SURVEY OF AI CODING RESEARCH PROJECTS 2020-2025 (by Jad Nohra)

⚠️ **CRITICAL: These studies become outdated rapidly.** AI models improve weekly. Studies from any date may not reflect current model performance. However, security vulnerabilities and code quality issues have remained consistent 2020-2025, suggesting some problems are systemic to the approach rather than model capability.

GITCLEAR 2020-2024: 211M LOC, 4 YEARS

Largest longitudinal study: 153M → 211M lines tracked industry-wide, not single company

Churn rate (reverted within 2 weeks)	2.1x increase
2021 baseline (pre-AI adoption)	3.3%
2024 measured	7.1%
Copy-paste (duplicate code %)	+48% in 3yr
Refactoring (code moved/reorganized)	-60% activity
5+ line duplicates (identical blocks)	8x increase
Commits w/ clones (contain duplicates)	10% by 2024

What this measures: Long-term code quality metrics across industry. Churn = lines reverted or updated within 2 weeks of authoring. Technical debt compounds ~1.48x/year (48%÷3).

Key finding: First time copy-pasted lines exceeded moved lines in their measurement history. Authors note "disconcerting trends for maintainability."

MICROSOFT/ACCENTURE/FORTUNE 100 2025: 4,867 DEVS, 2-8 MONTHS

Largest RCT: 3 companies (Microsoft 1,521, Accenture 316, Fortune 100 3,030), GitHub telemetry

PRs completed (feature units)	+26.08%
Statistical significance	$p<0.05$
Builds (compilation runs)	+38.38%
Merge rate (PR acceptance %)	+15%
Juniors (<2yr experience)	+40% PRs
Seniors (5+ yr experience)	+7% PRs
Adoption rate (jr vs sr usage)	81.6% vs 72.1%
Learning curve (weeks to plateau)	11 weeks
Build success (% passing CI tests)	-5.53% NS

What this measures: Production work with optional AI usage. Developers self-selected when to use tools. Only 31% of sample had experience-level data.

Authors claim: "Turning 8-hour day into 10 hours of output" based on PR count increase.

UPLEVEL 2024: 800 DEVELOPERS, 26 DAYS

Short-term production study with Copilot adoption, real workplace metrics

Bug rate (defects per PR)	+41%
PR cycle time (submission to merge)	No change
Throughput (PRs per developer)	No change
Net change	More bugs, same speed

What this measures: Initial adoption period in production environment. 26-day window captures early usage patterns.

Contradicts: GitHub's controlled study showing quality improvements.

FASTLY 2025: 791 DEVELOPERS SURVEYED

Cross-sectional survey, July 2025, self-reported usage patterns

Seniors (>50% code AI-generated)	32%
Juniors (>50% code AI-generated)	13%
Usage ratio	2.5x sr:jr
Seniors (time lost to fixes)	30% report
Juniors (time lost to fixes)	17% report

What this measures: Self-reported AI usage patterns by experience level. Perception-based, not measured.

Pattern: More experienced developers use AI more but also report more time fixing outputs.

CORNELL 2024: 452 GITHUB SNIPPETS ANALYZED

Production code marked Copilot-generated, passed human review, in repositories

With vulnerabilities (% of snippets)	29.6%
Total vulnerabilities	544
Distinct CWE types	38
Python (% vulnerable)	32.8%
JavaScript (% vulnerable)	24.5%
In CWE Top-25 (% of vulns)	40.1%

What this measures: Vulnerabilities in production code after passing human review. Real-world deployment rate.

Most common: Insufficient random values (23.3%), improper code generation (21.1%), OS command injection (14.9%).

MIT 2025: 150 LEADERS, 300 DEPLOYMENTS

Enterprise adoption study, interview-based, business metrics focus

Achieving rapid ROI (revenue/profit gain)	5%
No measurable value (failed pilots)	95%
Abandoning initiatives (current rate)	42%
Previous year (2024 rate)	17%
Year-over-year increase (abandonment)	2.5x

What this measures: Business outcomes vs technical metrics. ROI = return on investment measured by revenue or profit impact, not velocity.

Finding: Technical improvements (more PRs, faster coding) don't translate to business value (revenue, cost savings).

VERACODE 2025: 100+ LLMS, 80 TASKS

Comprehensive security testing across all major models and sizes

Pass rate (% generating secure code)	55%
Log injection (CWE-117 vulnerability)	88% fail
XSS (cross-site scripting, CWE-79)	86% fail
Java (% passing security tests)	28.5%
Python (% passing security tests)	61.7%
Model size impact (on security)	None
Improvement 2023-2025	0%

What this measures: Systematic failures across all models. Size doesn't help (training data issue). Stack Overflow lacks security context = models can't learn it.

Pattern: Syntax improved >90% since 2023, security stuck at 55%.

GITHUB 2022: 95 DEVELOPERS, SINGLE TASK

Most-cited study: Controlled experiment, HTTP server, clean environment

With Copilot (minutes)	71
Without (minutes)	161
Speed gain (best case)	+55.8%
Confidence interval	21-89%
Task complexity	Simple
Statistical significance	p=0.0017

What this measures: AI's ceiling performance on ideal task. No legacy code, no dependencies, well-defined problem. This 55.8% is what vendors cite, least representative of real work.

MCKINSEY 2023: 40+ DEVELOPERS, CONTROLLED

Task-specific testing, within-subjects design, US and Asia locations

Juniors <1yr (on complex tasks)	Slower
Documentation (writing specs/comments)	+45-50%
Refactoring (code reorganization)	+66%
Code generation (new features)	+35-45%
Complex unfamiliar (new frameworks)	<10%
Developer happiness (self-reported)	2x higher

What this measures: Task-specific performance variations. Pattern-matching tasks vs understanding-required tasks.

Key finding: Juniors need "foundational programming principles" training before AI helps.

METR 2025: 16 EXPERTS, 246 TASKS

RCT design, real repos (22k+ stars, 1M+ LOC), experienced developers (avg 5yr on project, 1500+ commits)

Predicted (pre-task estimate)	+24%
Believed (post-task perception)	+20%
Actual (screen-recorded time)	-19%
Perception-reality gap	39 points
Accept rate (% suggestions used)	44%
Modified (% needing changes)	56%
Review behavior (% of developers)	75% read all
Cleanup time (% of total time)	~9%

What this measures: Expert developers using AI in codebases they know deeply. Screen recordings captured actual time spent on tasks.

Context: Only 1/16 had >1 week Cursor experience. Economics experts predicted +39%, ML experts +38% gain.

ACM ICER 2024: 21 STUDENTS, EYE-TRACKING

Detailed observation study with eye-tracking, interviews, quality analysis

Completed tasks	20/21
Quality variance (between students)	Extreme
Strong students (impact)	Accelerated
Weak students (impact)	Hindered
Performance gap	Widened

What this measures: Learning impacts on novice programmers. Eye-tracking revealed attention patterns.

Key quote: AI "compounds metacognitive difficulties" for struggling students, creates "illusion of competence."

STANFORD 2022: SECURITY EXPERIMENT

Controlled experiment measuring security quality and developer confidence

Code security (measured)	Decreased
Developer confidence (self-rated)	Increased
Direction mismatch	Opposite

What this measures: Confidence-competence relationship when using AI. Objective security vs subjective confidence.

DEVELOPER FORUMS 2024-2025

Pseudonymous testimony (Reddit r/experienceddevs, HackerNews):

- "AI produces 80% in minutes, takes ages to fix duplicate code, bad design, bugs" (436 upvotes)
- "Best suggestions = what linters catch. Is this what we've come to?" (HN top comment)
- "Juniors ship faster but can't debug. Never debugged code they don't understand"
- "Without Copilot: deer in syntax-shaped headlight" (dependency formed)
- "The 70/30 problem: 70% quick, 30% takes longer than 100% from scratch"
- "People overestimate because it's fun to use" (dopamine vs productivity)

What this measures: Gap between public surveys (80-95% satisfaction) and pseudonymous forum discussions.

THE PATTERN ACROSS 14 STUDIES

- **Context matters more than tool:** GitHub's +55.8% (simple, isolated) vs METR's -19% (experts in familiar code) shows task-fit determines outcome
- **Perception gap is consistent:** METR devs believed +20% faster while measuring -19% slower. Similar gaps appear across studies
- **Experience creates paradox:** Juniors gain +21-40% on routine work but can't debug. Seniors use 2.5x more AI but spend 30% of gains fixing it
- **Quality metrics diverge from speed:** +26% more PRs (Microsoft) while code churn doubles (GitClear) and bugs increase +41% (Uplevel)
- **Security shows no learning:** 45% vulnerability rate unchanged 2020-2025 despite model improvements. Pattern suggests training data problem
- **The 70/30 problem:** Developers report "70% done quickly, last 30% takes longer than writing 100% manually" (multiple forums)
- **Skill atrophy reported widely:** "Can't code without Copilot" appears frequently. 89% of students encounter wrong information (ACM study)
- **Business metrics disconnect:** Only 5% achieve ROI (MIT) despite technical metric improvements. 42% abandon initiatives within year