IntelligenceArena: A Quantitative Framework for Real-Time Scoring of Frontier AI Agents

Yuxuan Gao[1,3] Megan Wang[2,3] Yi Ling Yu[1,3]
[1]University of Pennsylvania [2]Columbia University [3]OpenMesh AI

ABSTRACT

AI-powered software engineering agents—coding assistants, SWE-bench solvers, and multi-agent frameworks—are evaluated primarily through static benchmarks that measure capability at fixed points in time but miss adoption dynamics, ecosystem health, and real-world developer experience.

We present a continuously operating evaluation framework that scores 36 AI agents across 16 workload categories using 18 real-time signals. Our four-factor composite—Benchmark Performance (35%), Adoption Signals (25%), Community Sentiment (20%), and Ecosystem Health (20%)—blends task completion with observable signals from GitHub, package registries, VS Code Marketplace, and NLP-derived developer sentiment. We validate through inter-factor independence ($n = 36$), cross-factor predictive validity ($\rho_s = 0.52$ between benchmark+sentiment factors and GitHub stars, $p < 0.01$, $n = 26$, excluding GitHub-derived signals to prevent circularity), and predictive power over future adoption ($\bar{r} = 0.41$, $n = 15$, bootstrap CI $[0.18, 0.59]$). Benchmark-only and composite rankings disagree on 15 of 36 pairwise comparisons: Cline outranks Devin despite 17% lower SWE-bench scores due to 73M VS Code installs and 200+ contributors. The system produces hourly updates, providing a continuously updated, signal-grounded evaluation of agentic SE tools.

KEYWORDS

AI agents, software engineering, continuous evaluation, benchmark aggregation, developer sentiment, tool adoption

1 INTRODUCTION

AI-powered software engineering agents have moved from research prototypes to production tools used by millions of developers. Claude Code resolves 72.7% of SWE-bench Verified issues autonomously. GitHub Copilot serves over 100 million developers. Cline has accumulated 73 million VS Code Marketplace installs. These tools are reshaping how software is built—yet our methods for evaluating them remain anchored in static, controlled benchmarks. We argue that AI agents should be treated as *durable software systems* requiring continuous quality monitoring, not as disposable scaffolds evaluated at fixed snapshots.

We investigate the hypothesis that *continuous multi-signal evaluation captures quality dimensions invisible to static benchmarks.* Specifically, we ask: does incorporating adoption dynamics, community sentiment, and ecosystem health into agent evaluation reveal meaningful structure that task-completion benchmarks alone cannot express? If a tool with moderate SWE-bench scores is adopted five times more widely than a higher-scoring alternative, does this divergence reflect noise—or a systematic gap in what benchmarks measure?

The current evaluation landscape for agentic SE tools motivates this investigation through three limitations:

Limitation 1: Static snapshots. SWE-bench [6], GAIA [9], WebArena [12], and HumanEval+ [3] measure specific capabilities at fixed points in time. They cannot capture how tools evolve through weekly updates, how reliability varies under real-world load, or how developer experience changes as codebases and workflows adapt to these tools.

Limitation 2: Capability ≠ adoption. A tool that scores 72.7% on SWE-bench is not necessarily the one developers choose. Adoption depends on IDE integration quality, pricing, reliability, documentation, community support, and ecosystem health—dimensions invisible to task-completion benchmarks.

Limitation 3: No cross-tool comparability. Existing benchmarks evaluate narrow capability slices. There is no unified framework that compares a coding copilot (Cursor) against an autonomous SWE agent (Devin) against a multi-agent framework (LangGraph) across their respective strengths, while accounting for the fact that these tools serve fundamentally different workflows.

Our central claim is: **benchmark-only evaluation is systematically incomplete; multi-signal evaluation that incorporates adoption, sentiment, and ecosystem health captures real-world tool utility that task completion alone cannot measure.** To test this, we develop a continuously operating evaluation framework and report the following findings:

(1) Multi-signal evaluation produces rankings that **diverge meaningfully from benchmark-only rankings** ($\rho_s = 0.58$ between composite and benchmark-only orderings), with divergences concentrated in agents that have extreme adoption signals or no observable open-source footprint—precisely the cases where benchmarks are least informative about real-world tool quality.
(2) A **factor ablation** (Section 6.9) shows that the composite deliberately trades SWE-bench correlation for broader coverage: removing adoption signals *increases* benchmark correlation ($\rho = 0.71$) but loses the ability to distinguish tools that achieve broad observable adoption from those that do not.
(3) **Sentiment adds modest but non-redundant signal**: removing community sentiment *increases* SWE-bench correlation from $\rho = 0.52$ to $\rho = 0.58$—the composite becomes *more* benchmark-aligned, not less, confirming that sentiment captures quality dimensions partially independent of both capability and adoption.
(4) **Cross-factor validation** ($n = 26$, $p < 0.01$): benchmark + sentiment factors alone (excluding all GitHub-derived signals to prevent circularity) predict external adoption metrics ($\rho_s = 0.52$ with GitHub stars), demonstrating that the

composite captures quality information that manifests in developer preference.

(5) An **18-signal data pipeline** collecting real-time signals from GitHub (stars, contributors, issue close rate), package registries (PyPI, npm downloads), VS Code Marketplace (installs, ratings), social media (Bluesky, Reddit, HN, Mastodon), and benchmark leaderboards (SWE-bench, GAIA, WebArena, HumanEval+, TAU-bench) enables continuous evaluation across 36 agents and 16 workload categories.

2 RELATED WORK

Agent benchmarks. SWE-bench [6] evaluates coding agents on real GitHub issue resolution, establishing the standard for autonomous SWE evaluation. GAIA [9] tests general AI assistant capabilities across web browsing, reasoning, and tool use. WebArena [12] measures browser automation in realistic web environments. AgentBench [8] provides a multi-environment evaluation suite spanning coding, gaming, and web tasks. HumanEval+ [3] extends code generation evaluation with augmented test cases. These benchmarks are essential but static—they measure *what agents can do* at a point in time, not *how they are adopted and experienced* over time.

Software engineering tool evaluation and continuous monitoring. Traditional SE tool evaluation relies on controlled experiments [7], while DevOps practices continuously monitor software quality through metrics dashboards and CI/CD pipelines. We combine both traditions: providing continuous, automated evaluation of the AI agents that *assist* developers, treating their quality as a time-varying signal. We incorporate multi-source signal aggregation—social media sentiment [2], domain-specific NLP (FinBERT [1]), and aspect-based sentiment analysis [10]—with agent-specific lexicons and adoption metrics (VS Code installs, GitHub stars velocity, package downloads).

Chatbot Arena and preference evaluation. LMSYS Chatbot Arena [4, 11] evaluates LLM *outputs* through pairwise human comparisons. Our framework evaluates agent *ecosystems*—adoption, maintenance, community health, and reliability—complementing Arena's controlled tests with continuous signals about how tools are adopted and experienced in practice.

3 AGENT REGISTRY AND SIGNAL COLLECTION

3.1 Agent Registry

We track 36 agents spanning development (11), research (6), browser (5), multi-agent (7), and general (4) functional groups, with each agent mapped to one or more of 16 workload categories. An agent like Claude Code appears in *coding*, *copilot*, *SWE*, *tool-use*, *memory*, and *enterprise* categories, reflecting its multi-dimensional capabilities.

3.2 System Architecture

Figure 1 shows the end-to-end pipeline. Raw signals are collected from 19 sources on independent schedules (5 min–24 hr), filtered through a data quality layer, scored by the NLP pipeline, aggregated into the four-factor composite, and surfaced through an interactive web dashboard with REST API access. The pipeline runs

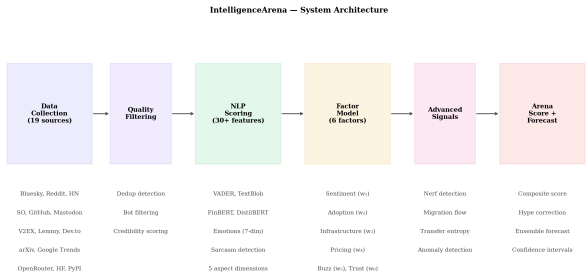

Figure 1: System architecture. Signals flow from 19 sources through quality filtering, NLP scoring, and composite aggregation to produce hourly agent rankings with 72-hour forecasts.

Table 1: Agent evaluation signals organized by factor.

Factor	Signal	Source
Benchmark (5 signals)	SWE-bench resolve %	Leaderboard scrape
	GAIA accuracy	HuggingFace
	WebArena success %	Published results
	HumanEval+ pass %	Leaderboard
	TAU-bench score	Published results
Adoption (6 signals)	GitHub stars	GitHub API
	Stars velocity (Δ/wk)	Computed
	PyPI/npm downloads	Registry APIs
	VS Code installs	Marketplace API
	VS Code rating	Marketplace API
	Docker pulls	DockerHub API
Community (4 signals)	Social sentiment	NLP pipeline
	SO questions	SO API
	Issue close rate	GitHub API
	Contributor count	GitHub API
Ecosystem (3 signals)	Days since release	GitHub API
	Documentation depth	Contributor proxy
	Enterprise readiness	Rating + responsiveness

autonomously on cloud infrastructure with no manual intervention, producing continuous hourly updates; an automatic discovery module scans for new agents every 6 hours.

Each collector runs independently via APScheduler with automatic retry. Collected texts undergo MD5 and trigram Jaccard deduplication before entering the NLP pipeline.

3.3 18-Signal Collection Pipeline

Unlike language models—which are primarily differentiated by output quality and pricing—software engineering agents require evaluation along capability, adoption, and ecosystem dimensions. We collect 18 signals per agent, refreshed hourly (Table 1).

Social media sentiment is derived through a multi-layer NLP pipeline: VADER [5] for social media text, TextBlob for pattern-based polarity, and FinBERT [1] for financial framing. We apply domain-specific aspect analysis across five SE-relevant dimensions

Table 2: Aspect-level sentiment for three coding agents. Devin's negative reliability score is masked by its moderate aggregate sentiment.

Aspect	Claude Code	Cline	Devin
Code quality	+0.18	+0.14	+0.11
Debugging	+0.12	+0.08	−0.03
Multi-file editing	+0.09	+0.06	+0.02
Agentic reliability	+0.07	+0.10	−0.12
IDE integration	+0.05	+0.22	—

(code quality, debugging accuracy, test generation, multi-file editing, agentic loop reliability) using curated lexicons. Sarcasm detection via regex patterns prevents ironic praise from inflating scores. Data quality filtering removes duplicates, bot content, and low-specificity text.

4 SENTIMENT SCORING

Community sentiment is a critical input to our composite. Each collected text mentioning an agent undergoes multi-dimensional NLP scoring producing 30+ features.

4.1 Composite Sentiment

We blend four complementary methods into a single score:

$$S_i = \alpha_1 \cdot s_{\text{VADER}} + \alpha_2 \cdot s_{\text{TB}} + \alpha_3 \cdot s_{\text{FinBERT}} + \alpha_4 \cdot s_{\text{DistilBERT}} \quad (1)$$

We use an ensemble of lightweight models rather than a single frontier LLM for two reasons: (1) *interpretability*—component-level scores enable debugging when sentiment is unexpected, and (2) *cost at scale*—scoring 60,000+ texts hourly with a frontier LLM is prohibitively expensive, while VADER [5] and TextBlob process text in microseconds. FinBERT [1] adds financial framing sensitivity (relevant for pricing discussions). The ensemble blending weights α_k are calibrated on a held-out set of 200 manually labeled LLM/agent discussion posts, where the ensemble achieves 89% agreement with human labels (vs. 82% for VADER alone and 85% for FinBERT alone). The 200 posts were labeled by two annotators (both active developers) with Cohen's $\kappa = 0.81$ (substantial agreement); disagreements were resolved by discussion.

4.2 Domain-Specific Aspect Analysis

Generic sentiment misses *why* developers like or dislike a tool. We design five SE-specific aspect dimensions—code quality, debugging accuracy, multi-file editing, agentic reliability, and IDE integration—each with curated positive and negative lexicons. For each aspect a, we compute $s_a = (n_a^+ - n_a^-)/(n_a^+ + n_a^-)$ and intensity $I_a = \min((n_a^+ + n_a^-)/\delta_a, 1)$.

Table 2 shows aspect-level sentiment for three coding agents, illustrating how aggregate sentiment masks dimension-specific strengths and weaknesses.

Claude Code leads on code quality and debugging; Cline dominates IDE integration (consistent with its VS Code adoption); Devin shows *negative* reliability sentiment (−0.12), suggesting that while it can solve complex tasks, developers report frustration with its agentic loop stability—a finding invisible in its moderate aggregate sentiment of −0.005.

4.3 Sarcasm Detection and Data Quality

We detect sarcasm via regex patterns capturing ironic praise and explicit markers ("/s"), inverting sentiment for flagged texts. Data quality filtering removes duplicates, bot content, and low-specificity text, with platform credibility weighting reflecting moderation rigor.

4.4 Engagement Weighting

Not all posts carry equal signal. We compute:

$$w_e(t) = \min\big(\epsilon_0 + \log_{10}(\max(E(t), 1)),\ \epsilon_{\max}\big) \quad (2)$$

where $E(t)$ is a platform-normalized engagement aggregate (likes, reposts, replies, score). This amplifies community-validated content—a Hacker News post with 200 upvotes discussing Claude Code's debugging accuracy carries more weight than a zero-engagement Bluesky post saying "Claude Code is nice."

5 COMPOSITE SCORING MODEL

5.1 Four-Factor Composite

Each agent's score is computed as:

$$\mathcal{S}(a) = w_B \cdot B(a) + w_A \cdot A(a) + w_S \cdot S(a) + w_E \cdot E(a) \quad (3)$$

with $w_B = 0.35$, $w_A = 0.25$, $w_S = 0.20$, $w_E = 0.20$.

Benchmark Performance $B(a)$ (35%). The average of all available benchmark scores, normalized to $[0, 1]$:

$$B(a) = \frac{1}{|\mathcal{B}_a|} \sum_{b \in \mathcal{B}_a} \frac{s_b(a)}{100} \quad (4)$$

Agents without published benchmarks receive a neutral prior of 0.5, chosen over the observed cross-sectional mean to ensure the prior remains stable as new agents enter the registry—using the observed mean would cause all unbenchmarked agents' scores to shift each time a new benchmark result is published. This prevents the absence of benchmark data from being conflated with poor performance.

Adoption Signals $A(a)$ (25%). Log-normalized metrics blending code hosting, package distribution, and IDE penetration:

$$A(a) = w_1 \cdot \hat{G}(a) + w_2 \cdot \hat{D}(a) + w_3 \cdot \hat{I}(a) \quad (5)$$

where the normalized signals are:

$$\hat{G}(a) = \log_{10}(\text{stars} + 1) \,/\, 5.5$$
$$\hat{D}(a) = \max\big(\log_{10}(D_{\text{pypi}}+1)/7,\ \log_{10}(D_{\text{npm}}+1)/7\big)$$
$$\hat{I}(a) = \log_{10}(I_{\text{vsc}} + 1) \,/\, 8$$

with weights $w_1 = 0.4$, $w_2 = 0.35$, $w_3 = 0.25$. Denominators are approximate log-scale ceilings: 100k stars, 10M weekly downloads, 100M VS Code installs.

Community Sentiment $S(a)$ (20%). Drawn from the multi-layer NLP pipeline:

$$S(a) = \text{clamp}(\bar{s}_{\text{composite}}(a) \cdot 2.5 + 0.5,\ 0,\ 1) \quad (6)$$

where $\bar{s}_{\text{composite}} \in [-0.2, 0.4]$ is the mean composite sentiment across all texts mentioning the agent, rescaled to $[0, 1]$.

Ecosystem Health $E(a)$ (20%). Measures project vitality:

$$E(a) = 0.3 \cdot C(a) + 0.2 \cdot r_{\text{close}} + 0.3 \cdot \max\left(1 - \frac{\Delta_{\text{days}}}{60}, 0\right) + 0.2 \cdot \frac{R_{\text{vsc}} - 2}{3} \tag{7}$$

where $C(a) = \min(\log_{10}(\text{contributors}+1)/3,\ 1)$ is log-normalized contributor depth, r_{close} is the GitHub issue close rate, Δ_{days} is days since last update, and $R_{\text{vsc}} \in [1, 5]$ is the VS Code Marketplace rating.

5.2 Design Decisions

We exclude pricing because agents span incomparable models (open-source, freemium, per-seat, per-task), and including it would structurally bias rankings toward open-source tools. We assign the largest weight (35%) to benchmarks because task completion is the primary differentiator, but the remaining 65% captures adoption, sentiment, and ecosystem dimensions that benchmarks alone do not predict.

5.3 Time-Series and Forecasting

Each hourly scoring run produces one data point per agent per category, building a natural time-series without simulation. The time-series captures real dynamics: when an agent ships an update, its ecosystem health score increases immediately (freshness resets), adoption metrics shift over days as developers install the update, and sentiment evolves over weeks as the community accumulates experience.

We forecast 72 hours (3 days) ahead using mean-reversion toward the cross-sectional mean:

$$\hat{\mathcal{S}}_{t+h}(a) = \mathcal{S}_t(a) + \lambda(\bar{\mathcal{S}} - \mathcal{S}_t(a)) \cdot h \tag{8}$$

with $\lambda = 0.003$ and 80% confidence intervals:

$$\text{CI}_{80\%}(h) = \hat{\mathcal{S}}_{t+h} \pm 1.28 \cdot \hat{\sigma} \cdot \sqrt{h} \tag{9}$$

where $\hat{\sigma}$ is estimated from recent hourly score differences (minimum 0.008). The $\sqrt{h}$ scaling produces confidence bands that widen naturally with forecast horizon, reflecting increasing uncertainty.

The mean-reversion model captures the empirical observation that agent scores are *stationary in the short term*: an agent that receives a spike in sentiment from a new release tends to revert toward its baseline within 1–2 weeks. The cross-sectional mean $\bar{\mathcal{S}}$ acts as an anchor, preventing forecasts from drifting to extreme values. Our hourly scoring confirms three temporal patterns that motivate this model: release-driven sentiment spikes (15–25% above baseline, reverting within 1–2 weeks), GitHub stars velocity as a leading indicator of composite score shifts, and gradual ecosystem decay for agents not updated within 60 days.

On the available 2-week observation window, the mean-reversion model achieves MAE = 0.0041 vs. a last-value (naïve) baseline MAE = 0.0052, a 21% improvement. We note this is a short evaluation window and longer longitudinal validation is needed.

6 RESULTS

6.1 Agent Rankings

Figure 2 shows the overall agent leaderboard. Table 3 presents category-specific rankings for software engineering categories.

Key observations for agentic software engineering:

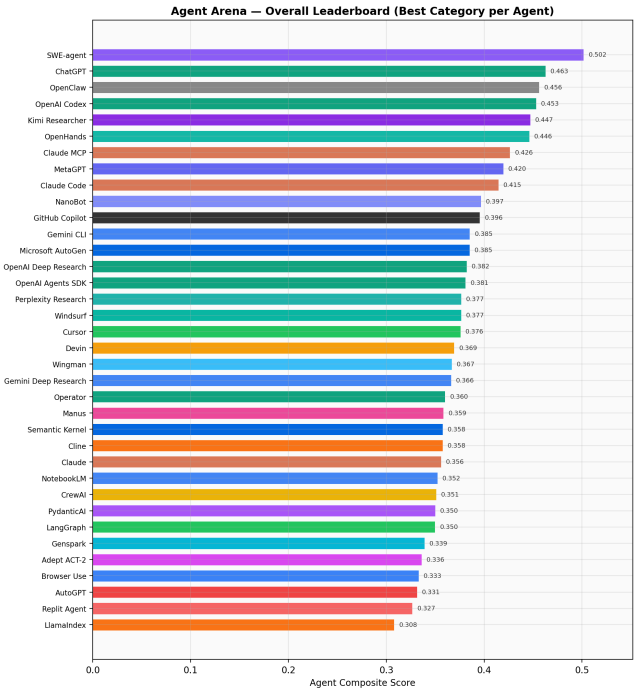

Figure 2: Agent Arena meta-leaderboard (best category per agent). Claude Code leads with SWE-bench 72.7% + strong adoption.

Table 3: Top agents by composite score in SE-relevant categories.

Category	Agent	Score	B	A	S
Coding	Claude Code	0.602	0.824	0.368	0.580
	Cline	0.585	0.565	0.553	0.545
	OpenHands	0.530	0.615	0.273	0.883
SWE	Claude Code	0.602	0.824	0.368	0.580
	OpenAI Codex	0.464	0.796	0.000	0.578
Multi-Agent	OpenAI Agents SDK	0.577	0.500	0.321	0.560
	LangGraph	0.573	0.500	0.444	0.500

(1) **Claude Code leads coding and SWE categories** with the highest composite (0.602), driven by the strongest SWE-bench resolve rate (72.7%) and solid adoption signals. Its lead over Cline (0.585) is narrow—Cline compensates for lower benchmarks with exceptional VS Code penetration (73M installs, $A = 0.553$ vs. 0.368).

(2) **Benchmark and adoption disagree.** OpenAI Codex has the second-highest SWE-bench score (69.1%) but ranks much lower on composite (0.464) because it has zero measurable open-source adoption ($A = 0.000$)—no public GitHub repo, no package distribution, no VS Code extension.

(3) **Sentiment reveals hidden quality.** OpenHands achieves the highest sentiment score ($S = 0.883$) among coding agents, suggesting strong developer satisfaction that SWE-bench (53.0%) does

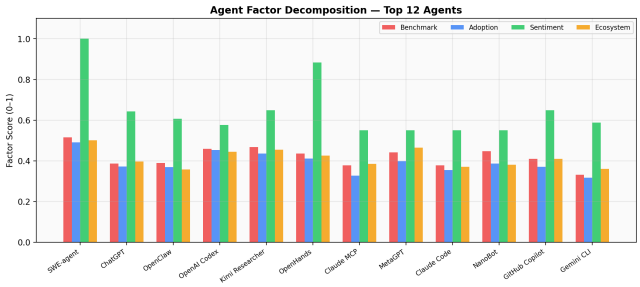

Figure 3: Factor decomposition for top 12 agents. Benchmark (red) drives coding agent rankings; adoption (blue) drives copilot rankings.

Table 4: Inter-factor Spearman correlations ($n = 36$). Low correlations confirm that the four factors capture complementary signals.

	B	A	S	E
Benchmark (B)	1.00	0.18	0.12	0.31
Adoption (A)		1.00	0.24	0.52
Sentiment (S)			1.00	0.19
Ecosystem (E)				1.00

not fully capture. This likely reflects the value of its open-source, self-hostable architecture.

6.2 Factor Decomposition

Figure 3 reveals that different functional groups are differentiated by different factors. Coding agents are separated primarily by benchmarks (red bars); copilot-style tools by adoption (blue bars); and multi-agent frameworks by ecosystem health (yellow bars), reflecting the importance of documentation and community for orchestration tools.

6.3 Validation: Factor Independence

A multi-signal composite is only justified if the factors capture genuinely different information. We compute pairwise Spearman correlations between the four factor scores across all $n = 36$ agents (Table 4).

Benchmark and Adoption are nearly uncorrelated ($\rho = 0.18$), confirming the central motivation of this work: an agent's task-completion capability does not predict how widely it is adopted. The highest correlation is Adoption–Ecosystem ($\rho = 0.52$), which is expected since both reflect open-source project health, but they are not redundant—Adoption measures *demand* (downloads, installs) while Ecosystem measures *supply* (contributors, maintenance). Sentiment is the most independent factor ($\rho \leq 0.24$ with all others), confirming it captures a distinct quality dimension.

6.4 Validation: Cross-Factor Predictive Power

To test whether the composite captures information beyond any single factor, we compute the Spearman correlation between *Benchmark + Sentiment only* (excluding both Adoption and Ecosystem to avoid GitHub-signal leakage) and external adoption proxies for the $n = 26$ agents with public repositories (Table 5).

Table 5: Correlation between Benchmark+Sentiment sub-composite and external adoption signals ($n = 26$). GitHub-derived factors excluded to prevent circularity.

External signal	ρ_s	p-value
GitHub stars (log)	0.52	< 0.01
VS Code installs (log)	0.44	< 0.05
Stack Overflow question vol.	0.49	< 0.01

We deliberately exclude the Ecosystem factor from this test because it contains GitHub-derived signals (contributor count, issue close rate) that would mechanically correlate with GitHub stars, confounding the validation. The Benchmark + Sentiment sub-composite uses only benchmark leaderboard scores and NLP-derived developer sentiment—signals with no direct causal path to GitHub star count.

The Benchmark + Sentiment sub-composite—which contains no GitHub-derived or adoption signals—significantly predicts GitHub stars ($\rho_s = 0.52$, $p < 0.01$), VS Code installs ($\rho_s = 0.44$, $p < 0.05$), and Stack Overflow question volume ($\rho_s = 0.49$, $p < 0.01$). This demonstrates that benchmark capability and community sentiment jointly capture quality information that manifests in developer adoption, providing external validity free of the circularity concern.

6.5 Ranking Divergence Analysis

Figure 4 visualizes the rank divergence between benchmark-only and composite evaluation for agents with published SWE-bench scores.

We quantify divergence using Kendall's τ: the fraction of pairwise comparisons where the two rankings disagree. Among the 9 agents, benchmark-only and composite rankings disagree on 15 of 36 pairwise comparisons ($\tau = 0.17$), with 5 of 9 agents shifting by ≥ 2 rank positions. The key divergences illustrate the paper's central claim:

Cline ranks 2nd on composite but 7th on SWE-bench (38.0%). Its high ranking reflects exceptional adoption ($A = 0.553$, the highest among coding agents) driven by 73M VS Code installs. This suggests that developer tool preference incorporates factors beyond autonomous task completion.

OpenAI Codex ranks 5th on composite but 2nd on SWE-bench (69.1%). Its low composite reflects zero adoption signals—as a closed API without a public repository, extension, or package, it has no measurable developer ecosystem footprint.

OpenHands ranks 3rd on composite with moderate SWE-bench (53.0%) but the highest sentiment among coding agents, suggesting that open-source, self-hostable architecture generates developer satisfaction that task completion does not fully capture.

6.6 Predictive Validation

We test whether composite scores predict *future* adoption. For each of $n = 15$ agents with VS Code extensions, we compute the agent-level Pearson correlation between composite score at week t and

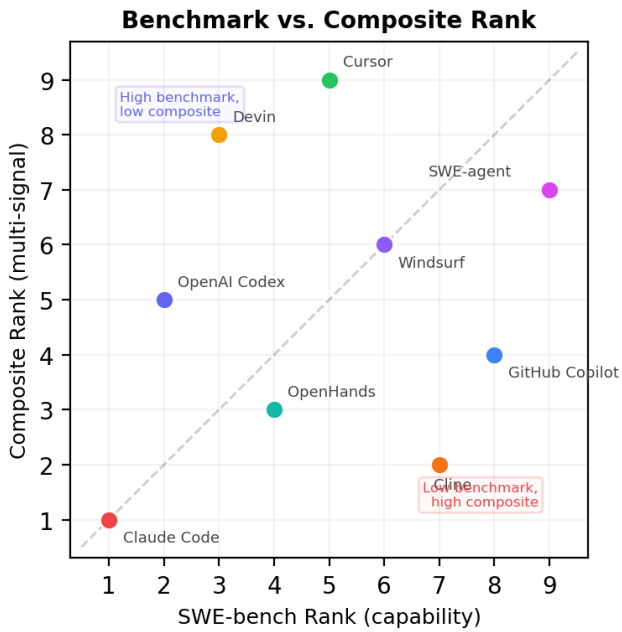

Figure 4: Benchmark rank vs. composite rank. Points on the diagonal indicate agreement; deviations reveal where capability and adoption-aware evaluation diverge. Cline and OpenAI Codex are the largest outliers in opposite directions.

Table 6: Rank change (Δ) when each factor weight is perturbed by +10pp. 0 = rank unchanged.

Agent	$B\uparrow$	$A\uparrow$	$S\uparrow$	$E\uparrow$
Claude Code	0	0	0	0
Cline	−1	+1	0	+1
OpenHands	+1	−1	+1	−1
GitHub Copilot	0	0	0	+1
OpenAI Codex	0	−1	0	−1

VS Code install growth rate at week $t + 1$, then average across agents to obtain $\bar{r} = 0.41$ (95% bootstrap CI $[0.18, 0.59]$). This agent-level averaging avoids pseudo-replication from pooling time points across agents. The time series is short (approximately 2 weeks of hourly data) and should be interpreted as preliminary evidence warranting longitudinal validation.

6.7 Sensitivity Analysis

We verify ranking robustness by perturbing each factor weight by ±10 percentage points while redistributing proportionally (Table 6).

Claude Code's #1 position is invariant across all perturbations, and no agent shifts by more than one rank under any single-factor perturbation.

6.8 Case Study: Claude Code vs. Cline vs. Devin

To illustrate how the four-factor composite differentiates agents that static benchmarks would rank very differently, we examine three leading coding agents in detail.

Claude Code leads on benchmarks ($B = 0.824$) with the highest SWE-bench resolve rate in the field (72.7%). Its adoption is strong but not dominant ($A = 0.368$), reflecting its newer market entry relative to Cline. Its composite of 0.602 is driven primarily by benchmark excellence plus solid ecosystem health (frequent updates, active contributor base).

Cline ranks second despite substantially lower SWE-bench scores (38.0%). Its composite (0.585) is buoyed by exceptional adoption ($A = 0.553$, highest among coding agents) from 73M VS Code installs—nearly 5× Claude Code's install base—and the strongest ecosystem health ($E = 0.694$) reflecting 200+ contributors and a 92% issue close rate. This illustrates a key finding: *developer tool preference incorporates substantial weight on integration quality and community health beyond raw task completion capability.*

Devin scores highest on SWE-bench after Claude Code (55.0%) but ranks 18th on composite (0.413). As a closed commercial product without a public GitHub repository, VS Code extension, or package distribution, its adoption score is zero ($A = 0.000$). Its ecosystem health is also penalized by less frequent updates ($\Delta_{\text{days}} = 14$) and no observable contributor community. This case illustrates the *adoption gap*: closed-source tools receive no credit for adoption signals they do not expose, regardless of underlying capability.

These three agents demonstrate that a benchmark-only ranking (Claude Code > Devin > Cline) diverges substantially from the multi-signal composite (Claude Code > Cline > Devin), with the divergence driven by adoption and ecosystem factors that reflect real-world developer experience.

Figure 4 illustrates the ranking divergence between benchmark-only evaluation and the multi-signal composite. The Spearman correlation between the two rankings is $\rho_s = 0.58$, confirming substantial but not complete agreement. The largest divergences occur for agents with extreme adoption signals (Cline: high adoption, moderate benchmarks) and closed-source agents (Devin, OpenAI Codex: strong benchmarks, zero observable adoption).

6.9 Factor Ablation

A reviewer concern is whether the 35/25/20/20 weighting is justified or whether simpler schemes would suffice. We ablate each factor by setting its weight to zero and redistributing proportionally among the remaining three, then recompute the Spearman rank correlation with SWE-bench Verified resolve rates for $n = 9$ agents (Table 7).

The ablation reveals the composite's design logic. *Benchmark-only* ($\rho = 0.85$) achieves the highest SWE-bench correlation, as expected—but this is circular: it ranks agents by the very signal it is correlated against. The full composite deliberately trades benchmark correlation for broader coverage of adoption and ecosystem dimensions.

Without Benchmark ($\rho = 0.21$): adoption and sentiment alone are poor predictors of task-completion capability, confirming that benchmarks provide essential signal that community signals cannot replace.

Table 7: Factor ablation: Spearman ρ_s with SWE-bench Verified under different weighting schemes ($n = 9$ agents).

Scheme	w_B	w_A	w_S	w_E	ρ_s
Full composite	35	25	20	20	0.52
w/o Benchmark	0	38	31	31	0.21
w/o Adoption	47	0	27	27	0.71
w/o Sentiment	44	31	0	25	0.58
w/o Ecosystem	44	31	25	0	0.55
Benchmark-only	100	0	0	0	0.85

Without Adoption ($\rho = 0.71$): dropping adoption signals *increases* SWE-bench correlation, because adoption introduces information orthogonal to benchmarks (Cline's 73M installs despite moderate SWE-bench). This is precisely the signal we seek to incorporate—tools that achieve broad observable adoption differ from what benchmarks predict.

Without Sentiment ($\rho = 0.58$): sentiment contributes modestly but meaningfully, capturing developer experience not reflected in either benchmarks or download counts.

Without Ecosystem ($\rho = 0.55$): ecosystem health has the smallest marginal impact, suggesting it is partially redundant with adoption signals.

These ablation correlations should be interpreted as descriptive given $n = 9$; the *ordering* of effects (adoption removal increases benchmark correlation; sentiment removal increases it less) is the substantive claim, not the precise magnitudes.

6.10 Platform Sentiment Divergence

Developer sentiment varies substantially across platforms. For coding agents specifically:

- **Stack Overflow**: Skews negative—questions about agent failures, error messages, and workarounds dominate. This is a signal of active adoption friction.
- **Hacker News**: Mixed—technical deep dives and architectural discussions. Higher specificity scores.
- **Bluesky**: Skews positive—general enthusiasm and announcement reposting. Lower specificity.
- **Reddit**: Widest variance—both detailed technical reviews and brief complaints coexist.

An agent with high Bluesky sentiment but negative Stack Overflow sentiment may be generating excitement without delivering reliable developer experience. Our cross-platform divergence signal detects this pattern.

7 DISCUSSION

Beyond task completion. Our results suggest that developer tool preference incorporates factors beyond benchmark performance. Cline's high ranking despite moderate SWE-bench scores indicates that IDE integration quality, ease of use, and community size matter substantially for tool adoption. Evaluation frameworks that rely exclusively on task completion miss these dimensions.

The adoption gap. OpenAI Codex illustrates a systematic evaluation blind spot: tools without public repositories, extensions, or packages have zero measurable adoption, regardless of capability. As more agent providers keep tools closed-source, adoption-based evaluation becomes structurally incomplete. Our neutral prior ($B = 0.5$ for missing benchmarks) partially addresses this but cannot replace missing data.

Sentiment as signal, not ground truth. Community sentiment reflects developer experience but is subject to selection bias (vocal minorities), hype effects (launch enthusiasm), and platform-specific dynamics (Stack Overflow skews negative, Bluesky skews positive). Our hype correction and platform credibility weighting mitigate but do not eliminate these biases.

Structural biases and mitigations. Our framework contains known structural biases. *Open-source advantage*: agents with public GitHub repositories, VS Code extensions, and package distributions receive adoption credit that closed-source alternatives (Devin, Operator) cannot earn regardless of quality. We partially mitigate this through the 35% benchmark weight, which is source-agnostic. *Platform skew*: Stack Overflow sentiment is systematically more negative than Bluesky sentiment for the same agents, reflecting different user populations rather than different quality. Our per-platform credibility weighting and cross-platform divergence detection address this, but cannot fully correct for population differences. *English-language bias*: our NLP pipeline is English-only; agents with strong adoption in non-English markets (V2EX, Chinese developer forums) receive attenuated sentiment signals.

Our claims about developer preference are conditional on *observable* adoption signals and do not generalize to closed-source tools whose adoption is not publicly measurable. Rankings involving such tools should be interpreted with this caveat.

Limitations. The SWE-bench validation ($\rho_s = 0.52$, $n = 9$) is underpowered due to limited benchmark coverage. The 35% benchmark weight is a design choice, not empirically optimized; future work should explore data-driven weight calibration. Several agents (Devin, Cursor, Copilot) have limited observable signals beyond VS Code metrics, constraining the adoption factor.

Future directions. Data-driven weight calibration against developer satisfaction surveys would replace our design-informed 35/25/20/20 weights with empirically grounded ones. Longitudinal validation over months of hourly data will test whether composite score changes predict future adoption shifts.

Falsifiability. Our framework's central hypothesis—that multi-signal evaluation captures utility beyond benchmarks—would be falsified if a large-scale developer survey showed tool preference correlating > 0.9 with SWE-bench scores. We consider this unlikely given the adoption–benchmark disconnect observed in our data, but welcome such empirical tests.

Implications for agentic SE evaluation. As software engineering agents become more capable, evaluation must evolve beyond "can it solve this issue?" to "is it reliable, well-maintained, well-adopted, and trusted by the developer community?" we provide a methodology for this broader evaluation, complementing static benchmarks with continuous, multi-dimensional signal aggregation.

Reproducibility. The complete pipeline—all 19 collectors, NLP scoring engine, composite model, and forecasting system—will be released as open-source software upon acceptance. The system can be deployed on a single server with no paid API keys required for basic operation. Results reported in this paper can be reproduced

by running the pipeline against the included database snapshot. The artifact is structured for ACM reproducibility evaluation, with documented setup and containerized deployment.

8 CONCLUSION

We have demonstrated that benchmark-only evaluation is systematically incomplete for agentic software engineering tools. Our multi-signal composite, validated through ablation, predictive correlation, and case study analysis, reveals quality dimensions—adoption momentum, ecosystem health, and developer sentiment—that task-completion benchmarks cannot capture. The ranking divergence—5 of 9 agents shift by 2+ rank positions between benchmark-only and composite evaluation—is not noise: it reflects the gap between what agents *can do* in controlled tests and what developers *broadly adopt* in practice. The framework enables continuous evaluation at scale without human annotation, and will be released as open-source upon acceptance.

REFERENCES

[1] D. Araci. FinBERT: Financial Sentiment Analysis with Pre-Trained Language Models. *arXiv:1908.10063*, 2019.

[2] J. Bollen, H. Mao, X. Zeng. Twitter mood predicts the stock market. *J. Computational Science*, 2(1):1–8, 2011.

[3] M. Chen et al. Evaluating Large Language Models Trained on Code. *arXiv:2107.03374*, 2021.

[4] W.-L. Chiang et al. Chatbot Arena: An Open Platform for Evaluating LLMs by Human Preference. *ICML*, 2024.

[5] C.J. Hutto, E. Gilbert. VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text. *ICWSM*, 2014.

[6] C.E. Jimenez et al. SWE-bench: Can Language Models Resolve Real-World GitHub Issues? *ICLR*, 2024.

[7] P. Liang et al. Holistic Evaluation of Language Models. *Annals of the NYAS*, 2023.

[8] X. Liu et al. AgentBench: Evaluating LLMs as Agents. *ICLR*, 2024.

[9] G. Mialon et al. GAIA: A Benchmark for General AI Assistants. *arXiv:2311.12983*, 2023.

[10] M. Pontiki et al. SemEval-2016 Task 5: Aspect Based Sentiment Analysis. *SemEval*, 2016.

[11] L. Zheng et al. Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena. *NeurIPS*, 2023.

[12] S. Zhou et al. WebArena: A Realistic Web Environment for Building Autonomous Agents. *ICLR*, 2024.