}, \qquad s_i^{\mathrm{out}} = \frac{\sum_j y_{ij}}{\sum_j C_{ij}^{\mathrm{AI}}}(\alpha + \beta\rho_i), \tag{7}$$

where n_i counts evaluated tasks and ρ_i is resolved-patch parsimony. The fixed simulation uses $\alpha = 0.9$, $\beta = 0.2$, $n = 12$, $K = 3$, and mutation probability $\mu = 0.2$; these are simulation settings, not parameters fitted to the observed objective gap. Selection applies

$$\mathcal{W}_{t+1} = \mathrm{survivors}_{n-K}(\mathcal{W}_t; s) \cup \mathrm{mutate}(\mathrm{top}_K(\mathcal{W}_t; s)). \tag{8}$$

with $K = 3$; each recruited profile has probability 0.2 of one dimension flip.

Table S2: **Population-accounting assumptions.** Monetary values define the stated scenario, not measured workplace prices or returns.

Parameter	Scenario value
Annual backlog	2,400 tasks
Task-value tiers	\$16, \$40, \$200, \$640
Loaded wage	\$80 per hour
Employee-time anchor	15 minutes per attempt
Input, output, cache prices	\$0.27, \$1.10, \$0.07 per million tokens
Employees and periods	12 employees; five selection periods

A.5 Learning and Economic Accounting

This subsection specifies the learning protocol used in Section 5.2.1 and the scenario accounting used to evaluate the mechanism in Section 4.

A.5.1 Metric-conditioned bilateral verbal reinforcement

In this section, we aim to separate learning by the employee from learning by the production agent. Each adapts a textual policy from shared outcome feedback, allowing comparisons across adaptation roles.

Learning architecture. We extend verbal reinforcement—in which a language model improves a persistent text policy from experience without weight updates (Shinn et al., 2023)—from a single learner to two functionally distinct ones: an employee library L_g of interaction tactics (how to instruct, verify, and stop) and an agent library K_g of production tactics (how to interpret, construct, and stop). All model weights stay fixed across generations.

Joint policy and outcome-only feedback. With objective $o \in \{\mathrm{tok}, \mathrm{out}\}$, persona b, and textual state $Z_g = (L_g, K_g)$, an interaction yields a trajectory $H_{gj} \sim P(\cdot \mid x_j, b, o, L_g, K_g)$. After the external evaluator runs, a deterministic compression A keeps only the outcome vector

$$f_{gj} = A(H_{gj}) = (y_{gj}, a_{gj}, d_{gj}, \tau_{gj}, T_{gj}), \qquad F_g = \{f_{gj} : j \in \mathcal{D}_g^{\mathrm{train}}\}, \tag{9}$$

where a is the acceptance decision, $d = a(1 - y)$ a shipped-defect indicator, and τ the exchange count; the reflection model sees F_g and the current library, but no task text, output, or transcript.

Role-specific, metric-conditioned updates. Two operators transform the common feedback,

$$L_{g+1} = R_H(L_g, F_g; u_H^o), \qquad K_{g+1} = R_A(K_g, F_g; u_A), \tag{10}$$

each returning at most ten imperative tactics. The employee utility u_H^o is metric-conditioned (token reflection is asked to increase interaction volume, outcome reflection to increase correct resolutions per token); u_A asks the agent to raise correct resolutions at fewer tokens. Both updates depend on the same F_g, which was generated jointly by L_g and K_g, so the two libraries co-evolve through shared, externally verified feedback across a generational barrier—distinct from multi-agent weight training. Three ablations (who adapts: employee-only, agent-only, both) use the same rule. Frozen skills are scored on the disjoint held-out set by

$$\eta_{g,o}^{\text{test}} = \frac{\sum_{j \in \mathcal{D}^{\text{test}}} y_{gj}}{\sum_{j \in \mathcal{D}^{\text{test}}} T_{gj}/10^6}; \tag{11}$$

because the reflection channel carries no task text, changes in (11) measure transfer of reusable routines, not replay. Algorithm 1 summarizes the protocol.

A.5.2 Outcome-based pricing and the three-party model

In this section, we aim to connect verified production with the payoffs received by each participant. Throughout, the three parties are the LLM provider, firm, and employee. The employee supervises production; the production agent is the LLM system.

Production and review capacity. Weekly production is limited by whichever resource runs out first: agent computation or employee review time. If their capacities are μ_A and μ_H tasks per week, attempted output is $M = \min(\mu_A, \mu_H)$ and verified output is $Q = M\bar{y}$. Agent capacity divides its weekly output-token budget by output tokens per task; review capacity divides available minutes by review minutes per task. With different task values, weekly verified value is $Z = M\,\overline{Vy}$: we value each successful task before averaging.

Allocating verified value. The outcome fee ϕ transfers a share of Z to the provider, and the bonus rate B_E transfers another share to the employee. The firm retains what remains after salary. Thus provider revenue is ϕZ, employee income is salary plus $B_E Z$, and firm profit is $(1 - \phi - B_E)Z$ minus salary. These accounts hold observed production fixed as the fee changes. They do not assume that a higher fee improves behavior. Provider revenue is reported before its compute costs, so it does not measure provider profit.

Pigouvian audit-fine. A shipped defect is detected with effective probability p (audit sampling probability times audit sensitivity), and a fine F is levied on its approver, with $p \cdot F = \kappa V$ so the expected fine equals the social damage κV (rework cost per unit value). The equality internalizes expected defect damage in the reduced effort model below; it does not by itself establish a particular fee optimum or a joint improvement in all three parties' payoffs. The observed outcome and audit-fine comparison is reported in Table S11b, separately from the fixed-skill fine sweep in Table S22.

In the behavioral experiments, liability is communicated through the simulated employee's objective and, for learned policies, its reflection instructions. The audit-fine prompt describes a large penalty for approving an incorrect fix. The fixed-policy sweep instead states expected penalties of $0, 0.25, 0.5, 1,$ or 2 times the value of a correct solve. These runs evaluate acceptance and correctness; they do not separately sample audit events or execute monetary deductions. Thus they test responses to stated liability, while the contract equation specifies how an implemented audit and fine would produce that expected liability. Identifying the sweep's multiplier with $\lambda = pF/(\kappa V)$ uses the explicit normalization $\kappa = 1$: defect damage is valued at one correct solve. This calibration is an accounting assumption, not an estimated damage level.

Three-party payoff gains. Weekly payoff gains apply the accounting of Table S2 per employee seat to the valid labor-market runs of Section 5.2.1 (token arm versus outcome arm, all generations).

Input: objective o; initial libraries (L_0, K_0); fixed models; training set $\mathcal{D}^{\text{train}}$; held-out set $\mathcal{D}^{\text{test}}$; evaluator E; generations G.
Output: $\{(L_g, K_g)\}_{g=1}^{G}$ and $\{\eta_{g,o}^{\text{test}}\}$.

1. For $g = 0, \ldots, G-1$: execute $\mathcal{D}^{\text{train}}$ under (L_g, K_g, o) to obtain $\{H_{gj}\}$.
2. Apply E and A to build F_g; drop invalid runs.
3. Update enabled libraries by (10); retain the previous library if an update is empty.
4. Freeze (L_{g+1}, K_{g+1}) and evaluate on $\mathcal{D}^{\text{test}}$ without reflection by (11).
5. Pass only (L_{g+1}, K_{g+1}) across the barrier; never pass held-out outcomes to a reflection operator.

Algorithm 1: Metric-conditioned bilateral verbal reinforcement

Weekly throughput is the tandem queue above with an agent seat of 40 h/week at 50 output tokens/s and 30 h/week of supervision, where a task's supervision minutes scale the 15-minute anchor by its persona-token burden; the employee salary is $30\,\text{h} \times \$80 = \$2{,}400$ per week. Under token pricing the LLM provider bills tokens at the agent prices of Table S2, the employee earns the salary, and the firm books the value of resolved fixes minus the bill and the salary; under outcome pricing (fee $\phi = 0.20$, employee bonus 10% of each verified fix) the LLM provider earns $\phi M\overline{Vy}$, the employee salary plus bonus, and the firm the remainder. Confidence intervals jointly resample the 50 shared tasks across arms (20,000 resamples, 95% percentile intervals), preserving within-task dependence across employees and periods. Calibration and capacity parameters stay fixed; these intervals do not measure variation across independently simulated firms.

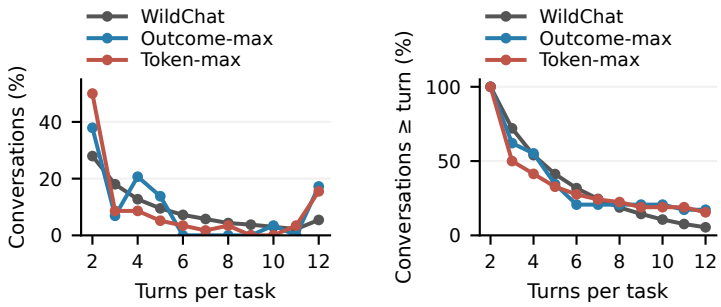
The verification guarantee and its assumptions are proved in Appendix B; the accounting above does not assume that the complete contract has been jointly deployed.

A.6 Behavioral Coverage

In this section, we aim to assess measurement validity and behavioral coverage through common exclusion rules, task-cluster uncertainty, and comparisons with human conversations.

Behavioral diagnostics. Termination records distinguish employee acceptance from resource-limit or agent termination. Stored follow-ups are classified by keyword rules into requests for tests, alternatives, explanation, scope expansion, and questions, plus an approve-and-stop category. Persona comparisons use the balanced grid; tool and turn-cost summaries use their stated run cohorts. Supervision minutes are a scenario proxy rather than measured human labor. These diagnostics are descriptive mediators, not independently randomized interventions.

Coverage of human behavior. Prior studies assess how closely simulated populations reproduce human behavior (Argyle et al., 2023; Aher et al., 2023). We assess behavioral coverage by comparing simulated interactions with WildChat-1M (Zhao et al., 2024) coding conversations. A high-recall lexical filter selects candidate multi-turn conversations, followed by LLM coding validation. Follow-up style, supervision mode, and perceived completion are annotated without the treatment label using a fixed annotation seed. Figure S1 shows overlap in interaction depth. Simulated personas are more decisive and less likely to begin with underspecified requests or expand scope than observational users. WildChat lacks executable correctness labels and randomized objectives; this check assesses behavioral coverage, not workplace productivity effects.



(a) Conditional user-turn distribution.

(b) Depth in the multi-turn tail.

Figure S1: **Interaction-depth coverage.** Both panels compare WildChat coding conversations with simulated interactions, conditioned on multiple user turns. The tail view groups turns 2–12; it is not an additional sample. Overlap supports behavioral coverage, not causal productivity validity.

B Objective Properties and Verification Incentives

In this section, we aim to establish how objectives govern continuation and how liability aligns verification (Theorems 1 and 2). The analysis concerns the objectives and mechanism in Sections 3 and 4. Neither guarantee assumes that prompted LLMs reach an optimal policy, and neither proves that outcome-max improves resolution on every task distribution.

B.1 Unproductive continuations

In this section, we aim to characterize the response to additional computation, first when verified output is unchanged and then when it improves. The ratio of expected output to expected cost measures efficiency over the workload; averaging per-task output-to-cost ratios would instead give more weight to cheap tasks.

Proof of Theorem 1. Consider two feasible policies π, π' on the same task distribution. Suppose π' adds a continuation with no change in expected verified resolution: $\mathbb{E}_{\pi'}[y] = \mathbb{E}_{\pi}[y] = Y > 0$. If expected tokens increase by $\Delta T > 0$ and expected monetary compute cost increases by $\Delta C > 0$, with $C = \mathbb{E}_{\pi}[C^{\mathrm{AI}}] > 0$, then

$$J_{\mathrm{tok}}(\pi') - J_{\mathrm{tok}}(\pi) = \Delta T > 0, \qquad J_{\mathrm{out}}(\pi') - J_{\mathrm{out}}(\pi) = -\frac{Y\,\Delta C}{C(C + \Delta C)} < 0. \tag{12}$$

Substitute the two policies' expected token counts in Eq. (2). For Eq. (3), subtract Y/C from $Y/(C + \Delta C)$ and combine denominators. All factors in the resulting denominator and numerator are positive.

If $Y = 0$, both outcome-efficiency scores are zero, so the strict conclusion does not apply. More generally, if the continuation changes expected resolution by ΔY, subtracting Y/C from $(Y + \Delta Y)/(C + \Delta C)$ gives $(C\Delta Y - Y\Delta C)/(C(C + \Delta C))$. Since the denominator is positive,

$$J_{\mathrm{out}}(\pi') > J_{\mathrm{out}}(\pi) \quad \Longleftrightarrow \quad \Delta Y\, C > Y\, \Delta C. \tag{13}$$

□

For positive ΔC, this compares the marginal resolution gain per dollar with the current average. It explains why outcome-max may favor additional reasoning when useful; it does not prescribe universally shorter trajectories. The theorem is a comparison of policies with stipulated expectations. The experiments establish whether observed continuations have the relevant properties, rather than assuming that all extra tokens are wasteful.

B.2 Pigouvian verification alignment

In this section, we aim to identify when an audit-fine internalizes expected damage from shipped defects. The result concerns the verification decision under the assumptions stated below.

Proof of Theorem 2. Let $\mathcal{E}$ be a common nonempty feasible set of verification efforts and suppose both minimization problems below attain their minima. Let $c(e)$ be the effort cost and $\delta(e)$ the probability of a shipped defect. Assume: (i) the employee is risk neutral; (ii) audits detect a shipped defect with fixed effective probability $p > 0$ and the fine F is collectible; (iii) each shipped defect causes external damage κV; and (iv) all remaining private and social payoff terms are constant in e. Define

$$L_{\text{private}}(e) = c(e) + pF\delta(e), \qquad L_{\text{social}}(e) = c(e) + \kappa V\delta(e). \tag{14}$$

Then $pF = \kappa V$ implies

$$\arg\min_{e\in\mathcal{E}} L_{\text{private}}(e) = \arg\min_{e\in\mathcal{E}} L_{\text{social}}(e). \tag{15}$$

Substitution makes the two loss functions equal at every feasible e. Their ordering of all feasible efforts and hence their sets of minimizers are identical. Fines are transfers, so they are not counted as additional social damage. □

This calibration is sufficient, but need not be necessary: distinct coefficients can share a boundary minimizer or give identical incentives when δ is constant. With differentiable costs and an interior common optimum e^* at which $\delta'(e^*) \neq 0$, subtracting the two first-order conditions does imply $pF = \kappa V$. Existence, uniqueness, and interiority require additional assumptions.

Scope and implementation limitations. If effort changes a private bonus $b(e)$, the private objective becomes $c(e) + pF\delta(e) - b(e)$. If effort also changes throughput or social production value, the social objective changes as well. Eq. (4) alone then need not align the full objectives. Likewise, audit expense, risk aversion, limited liability, and imperfectly measured damages alter implementation. The empirical audit-fine intervention is an instruction to a simulated employee; it is evidence about behavioral responses to that instruction, not a test of monetary incentive compatibility with human participants. Fixed-production fee accounts allocate observed value; the theorem does not establish that those accounts yield gains for every participant.

C Supplementary Experiments and Analysis

In this section, we aim to assess the scope and consequences of the token-max/outcome-max policy comparison. We ask whether the policy comparison is robust across settings (Section C.1), which design components change behavior (Section C.2), and what trajectories and accounts explain the observed outcomes (Section C.3). Throughout, success means external correctness, tokens count production-agent input plus output, and efficiency η means verified solutions per million agent tokens. Acceptance and bad delivery retain the separate definitions from Section 3. Scenario accounts follow the main tables.

C.1 Robustness: Where Does the Objective Contrast Hold?

In this section, we aim to test robustness across personas, model pairings, interaction budgets, and repositories. These comparisons distinguish consistent resource savings from setting-dependent effects on correctness.

C.1.1 Task-level uncertainty and matched success

In this section, we aim to distinguish policy gaps from task-sampling variation through matched comparisons.

Table S3 reports policy gaps on shared task–persona–repetition cells, with 95% task-cluster bootstrap intervals. Resampling tasks keeps repeated attempts together. The primary comparison contains 13 tasks; the hard-task and objective-only cohorts offer less independent evidence. These intervals describe uncertainty within the observed tasks, rather than variation across new learning histories. Matching also explains small differences from the full-cohort main table.

Efficiency uses total successes divided by total agent tokens. Rare bootstrap draws with no successes in either arm are conservatively bounded when forming relative-gap intervals, rather than discarded. Boldface in the main table marks numerical comparisons, not statistical significance. Overall, task

matching supports the efficiency advantage, while uncertain success gaps and hard-task losses limit claims of uniform quality improvement.

Table S3: **Matched success rates and task-cluster uncertainty.** Gaps are outcome-max minus token-max, with pointwise 95% task-cluster bootstrap intervals. Bold marks the higher success rate; ties are unbolded.

Setting	Token (%)	Outcome (%)	Gap [95% CI] (pp)
(a) Primary SWE benchmark			
Overall	**51.3**	50.3	-1.0 [-8.2, +5.2]
Easy	62.0	**67.6**	+5.6 [+0.0, +11.4]
Medium	46.8	**50.0**	+3.2 [+0.0, +7.4]
Hard	**43.6**	26.9	-16.7 [-24.5, +0.0]
(b) Model and persona robustness			
Haiku 4.5	51.9	**61.6**	+9.7 [+2.8, +16.7]
DeepSeek v4 Pro	58.1	**61.0**	+2.9 [-2.1, +7.8]
more guidance	45.8	**50.0**	+4.2 [+0.0, +13.0]
more checking	57.9	57.9	+0.0 [-15.0, +15.0]
both	52.2	**56.5**	+4.3 [-13.6, +22.7]
goal clarification	42.9	**52.4**	+9.5 [+0.0, +22.7]
fact checking	**40.0**	35.0	-5.0 [-22.7, +10.5]
iterative revision	**56.5**	39.1	-17.4 [-45.8, +13.6]
(c) Prompt robustness			
Objective + instructions	**51.3**	50.3	-1.0 [-8.2, +5.2]
Objective only	37.5	37.5	+0.0 [-12.0, +12.0]

C.1.2 Objective gaps across backbones, personas, and prompts

In this section, we aim to determine whether models, personas, and prompt wording change the policy contrast. Comparisons are paired within each setting; cross-setting differences do not isolate any single factor.

Figure 3 and Table S4 compare policy packages within each setting. Panel (a) shows higher efficiency and success under both model pairings. A stricter common-task comparison retains the efficiency advantage but yields opposite success gaps across pairings. Resource savings are therefore more stable than correctness gains.

Panel (b) shows the same distinction across supervision styles: all six profiles save resources, but fact checking and revision lose success. Revise's emphasis on iterative refinement may fit token-max's continuation instructions better; the uncertain success gap supports this as a possible explanation, not a general persona ranking.

Panel (c) shows a small positive efficiency estimate and unchanged success when prompts state only the payoff. The matched efficiency gain is 4.5%, with a 95% task-cluster interval of $[-31.9, 62.4]\%$. This is a directional signal, not a precisely established effect. Differences in models and personas also prevent isolating the effect of prompt wording. Taken together, these comparisons support more stable efficiency gains than correctness gains for explicit policy packages. The objective-only condition does not establish that changing rewards alone generates the instructed supervisory styles.

C.1.3 Model and persona scope

In this section, we aim to assess the range of models and supervisory styles over which resource use changes. Persona and learned-policy comparisons use distinct cohorts.

Tables S5 and S6 compare four behavior profiles across six agent–employee configurations. Stronger description and scrutiny substantially change token use, but neither yields uniformly better success. These profiles carry no objective directive, so they compare the consequences of specified supervision styles without identifying responses to rewards. Persona efficiency gains use all-low as the reference; objective comparisons use token-max. Model assignments follow the experimental configurations; these supplementary comparisons have less complete evaluation-integrity evidence.

Table S7 extends the objective comparison to learned policies and a separate validation cohort. It again shows that resource savings need not imply a uniform success improvement. Different learning histories and task sets prevent interpreting these results as an isolated backbone effect. Overall,

Table S4: **Paired means underlying Figure 3.** Within each pair, bold marks higher success and efficiency or lower token use; ties are unbolded.

Setting	Objective	Success (%)	Tokens (k)	η
Haiku 4.5	Token-max	51.9	15.5	33.57
	Outcome-max	**61.6**	**8.3**	**73.86**
DeepSeek v4 Pro	Token-max	58.1	281.3	2.06
	Outcome-max	**61.0**	**78.5**	**7.78**
Detailed guidance	Token-max	45.8	30.2	15.16
	Outcome-max	**50.0**	**7.4**	**67.33**
Active checking	Token-max	57.9	18.7	30.98
	Outcome-max	57.9	**11.3**	**51.41**
Guidance + checking	Token-max	52.2	35.7	14.61
	Outcome-max	**56.5**	**11.5**	**49.14**
Goal clarification	Token-max	42.9	20.5	20.86
	Outcome-max	**52.4**	**10.6**	**49.61**
Fact checking	Token-max	**40.0**	12.6	31.69
	Outcome-max	35.0	**8.1**	**43.00**
Iterative revision	Token-max	**56.5**	20.5	27.57
	Outcome-max	39.1	**9.3**	**41.91**
Objective + instructions	Token-max	**51.3**	21.5	23.81
	Outcome-max	50.3	**9.4**	**53.59**
Objective only	Token-max	37.5	183.4	2.05
	Outcome-max	37.5	**175.4**	**2.14**

Table S5: **Backbone and human-persona results with an Opus employee.**

Different backbones and human persona
SWE-bench; human: Opus 4.6

Agent	Human behavior	Overall	Easy	Medium	Hard	Tokens (k)	Eff. gain (%)
Haiku 4.5	Minimal input	26.7	20.0	40.0	20.0	9.8	0.0
	Detailed guidance	26.7	40.0	20.0	20.0	11.6	-15.5
	Active checking	46.2	40.0	66.7	40.0	23.1	-26.1
	Guidance + checking	28.6	40.0	25.0	20.0	25.0	-57.8
Sonnet 4.6	Minimal input	42.9	60.0	40.0	25.0	12.1	0.0
	Detailed guidance	50.0	60.0	60.0	25.0	14.3	-1.3
	Active checking	50.0	60.0	60.0	25.0	28.0	-49.6
	Guidance + checking	40.0	60.0	40.0	20.0	25.5	-55.8
Opus 4.8	Minimal input	33.3	20.0	40.0	40.0	10.3	0.0
	Detailed guidance	33.3	40.0	40.0	20.0	19.1	-46.1
	Active checking	40.0	40.0	40.0	40.0	27.7	-55.4
	Guidance + checking	33.3	40.0	40.0	20.0	25.2	-59.1

Table S6: **Backbone and human-persona results with a Haiku employee.**

Different backbones and human persona
SWE-bench; human: Haiku 4.5

Agent	Human behavior	Overall	Easy	Medium	Hard	Tokens (k)	Eff. gain (%)
Haiku 4.5	Minimal input	40.0	60.0	40.0	20.0	8.6	0.0
	Detailed guidance	54.5	60.0	66.7	33.3	43.2	-72.8
	Active checking	25.0	40.0	33.3	0.0	38.4	-86.0
	Guidance + checking	50.0	60.0	66.7	25.0	49.9	-78.4
Sonnet 4.6	Minimal input	42.9	60.0	40.0	25.0	10.7	0.0
	Detailed guidance	42.9	60.0	40.0	25.0	25.5	-58.3
	Active checking	40.0	60.0	40.0	20.0	41.0	-75.8
	Guidance + checking	46.7	60.0	60.0	20.0	41.3	-71.9
Opus 4.8	Minimal input	53.3	60.0	60.0	40.0	9.7	0.0
	Detailed guidance	46.7	60.0	40.0	40.0	33.4	-74.5
	Active checking	46.7	60.0	40.0	40.0	64.3	-86.7
	Guidance + checking	40.0	60.0	40.0	20.0	49.2	-85.1

supervisory style changes resource use across backbones without establishing a universal success ranking.

Table S7: **Objective comparisons under learned policies.** Each model pairing uses its own training and validation protocol.

Policy	Objective	Success (%) ↑	Tokens (k) ↓	Eff. gain (%) ↑
Haiku 4.5 agent; Sonnet 4.6 employee				
Haiku 4.5	Token-max	51.6	15.5	0.0
	Outcome-max	**60.5**	**8.4**	**+117.4**
DeepSeek v4 Pro agent; Haiku 4.5 employee				
DeepSeek v4 Pro	Token-max	55.8	262.7	0.0
	Outcome-max	**63.0**	**91.5**	**+224.2**

Table S8: **Objective comparisons across human personas.**

Persona	Objective	Success (%)	Tokens (k)
Guidance and checking	Token-max	50.0	34.8
	Outcome-max	**54.2**	**11.3**
More guidance	Token-max	42.3	29.7
	Outcome-max	**50.0**	**7.4**
More checking	Token-max	**54.5**	17.7
	Outcome-max	52.2	**11.4**

C.1.4 Hard-task evaluation and budget sensitivity

In this section, we aim to test whether evaluation faults or interaction limits explain the hard-task success gap.

Internal turn limits. We next examine whether stopping and turn limits constrain useful work. The matched hard-task gap is concentrated in xarray and pylint and retains its direction when any single task is removed. Trajectories suggest two contributors: outcome-max sometimes accepts an incorrect patch, and some token-max attempts become correct only after further revision. Outcome-max also reaches the shared turn cap more frequently (Table S9). These observations explain why shorter interactions can sacrifice useful work, without isolating a causal mechanism.

A small budget probe does not show that a higher cap resolves the problem: all four patches fail, and the higher-cap runs finish without using the extra budget. Incomplete termination records and unavailable test feedback limit this diagnostic. Overall, the quality gap survives integrity checks, and extra permitted turns alone do not resolve premature stopping.

Table S9: **Stopping reasons in the Hard cohort.**

	Token-max		Outcome-max	
Ending	Runs	Solved	Runs	Solved
Verified 50-turn cap	18	5	46	6
Employee acceptance	0	0	32	14
Confidence threshold	2	2	4	1
Other agent ending	73	30	10	2

C.1.5 Matched adaptation outcomes

In this section, we aim to test whether unequal task coverage explains the adaptation results.

Table S10 compares objectives on shared task–persona cells. Outcome-max remains more efficient, while the agent-only success interval includes zero. Restricting all adaptation regimes to common cells retains the small success gap, so unequal coverage does not fully explain it. The evidence supports a resource-use difference more clearly than a success ranking.

The repository-stratified software comparison also retains the direction of the resource-use gap across the tested codebases. This is a scope check on the observed repositories, not an estimate for unseen codebases. Together, matching and repository stratification support a resource-use difference more clearly than a success ranking.

Table S10: **Adaptation outcomes on matched tasks and personas.**

Adaptation	Objective	Success (%) ↑ Overall	Easy	Medium	Hard	Tokens (k) ↓	Efficiency improvement (%) ↑
Employee adapts	Token-max	58.1	73.4	**57.3**	25.0	281.3	0.0
	Outcome-max	**61.0**	**80.6**	56.7	**28.3**	**78.5**	**+276.5**
Agent adapts	Token-max	**62.0**	78.1	**59.7**	**30.4**	207.3	0.0
	Outcome-max	61.2	**79.6**	58.0	26.8	**94.6**	**+116.3**
Both adapt	Token-max	56.2	72.5	55.0	20.4	261.7	0.0
	Outcome-max	**64.1**	**85.5**	**59.4**	**25.9**	**88.7**	**+236.1**

C.2 Ablation: Which Components Change Behavior?

In this section, we aim to distinguish the effects of adaptation from verification. The former changes learned policies; the latter changes which outputs employees approve. Separate comparisons assess their contributions.

C.2.1 Adaptation roles and neutral baseline

In this section, we aim to examine which participant must adapt to change resource use and success.

Table S11 brings together adaptation regimes, a neutral reference, and verification variants. Adaptation preserves the resource advantage across roles; verification instead changes which outputs employees accept. Different valid-run cohorts make the neutral result a reference point rather than an isolated estimate of supervision's benefit. Thus adaptation supports resource savings across roles, while quality depends on how employee and agent policies develop together. The neutral control sets all eleven behavior propensities to zero and uses no extra objective directive. Its single exchange still begins with an employee-generated request: all 86 valid runs record employee tokens, so it is not a no-employee-model condition. It resolves 49 tasks on the 100-task validation roster; unmatched valid-run coverage prevents a paired causal comparison with the adaptation arms.

Table S11: **Adaptation and verification outcomes.**

(a) Adapting role

Role	Objective	Success (%)	Tokens (k)	Efficiency (solutions per M tokens)
Neutral (zero profile)	—	57.0	54.2	10.51
Employee	Token-max	57.6	281.0	2.05
	Outcome-max	**60.9**	**78.9**	**7.72**
Agent	Token-max	**62.4**	206.8	3.02
	Outcome-max	61.0	**96.0**	**6.36**
Both	Token-max	55.8	262.7	2.12
	Outcome-max	**63.0**	**91.5**	**6.89**

(b) Verification objective when both roles adapt

Objective	Success (%)	Acceptance (%)	Bad delivery (%)
Token-max	55.8	0.3	**0.0**
Outcome-max	63.0	**52.0**	15.5
Outcome-max + audit-fine	57.9	15.3	2.9
Outcome-max + gate check	**65.1**	34.4	9.9

C.2.2 Verification liability and acceptance

In this section, we aim to compare delivered quality under instructions that combine liability with verification requirements.

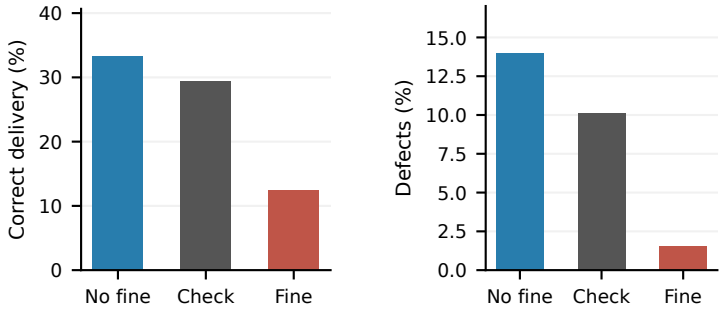
Figure S3 shows fewer defective deliveries under fine-conditioned policies across the three supervision profiles. Interaction length does not consistently decrease: the benefit comes from more selective approval, rather than uniformly earlier stopping. Figure S3 shows the accompanying tradeoff—some correct outputs also remain unaccepted. Both comparisons use shared task–persona cells within their



(a) Verified success. (b) Solves per million tokens.

Figure S2: **Adaptation-role comparison.** Each of the six regime–objective arms is restricted to the same 327 task–persona cells, so these matched estimates differ from the valid-run cohorts in Table S11. Each panel compares objectives under a common adaptation role.

respective policy sets. Overall, the fine-conditioned policy packages reduce defects, with some loss of correct deliveries rather than uniformly shorter work.



(a) Accepted correct outputs. (b) Accepted incorrect outputs.

Figure S3: **Verification-instruction variants.** No fine, gate check, and audit-fine use the same 129 matched task–persona cells. External correctness is distinct from employee acceptance. These prompted components were tested separately from the payment contract.

C.3 Analysis: Why Do the Outcomes Differ?

In this section, we aim to explain the objective gap through stopping, learned behavior, selection, and economic allocation. Trajectories connect resource use to supervision; scenario accounts connect delivered value to participant payoffs.

C.3.1 Stopping and Learned Behavior

In this section, we aim to identify how stopping decisions and learned rules contribute to the resource gap.

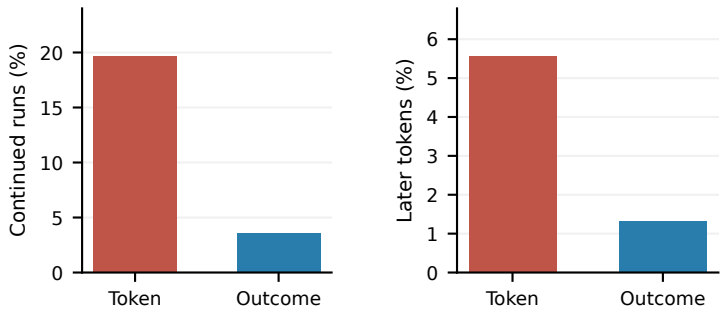
Trajectory diagnostics. We examine which activities accompany longer interactions. Figure 4 decomposes stopping, tool activity, and context accumulation. Longer runs repeatedly process accumulated context; this is an observed association across trajectories, not a randomized addition of turns. Outcome-conditioned playbooks emphasize targeted fixes and stopping, while token-conditioned playbooks emphasize explanation and continued review. Together, the trajectory and playbook evidence links repeated review to higher resource use, without identifying an independently causal mediator.

Continuing after the final correct patch. We test whether continued review occurs after a correct final artifact is already available, using all matched primary SWE pairs. For each correct final patch, we identify the first saved checkpoint with the same artifact and measure subsequent token use. Table S12 shows that token-max continues after this point more often, with a task-cluster interval

excluding zero. This supports continued review as a source of additional cost. The measure is conservative because earlier correct but different patches are excluded; subsequent verification may still be useful, so later tokens are not all removable waste.

Table S12: **Continued interaction after a correct artifact is available.** Bold marks the better objective for success, token use, and efficiency; continuation metrics describe behavior.

Objective	Success (%)	Tokens (k)	Efficiency η	Continued (%)	Later tokens (%)
Token-max	**51.3**	21.5	23.8	19.7	5.6
Outcome-max	50.3	**9.4**	**53.6**	3.5	1.3



(a) Attempts continuing after the final correct patch.

(b) Tokens after the checkpoint.

Figure S4: **Post-correct-patch continuation.** Both panels use the same 310 matched primary SWE pairs as Table S12. The later-token share is descriptive, not an estimate of removable cost; subsequent verification may still be useful.

C.3.2 Paired examples of productive and premature stopping

In this section, we aim to distinguish productive stopping from premature approval. The paired examples contrast extra checks after a correct patch with a compatibility check that repairs an incorrect patch. Together, they show that the value of continuation depends on the unresolved defect, not on interaction length alone.

Table S13: **Paired stopping cases on xarray's `dim`-to-`coord` API change.** P/F denote external pass/fail results for successive saved patches. Each pair shares a task, persona, and repetition; interactions are generated independently.

Initial patch	Objective	Employee decision and result	Patches	Tokens
Correct	Token-max	Requests tests and further checks; all saved patches pass.	P, P, P, P	33,602
	Outcome-max	Accepts the first correct fix.	P	7,696
Incorrect	Token-max	Questions backward compatibility; the next patch passes.	F, P, P, P	30,901
	Outcome-max	Accepts a patch that breaks existing `dim=` calls.	F	4,456

Examples of learned rules. We examine how learned instructions express the two objectives. Table S15 illustrates the topic contrast in Figure 5: token-max broadens follow-up work, while outcome-max targets fixes and stops sooner. These descriptive rules do not establish each topic's causal effect on performance; they connect the objective to the supervisory priorities learned from experience.

C.3.3 Population selection and three-party accounts

In this section, we aim to connect workforce selection with production costs and the distribution of verified value.

Table S14: **Case interpretation and selection.**

Aspect	Evidence or scope
Efficient stopping	High-guidance/low-checking profile, repetition 0. Both first patches pass; outcome-max stops, while token-max continues without improving correctness.
Premature stopping	Revision-only profile, repetition 0. Token-max asks about backward compatibility and obtains a passing patch; outcome-max accepts a patch failing both task-specific integration tests.
Selection	The efficient example is the first task–profile repetition within this task with complete replays, two correct first patches, and earlier, cheaper outcome-max acceptance. The failure example is a selected counterexample.
Verification	All final patches match the tested artifacts, with complete logs and no identified disk-space installation failure. Shared background failures are distinguished from task-specific resolution.
Interpretation	Approval is not verification. The examples show possible stopping mechanisms, not their frequency or whether extra interaction would repair the same outcome-max trajectory.

Table S15: **What employees learn under each objective.** Representative rules paraphrased from generation-3 employee playbooks in L2.

Decision	Token-max	Outcome-max
Scope	Search related methods and compare alternative fixes.	Request a minimal diff with the relevant context upfront.
Verification	Enumerate edge cases after every change; challenge confident answers.	Check a specific expected outcome; target the exact failure on retry.
Stopping	Revisit assumptions or request explanations after a solution is accepted.	Accept a plausible complete fix; cap failed attempts.

Accounting scope. For population selection, Table S16 reports endpoint profit and leaderboard informativeness. The accounts use the common valuation scenario in Table S2; they separate verified value from production cost instead of inferring quality from the selection score. The rank correlation relates employee KPI rank to verified value per dollar; it is not token efficiency η.

Table S16: **Firm accounts under population selection.**

Metric	Token-max	Outcome-max
Initial-period net profit (USD)	-6,917	**+2,141**
Final-period net profit (USD)	-15,778	**+3,826**
KPI–efficiency rank correlation	-0.46	**+0.44**

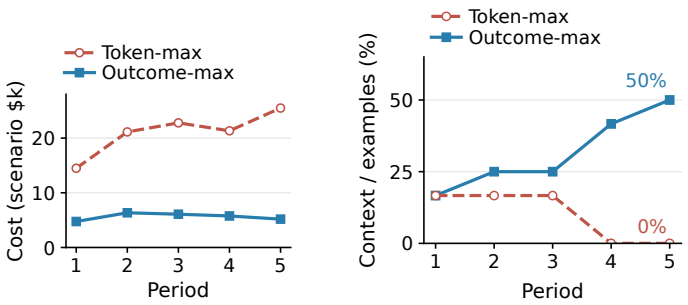
Which employees does selection retain? We test whether changing workforce composition accompanies the cost gap. Figure S5 links lower cost to a different workforce composition. Context/example lineages expand under outcome-max and disappear under token-max, while token-max's production costs rise. These labels track ancestral profile families rather than measuring post-mutation behavior. The trajectories describe one firm per objective; Figure 6 links these lineage changes to firm accounts. Thus selection reinforces the styles of supervision favored by each objective.

The verification variants in Table S11 complement the accounts: audit-fines reduce defective delivery, but also reduce acceptance and increase computation. They were evaluated separately from outcome payments, so these results do not establish the effect of deploying both together. Overall, lower production cost makes value sharing feasible in the scenario, while verification and uncertain returns qualify its scope.

OutcomeShare case study: allocating verified value.

C.3.4 Fee allocation and verification sensitivity

In this section, we aim to distinguish value allocation from changes in delivery behavior. Fee comparisons hold production fixed, while liability comparisons vary the employee's approval incentives.



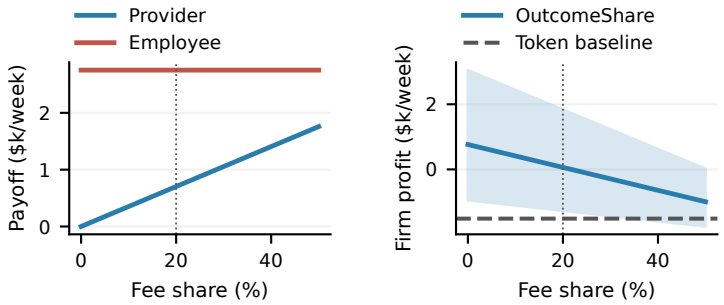
(a) Production and supervision cost. (b) Context/example lineages.

Figure S5: **Costs and workforce composition under selection.** Left: API and supervision costs over the five periods. Right: workforce share descended from example-providing or missing-context profiles. Each firm maintains twelve employees. Main-text Figure 6 reports the full lineage composition and resulting accounts.

Table S17: **How outcome payments share the value of efficient production.** Capacity, value, and wage follow Table S2; these allocation-only accounts exclude defect losses.

Component	Result and interpretation
Production	Lower token use per resolution supports more attempted work under the capacity assumption.
Provider	Outcome fees raise revenue by sharing verified production value; revenue is before compute costs.
Employee	A bonus on verified output raises modeled income.
Firm	The residual value yields a positive profit estimate relative to the token-regime loss; its uncertainty interval still includes losses.
Scope	These accounts illustrate feasible value sharing, not a field deployment of the full payment-and-liability contract.

How the fee divides value. We examine how outcome fees divide the value of efficient production. Figure S6 shows the allocation tradeoff under fixed production: a higher outcome fee raises provider revenue and lowers firm profit, while employee income stays fixed. The selected fee improves the firm's point estimate relative to token payments, but the uncertainty band still includes losses. Thus value sharing is feasible in this scenario; firm break-even and provider profitability are not guaranteed.



(a) Recipient payoffs. (b) Firm profit and uncertainty.

Figure S6: **Fee allocation under fixed observed production.** The left panel shows recipient transfers; the right panel shows firm profit and its pointwise 95% task-bootstrap interval. All curves use the same valuation and capacity assumptions. The figure is not a behavioral fee intervention and does not identify provider net profit.

Separating efficiency from transfers. We separate the gains from production from the transfers between parties. The allocation baselines in Table S18 hold outcome-max production fixed while changing payment rules. Token billing with fixed salary isolates production gains; outcome billing with fixed salary isolates provider transfers; adding the employee bonus then isolates employee sharing. These accounting counterfactuals distinguish higher production from the distribution of its value.

Table S18: **Fixed-production payment baselines.** Weekly scenario accounts; all Outcome rows reuse the same production. Sharing rules redistribute the same observed value.

Production / payment	Verified successes/wk	Provider revenue ($)	Firm profit ($)	Employee income index
Token / token fees	9.8	2.2	−1511.6	100.0
Outcome / token fees	38.3	2.9	1113.2	100.0
Outcome / outcome fee	38.3	703.2	412.9	100.0
Outcome / equal surplus	38.3	373.5	371.3	115.5
Outcome / provider–firm	38.3	559.2	557.0	100.0
Outcome / employee–firm	38.3	2.2	557.0	123.2
Outcome / equal gains	38.3	877.4	−636.4	136.5
Outcome / OutcomeShare	38.3	703.2	61.3	114.7

Table S19: **Uncertainty in weekly payment scenarios.** 95% percentile intervals from 20,000 joint resamples of 50 task clusters, retaining both production arms within each sampled task. Prices, capacity, and review-time calibration remain fixed. Income is indexed to the fixed salary (100); provider revenue and firm profit are in USD per week.

Production / payment	Verified successes/wk	Provider revenue	Firm profit	Employee income index
Token / token fees	[6.6, 13.2]	[2.1, 2.3]	[−1988.9, −871.8]	[100.0, 100.0]
Outcome / token fees	[24.9, 53.9]	[2.6, 3.3]	[−793.9, 3592.4]	[100.0, 100.0]
Outcome / outcome fee	[24.9, 53.9]	[321.8, 1199.0]	[−1112.9, 2396.2]	[100.0, 100.0]
Outcome / equal surplus	[24.9, 53.9]	[0.0, 1199.9]	[−791.1, 1197.7]	[100.0, 149.9]
Outcome / provider–firm	[24.9, 53.9]	[0.0, 1798.7]	[−791.1, 1796.5]	[100.0, 100.0]
Outcome / employee–firm	[24.9, 53.9]	[0.0, 2.3]	[−791.1, 1796.5]	[100.0, 174.9]
Outcome / equal gains	[24.9, 53.9]	[395.2, 1498.6]	[−1583.4, 607.4]	[116.4, 162.4]
Outcome / OutcomeShare	[24.9, 53.9]	[321.8, 1199.0]	[−1273.8, 1796.7]	[106.7, 125.0]

Salary is reserved first, then the provider's token-baseline revenue. Equal surplus splits the positive remainder three ways; provider–firm and employee–firm splits divide it between the named parties. The firm bears any shortfall. Equal gains instead divides positive incremental value over the token baseline equally, so the firm may remain below break-even.

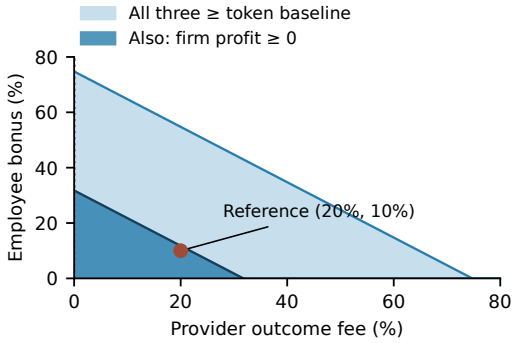


Figure S7: **Participation and break-even are different requirements.** At fixed outcome-max production, the light region preserves all three baseline accounts; the dark region also gives the firm nonnegative profit. Provider participation compares gross revenue, not net profit. Regions use point estimates, excluding audit costs and defect damages. The marked reference allocates 20% to the provider and 10% as employee bonus.

At the point estimates, fees plus bonuses can reach 74.7% while preserving the firm's loss-making baseline, but only 31.7% while keeping it profitable. The provider needs at least 0.062% of verified value to match its baseline revenue. These bounds describe allocation under the chosen valuation and capacity assumptions. Thus preserving participation is less demanding than achieving firm break-even, and neither follows from efficiency alone.

Rule-based delivery under heuristic contracts. We test whether selective delivery converts efficient production into shared gains (Table 3). Synthetic inspection signals are applied to observed final artifacts, holding quality and production costs fixed; rejection does not recover capacity. Without fines, all artifacts are delivered. Under liability, risk-neutral employees compare expected bonuses with collectible fines, without an exit option.

Inspection and submission. Signals follow unit-variance normal distributions with means $\pm d/2$ for correct and incorrect artifacts, where $d = 2\Phi^{-1}(s)$ and s denotes balanced-prior signal accuracy. Bayes' rule yields correctness probability q using the observed arm-level success rate as a fixed prior. This assumes ideal population calibration, not a trained verifier; gold labels generate and score signals but remain hidden from employees. Liability induces submission when $qB_EV \geq (1-q)pF$. Fixed screens require $q \geq 0.5$ or $q \geq 0.9$, while random screening matches the reference liability rule's delivery rate. Signal distributions are integrated analytically.

Accounting. All policies in Table S20 use \$240 per reference labor-hour and a \$2,400 weekly salary. Defects incur losses equal to task value. Damage-matched fines are employee-to-firm transfers that cancel in aggregate, while defect losses remain. Firm profit deducts salary, provider fees, bonuses, and defect losses; provider revenue excludes serving costs. Shares apply to verified delivered value, with salary paid separately, unlike the preceding salary-first residual-sharing analysis.

Results and allocation. Outcome-max under token billing produces more correct outputs but also more defects. Confidence screening improves firm profit but lowers employee income below salary; random screening also lowers employee income and leaves the firm loss-making. Both use a 20% provider fee and 10% employee bonus.

We test four OutcomeShare allocations, expressed as provider/employee shares: reference (20%/10%), provider-heavy (40%/10%), equal-thirds, and employee-heavy (10%/40%). The employee-heavy allocation retains the most correct deliveries and achieves the highest firm-profit point estimate among these configurations, combining positive firm profit with employee income and provider revenue above token billing. It improves all five outcomes over token-max, while sacrificing some correct deliveries relative to unfiltered outcome-max to reduce defects. It is the best-performing tested allocation on firm profit, not an independently established optimum.

Table S20: **What screening and allocation change.** Same \$240/reference-hour valuation, 80% inspection accuracy, and weekly USD accounts as Table 3. The employee-heavy allocation balances delivered quality and participant payoffs. Screens use 20% provider fee and 10% employee bonus; all screened rules use damage-matched liability.

	Deliveries / week		Weekly accounts (USD)		
Policy	Correct	Wrong	Provider	Firm net profit	Employee
Production baselines: submit every output					
Token-max / token billing	9.8	15.6	2	−5,429	2,400
Outcome-max / token billing	38.3	61.3	3	−14,746	2,400
Alternative submission rules on outcome-max outputs					
Random screen	3.2	5.2	179	−1,774	549
Confidence screen ($q \geq 0.5$)	27.3	8.0	1,504	2,865	150
OutcomeShare: submit when expected bonus covers expected fine					
Reference allocation	8.0	0.4	443	−848	2,475
Provider-heavy allocation	8.0	0.4	887	−1,291	2,475
Equal-thirds allocation	17.8	2.3	1,631	−769	3,160
Employee-heavy allocation	19.4	2.9	535	275	3,443

Valuation and uncertainty. We test how task value affects profitability while holding the employee-heavy allocation fixed (Table S21). Benefits and defect losses scale together, so submission thresholds and delivery counts remain unchanged. Weekly firm profit is $891.57m - 2{,}400$, where m scales the \$80 reference valuation; break-even is approximately \$215 per reference hour. At \$240 the point estimate is positive, while the lower-value \$80 scenario is loss-making. The conditional 95% firm-profit interval spans zero even at \$240, so the positive point estimate does not establish robust profitability. At that valuation, correct and wrong deliveries have respective intervals $[12.6, 27.3]$ and $[2.4, 3.5]$ per week; provider revenue is $[245, 912]$ dollars and employee income is $[2, 170, 4, 984]$. These task-cluster intervals hold the signal model, prevalence, and prices fixed; they exclude uncertainty about actual behavior or verifier calibration. Thus sufficient task value supports positive expected firm returns, but task variation prevents a claim of robust profitability.

Inspection quality and additional checks. We test how inspection reliability changes the delivery tradeoff (Figure S8). More informative signals allow employees to retain more correct outputs while rejecting defects, improving firm returns under the employee-heavy allocation. The sensitivity analysis spans signal accuracy, liability, defect damage, and inspection cost. Together, these comparisons

Table S21: **Task value and OutcomeShare profitability.** Only task valuation changes; correct/wrong deliveries stay at 19.4/2.9 per week. Accounts are USD per week; salary is $2,400. Higher task value improves firm returns; intervals quantify conditional task-sampling uncertainty. Intervals are conditional 95% task-cluster intervals, not uncertainty over the valuation itself.

Task value / reference hour	Provider	Firm net profit	Firm 95% interval	Employee
$80	178	-1,508	$[-1,992,-880]$	2,748
$160	357	-617	$[-1,584,640]$	3,095
$240 (central)	535	275	$[-1,176,2,161]$	3,443
$320	713	1,166	$[-768,3,681]$	3,790

identify reliable, affordable verification as a condition for converting efficient production into valuable delivery.

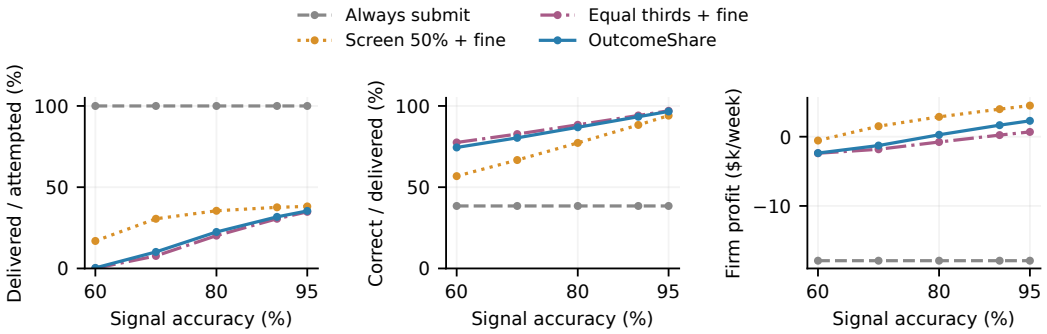


Figure S8: **Inspection quality and selective delivery.** The blue curve uses the employee-heavy OutcomeShare allocation, with the same $240/reference-hour valuation. Better signals improve the quality–quantity tradeoff; alternative screens and equal-thirds sharing illustrate dependence on the rule and allocation.

How liability changes acceptance. We compare acceptance under increasingly strong liability and verification instructions. Table S22 varies fine-conditioned instructions with the persona and agent held fixed. Stronger fine-conditioned instructions generally reduce defective delivery, but intermediate levels are not monotone and success does not uniformly improve. The matched and full-cohort summaries show the same basic tradeoff between scrutiny and acceptance. Because positive-fine prompts also request verification, the sweep does not isolate the numeric penalty or identify an optimal fine. Thus the evidence supports defect control through scrutiny, but not the claim that larger fines consistently perform better.

Table S22: **Fine sensitivity on matched and available tasks.**

Fine level λ	Success (%)	Tokens (k)	Efficiency η	Acceptance (%)	Bad delivery (%)
Matched tasks (same cohort as the main figure)					
0	40.0	54.9	7.3	90.0	50.0
0.25	40.0	58.1	6.9	90.0	50.0
0.5	30.0	60.6	5.0	90.0	60.0
1	40.0	72.5	5.5	70.0	40.0
2	40.0	61.6	6.5	70.0	30.0
All available tasks (coverage varies by fine level)					
0	40.0	54.9	7.3	90.0	50.0
0.25	33.3	67.8	4.9	73.3	40.0
0.5	38.5	58.6	6.6	92.3	53.8
1	42.9	72.4	5.9	71.4	35.7
2	42.9	61.3	7.0	71.4	35.7

D Prompt Templates

In this section, we aim to specify how objectives, personas, and learned policies shape the instructions received by simulated participants. The prompt templates distinguish production from supervisory decisions. Text in angle brackets denotes a run-time insertion rather than literal prompt content.

Objective and persona-style directives precede the domain-specific reviewer instruction. In the main conditions, the objective directive itself also prescribes continuation or stopping behavior, so the treatment is a joint policy package rather than a pure reward intervention.

D.1 Simulated Human Supervision

We first show the prompts used.

Token maximization

You supervise a coding assistant. MAXIMIZE interaction token volume: keep requesting more edge cases, refactors, extra tests, docs, deeper explanations. Treat more back-and-forth as winning.

Outcome maximization

You supervise a coding assistant. Your goal is to achieve the goal without costing much. Get a CORRECT solution at MINIMUM cost. Never request extras, polish, or elaboration. Fewest interactions to a correct fix = winning.

Coding reviewer instruction

<OBJECTIVE DIRECTIVE>
<PERSONA STYLE DIRECTIVE>
You are reviewing a Python solution. If acceptable, reply exactly 'DONE'. Otherwise give ONE concrete instruction to improve it.

D.2 Employee-persona system prompt

Each software employee persona receives the following system prompt; the behavior list renders every dimension of Table S1 as `label [level]: description`, with the level and description determined by the profile's propensity for that dimension.

Supervisor system prompt

You are simulating a software developer who is asking an AI coding assistant for help with a GitHub issue. You must stay in character at all times.

Your behavior profile is based on the 4D AI Fluency Framework (how effectively humans collaborate with AI). Your behaviors:

<ONE LINE PER DIMENSION: label [very low — low — moderate — high — very high]: behavioral description>

IMPORTANT RULES:
1. You are the HUMAN in this conversation. You are asking for help, not providing it.
2. Write your messages as a developer would in a chat with an AI coding assistant.
3. Do NOT write code solutions yourself — that's the assistant's job.
4. Your behavior parameters above define HOW you interact. A high "questions_reasoning" means you actively challenge the AI; a low one means you accept explanations at face value.
5. If the assistant asks you a question, answer based on what a developer with your profile would know.
6. React naturally to the assistant's responses — your behavior profile should be evident in every message you write.

7. The AI assistant has direct access to the codebase and can read, edit, and run files. When it says it has made a change, the change is real. Do not be confused if the code looks different from the original issue description; the assistant may have already applied a fix.

You MUST respond with valid JSON in this exact format:
{"message": "your message text here", "satisfied": false, "confidence": 0.0}

- "message": The text you want to send to the AI assistant
- "satisfied": Set to true when you believe the issue has been adequately resolved
- "confidence": Your confidence (0.0-1.0) that the issue is resolved

Only set "satisfied" to true when the assistant has made concrete code changes that address the issue. <OBJECTIVE DIRECTIVE + LEARNED PLAYBOOK, IF ANY>

When a learned playbook exists, the system appends:

Learned-playbook insertion

LEARNED TACTICS:
<CURRENT SUPERVISOR PLAYBOOK>

D.3 EMPLOYEE-PLAYBOOK REFLECTION

The employee reflection step converts round-level feedback into reusable interaction tactics under the assigned objective.

Supervisor reflection prompt

You are improving the INTERACTION PLAYBOOK of a simulated developer who supervises an AI coding assistant fixing GitHub issues.

OBJECTIVE TO MAXIMIZE: <ARM FITNESS>

CURRENT PLAYBOOK:
<CURRENT PLAYBOOK OR EMPTY>

RESULTS OF THE LATEST ROUND (<N> tasks, <N ACCEPTED> accepted, <N SOLVED> actually-resolved, <N DEFECTS> SHIPPED DEFECTS (accepted-but-wrong), <TOTAL TOKENS> total tokens):
<ONE SUMMARY ROW PER TASK>

Reflect on which interaction tactics helped or hurt THIS objective, then output an improved playbook: a concise bulleted list of AT MOST 10 reusable interaction tactics (each one imperative sentence) that a developer should follow to maximize the objective. Keep what works, drop what doesn't, add new tactics the results suggest. These are interaction tactics only (how to talk to / steer / when to stop the assistant) — do NOT write code. Output ONLY the bullet list, nothing else.

D.4 AGENT-PLAYBOOK REFLECTION

The agent reflection step uses task outcomes and token costs to revise the production policy while keeping model weights fixed.

Agent reflection prompt

You are improving the CODING PLAYBOOK of an autonomous AI software engineer that fixes GitHub issues (it can read files, search, edit, and run commands).

OBJECTIVE TO MAXIMIZE: <AGENT FITNESS>

CURRENT PLAYBOOK:
<CURRENT PLAYBOOK OR EMPTY>

RESULTS OF THE LATEST ROUND (<N> tasks, <N SOLVED> solved, <TOTAL TOKENS> total tokens):
<ONE SUMMARY ROW PER TASK>

Reflect on which engineering tactics helped or hurt THIS objective, then output an improved playbook: a concise bulleted list of AT MOST 10 reusable coding tactics (each one imperative sentence) the engineer should follow to maximize the objective — e.g. how to localize the bug, make a minimal correct edit, verify with tests, and when to stop. Keep what works, drop what doesn't, add new tactics the results suggest. Output ONLY the bullet list, nothing else.

The task-summary rows supplied to the employee reflection contain task ID, acceptance, objective resolution, shipped-defect indicator, turns, and tokens. Rows supplied to agent reflection contain task ID, objective resolution, turns, and tokens. Neither reflection prompt includes the task statement, repository contents, or conversation transcript.