



Hochschule Aalen
UNIVERSITY OF APPLIED SCIENCES

Lecture 8

MEASURING THE MACHINE

The €50 Bet

WOULD YOU BET €50?

A frontier model's launch post reads "**~95% on SWE-bench Verified.**" The implied promise: hand the system a closed Locust issue — just the issue text and the repo — and nineteen times out of twenty it produces a correct fix. **Would you bet €50 it transfers to your Locust work?**

Write your answer down now — before the lecture tells you anything.

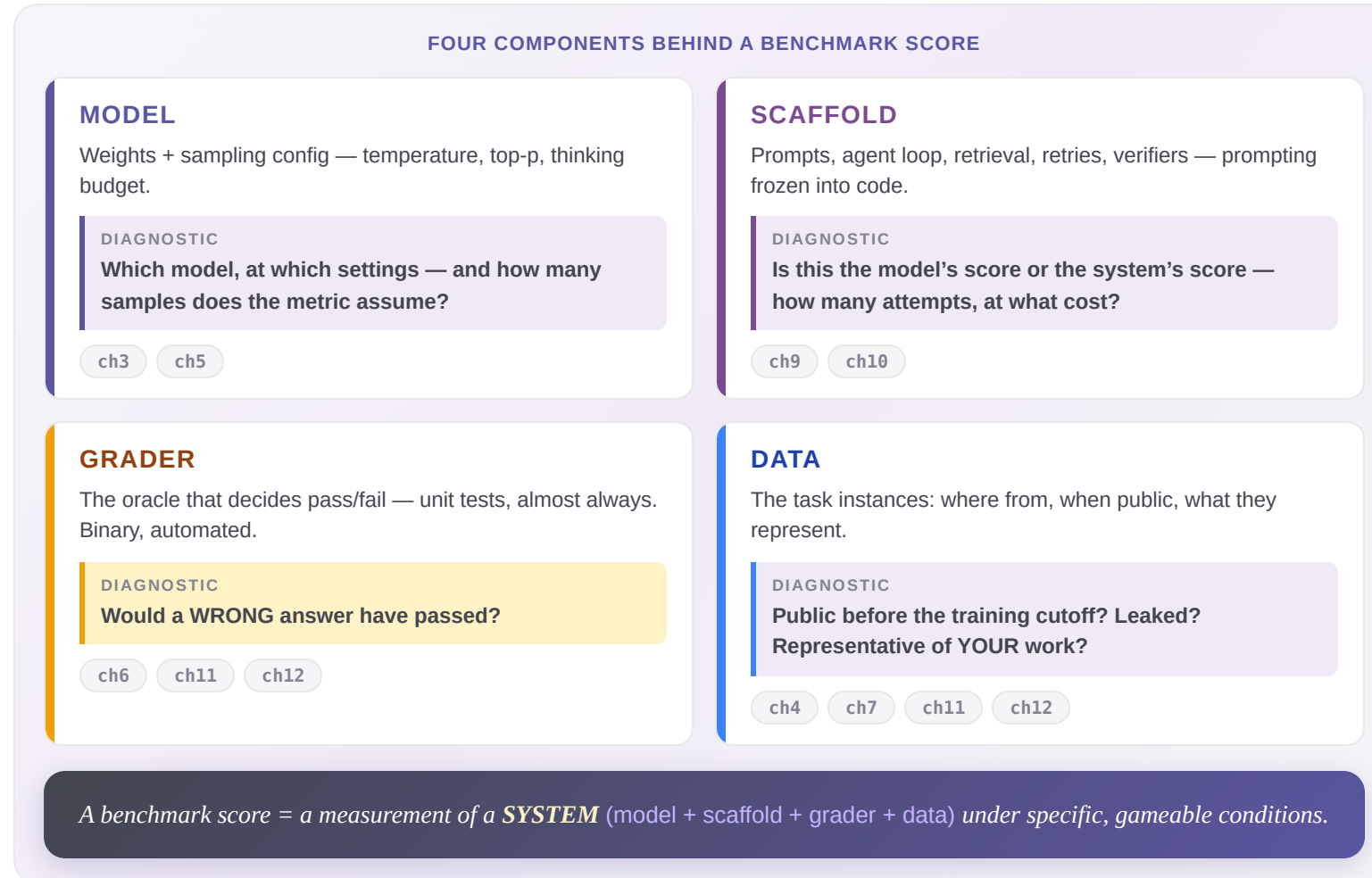
PRESS YOURSELF

Ninety-five percent of *what*, exactly?

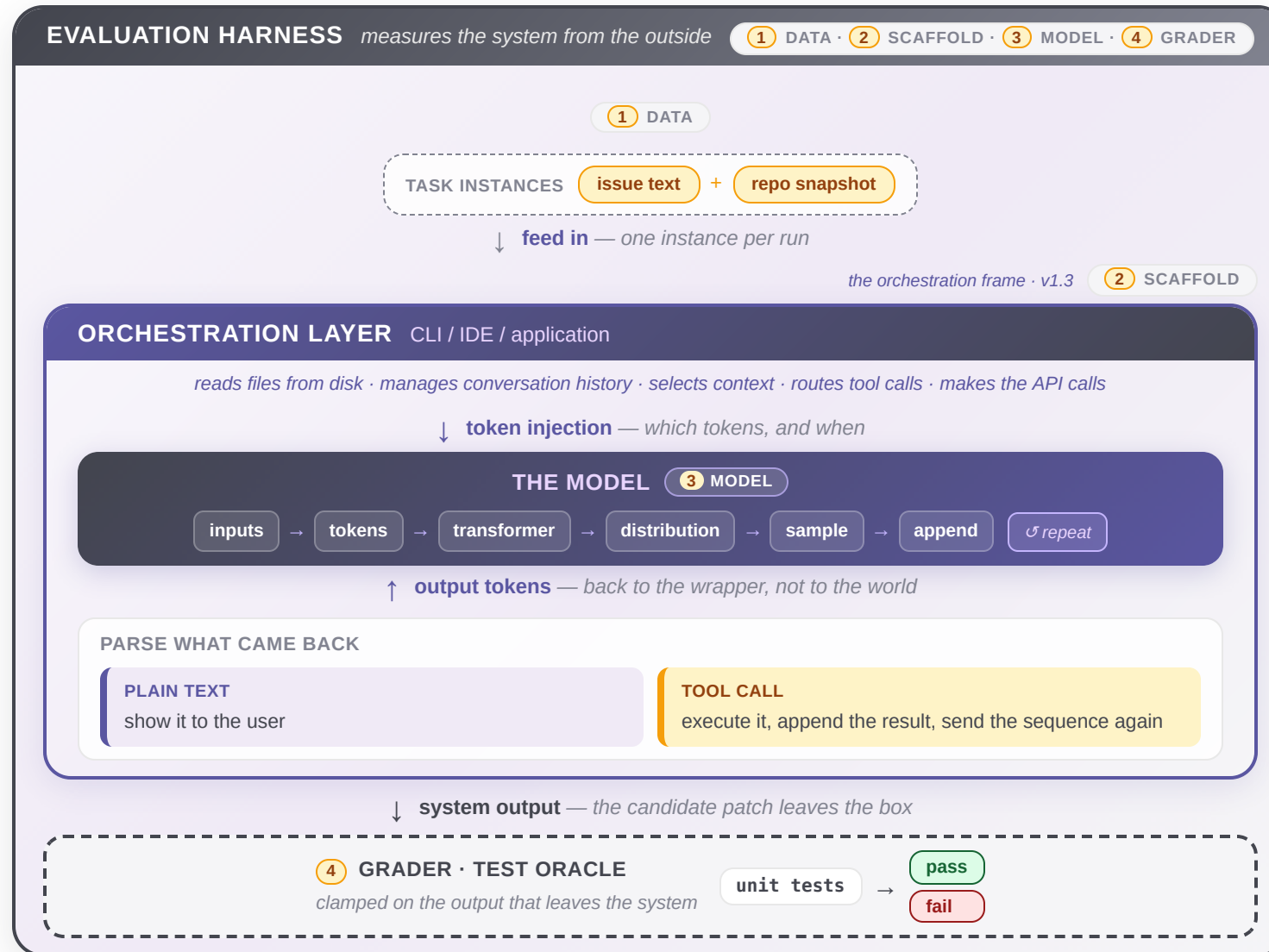
Which tasks, chosen how? · Model alone, or wrapped in retries and best-of-N? · Graded by what — would a wrong answer pass? · Public on GitHub before the training cutoff?

Each question has a precise, researched answer for SWE-bench — and those answers are this lecture.

Four Components of a Benchmark Score



LLM-in-One-Picture v1.4



A benchmark score is a measurement of a **system** — model + scaffold + grader + data — under specific, **gameable** conditions.

the sentence to carry to the exam · neither worship nor nihilism — a calibration manual

Block A / Block B

Block A — Building the measurement toolkit

Chapters 1–9, ~90 min. Three problems → pass@k → EvalPlus → SWE-bench → the leaderboard curve → a cliffhanger.

HYPE → FIRST CRACK

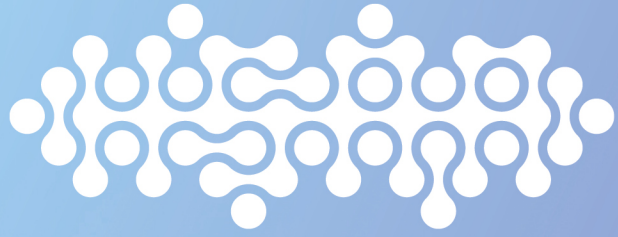
Block B — Why the number lies

Chapters 10–14, ~90 min. The scaffold problem → the manual audit → contamination → the nine questions → the lab.

REALISM → THE ARTIFACT

THE SHAPE OF THE DAY

First half builds the instruments; second half uses them to audit. **The break falls on a cliffhanger** — two numbers, eighteen points apart, for the same models.



Hochschule Aalen
UNIVERSITY OF APPLIED SCIENCES

The Claim Board

1

Five Claims, Zero Ratings

#	Claim, as pinned	Where it comes from	The first question it raises
1	"~95% on SWE-bench Verified"	Frontier-lab launch, Jun 2026 (two systems at ~95% / 95.5%, vendor-measured)	Which scaffold, how many attempts — who checked the grading?
2	"The best coding model in the world"	The same genre of launch post	Best by which measurement — what would falsify it?
3	"#1 on LMArena"	LMArena Text, entered Jun 10 2026 at 1510	What do blind preference votes measure?
4	"A 50% time horizon of ~16 hours"	METR time-horizon, May 2026	Sixteen hours of <i>what</i> , completed how often?
5	"80.3% on SWE-bench Pro"	Vendor-reported, Jun 2026 (public best ~46–59%)	Why 20+ points above the best public entry?

THE RULE OF THE BOARD

A claim only comes down **rated** — and we don't have the tools yet. None of these is an obvious lie; none is self-evidently true. That is exactly why they are on the board.

Three Verdicts

✓ VERIFIED

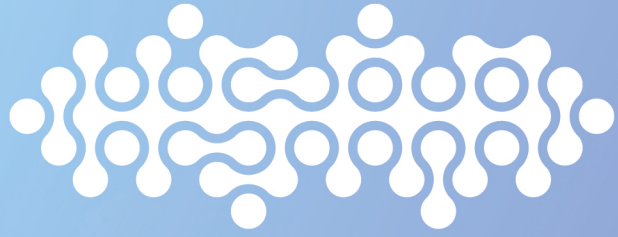
The claim survives the nine questions: measurement conditions disclosed, independently reproduced or checkable in principle, and it means what the headline implies.

? PLAUSIBLE-BUT-UNPROVEN

Nothing refutes it, but the disclosures needed to check it — scaffold, attempt count, sampling, error bars, subset — are missing. The claim may be true; it is currently unfalsifiable as published.

✗ REFUTED-AS-STATED

The number may even be real, but the claim *as phrased* means something materially different from what the headline implies.



Hochschule Aalen
UNIVERSITY OF APPLIED SCIENCES

Why Evaluating LLMs Is Hard

2

Three Fundamental Problems

THREE PROBLEMS A SINGLE BENCHMARK NUMBER HIDES

SAMPLING

The output is a **SAMPLE**, not a property.

One run is one Bernoulli draw from a per-task success probability p . Same model, same prompt: pass, then fail — nothing changed but the dice.

→ answered by a **metric**:
pass@k (ch5)

HARNESS

The program shapes the score — and fails **SILENTLY**.

Prompt format, answer extraction, execution environment all move the number; a bug just shifts a value, with no stack trace.

→ grader (ch6) scaffold (ch10)
audit (ch11)

CONSTRUCT

'Can code' is not one thing.

- function synthesis
- cross-file bug fixing
- repository navigation
- long-horizon autonomy

near 90% on HumanEval vs <2% on SWE-bench — same models, 2023. A 45x gap.

→ the reason **SWE-bench** exists
ch7

The three reasons a benchmark number can lie — and the lecture's table of contents in miniature.

The Anatomy of an Evaluation Run

ONE MACHINE BEHIND EVERY BENCHMARK

Strip any benchmark in this lecture to its plumbing and the same **six stages** appear, in order, turning a dataset into a headline number. The rest of the lecture is a tour of these boxes — **every remaining chapter zooms into one**.

AN EVALUATION RUN — ONE PIPELINE, TWO SHAPES

schematic — strip any benchmark in this lecture to its plumbing and these six stages appear, in this order



Function level — HumanEval: one forward pass

Sample = a single greedy draw · Execute = run three asserts. §4.2 walks one through.

Repository level — SWE-bench: a loop with tools ♻️

Sample = an agent acts & observes for dozens of turns · Execute = a per-instance Docker test suite. §7.5 walks one through.

*What changes between the two shapes is not the six stages — only how much machinery hides inside **Sample** and **Execute** (the amber boxes).*

HumanEval and the Founding Template

THE FOUNDING TEMPLATE (2021, THE CODEX PAPER)

Function-level · static · test-verified. A signature and a docstring in, a function body out, unit tests decide. **HumanEval** = 164 hand-written problems; **MBPP** = 974 crowd-sourced ones. Between them they defined what "can code" meant, in numbers, for the first years of the LLM era.

```
# signature + docstring in, function body out, tests decide
def common_prefix_length(a: str, b: str) -> int:
    """Return the length of the longest common prefix of a and b."""
    # ← the model writes everything below this line
    assert common_prefix_length("flag", "flake") == 3    # the grader: just assertions
    assert common_prefix_length("same", "same") == 4
```

One HumanEval Run — Prompt to `samples.jsonl`

THE PROMPT IS A DATASET FIELD, NOT A CHAT

The instance's `prompt` field is the model's input, verbatim — signature + docstring, nothing else. The model continues the file and, left alone, rolls into the *next* function; the harness gives the decoder **stop sequences** (`\ndef`, `\nclass`) that cut it. That trimming is the Extract step — in *your* code, before any test runs.

THE GENERATION SIDE — FROM PROMPT TO STORED COMPLETION

1 PROMPT — a field in the dataset, fed to the model verbatim

```
def common_prefix_length(a: str, b: str) -> int:
    """Return the length of the longest common
    prefix of the strings a and b."""
```

↓
the base model continues the text, one token at a time

2 RAW MODEL OUTPUT — it keeps writing past the function

```
n = min(len(a), len(b))
for i in range(n):
    if a[i] != b[i]:
        return i
return n
```

→ first `\ndef` → truncate here; keep everything above

```
def levenshtein(a, b):
    """Edit distance between a and b."""
    ...
```

↓
your generation code truncates at the stop sequence — before any test runs

3 STORED — one line in `samples.jsonl`, completion only (no signature, no docstring)

```
{"task_id": "HumanEval/0",
"completion": "    n = min(len(a), len(b))\n    for i in range(n):\n        if a[i] != b[i]:\n            return i\n    return n\n\n\ndef levenshtein(a, b):\n    """Edit distance between a and b."""\n    ...
```

All three steps are the **generation** stage — your code. The model emits more than you asked for; **your** sampler trims it. No grader has run yet, and the trimming is exactly why it matters who does it.

How the Grader Assembles the Program

THE GRADER DOES NOT JUDGE THE CODE — IT RUNS IT

It **concatenates four pieces verbatim** — prompt + your stored completion + the hidden test (which defines `check`) + one line to call `check` — and executes the result. It never parses, reformats, or trims your bytes. Had the discarded `def levenshtein` still been in your completion, it would have been pasted in too — which is exactly why truncation was *your* job upstream.

THE GRADING SIDE — WHAT THE HARNESS ACTUALLY RUNS

prompt — from the dataset

completion — your stored output

hidden test — from the dataset

check() — appended by the harness

```
def common_prefix_length(a: str, b: str) -> int:
    """Return the length of the longest common
    prefix of the strings a and b."""
    n = min(len(a), len(b))
    for i in range(n):
        if a[i] != b[i]:
            return i
    return n
def check(candidate):
    assert candidate("flag", "flake") == 3
    assert candidate("", "anything") == 0
    assert candidate("same", "same") == 4
check(common_prefix_length)
```

run as-is in a sandbox — the grader does no trimming of your completion

ran to the end, no `AssertionError` → PASS

any failed assert · any other exception · any timeout → FAIL. Binary, no partial credit.

Saturation

NEAR 90% ON HUMANEVAL BY 2023

When every frontier model passes nearly every task, the benchmark stops ranking anything. On a 164-problem set, one problem is worth ~0.6 points — two models "a point apart" differ on roughly **two problems out of 164**, well inside the sampling noise.

THE MISCONCEPTION TO KILL

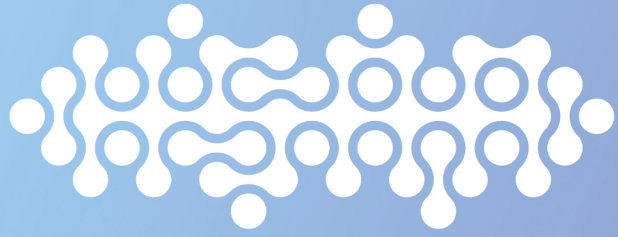
Saturation does **not** mean the models are equally good. It means the **test ran out of headroom** — the instrument hit its ceiling, not the population. A number quoted from a saturated benchmark is a fact about the benchmark.

Checkpoint: The Map So Far

HOLD THESE THREE

1. Three problems → three resolution chapters: sampling (ch5), harness/grader (ch6), construct (ch7). **2.** The founding template — function-level, static, test-verified — saturated near 90%. **3.** Saturation is about the yardstick, not the models.

Next: build the metric that fixes Problem 1 (sampling) — and do the statistics properly.



Hochschule Aalen
UNIVERSITY OF APPLIED SCIENCES

pass@k: Scoring a Stochastic Programmer

3

The Output Is a Sample

THE FOUNDATION

A benchmark run does not observe capability; it **samples** it. Run the same prompt twice at non-zero temperature and you get two different function bodies — sometimes both correct, sometimes neither.

For a fixed task there is a per-task success probability p . One run is one Bernoulli draw from it:

$$\Pr[\text{one run passes}] = p$$

$$\Pr[\text{one run fails}] = 1 - p$$

THE HONEST QUESTION

Not "can the model solve task t ?" (a boolean) but "**with what probability does one sampled attempt pass the tests?**"
pass@k is the field's formalization of the fix.

The Naive Estimator — and Its Bias

What we want is $1 - (1 - p)^k$, but p is unknown. The obvious move: generate n samples, count the c that pass, set $\hat{p} = c/n$, and plug in:

$$\widehat{\text{pass@}k}_{\text{naive}} = 1 - \left(1 - \frac{c}{n}\right)^k$$

FOR $k > 1$ THIS IS BIASED — NOT NOISY-BUT-FAIR, BUT SYSTEMATICALLY TOO LOW

Jensen's inequality: the all-fail term is a **convex** function of the noisy estimate, and a convex curve lies above its chords.

Honest variance gets rectified into a one-directional error. **Noise in, bias out** — and no amount of averaging over tasks cures it.

The Bias, as Three Lines of Algebra

Take the all-fail term $f(x) = (1 - x)^k$, convex for $k > 1$, with $\mathbb{E}[\hat{p}] = p$. Jensen — $\mathbb{E}[f(X)] \geq f(\mathbb{E}[X])$ — does the rest:

$$\begin{aligned}\mathbb{E}[(1 - \hat{p})^k] &\geq (1 - p)^k && \text{(Jensen)} \\ 1 - \mathbb{E}[(1 - \hat{p})^k] &\leq 1 - (1 - p)^k && \text{(subtract from one)} \\ \mathbb{E}[\widehat{\text{pass@}k}_{\text{naive}}] &\leq \text{pass@}k && \text{(naive sits below the truth)}\end{aligned}$$

THE BIAS HAS A SIZE, NOT JUST A SIGN

Only the first line is load-bearing; the rest is arithmetic. The gap Jensen leaves open is, to leading order, $\frac{1}{2} \cdot f''(p) \cdot \text{Var}(\hat{p})$ — variance bent upward by curvature. At $k = 2$ the curvature is constant, so it is **exact**: the plug-in is low by precisely $p(1-p)/n$.

The Unbiased Estimator

The fix ships in the Codex paper (Chen et al., 2021). Generate $n \geq k$ samples, count the c correct, and compute:

$$\text{pass@}k = \mathbb{E}_{\text{tasks}} \left[1 - \frac{\binom{n-c}{k}}{\binom{n}{k}} \right]$$

Two averages hide here, easy to conflate: $\mathbb{E}_{\text{tasks}} = \frac{1}{N} \sum_i$ runs **over tasks** — deterministic, given the data; the word *unbiased* names a different average, **over re-sampling one task**, where $c \sim \text{Binomial}(n, p)$.

READ THE FRACTION AS A COUNTING STATEMENT

C(n-c, k) counts the ways to pick k samples **entirely from the failures**; **C(n, k)** counts all ways to pick k of n . The fraction is "all k miss"; one minus it is "at least one of k passes." You ask the pass@ k question **directly of the samples you have** — no plug-in, no curve, no bias.

Worked Example: $n = 10$, $c = 3$, pass@2

Naive plug-in — same ten samples:

$$1 - \left(1 - \frac{3}{10}\right)^2 = 1 - 0.49 = 0.51$$

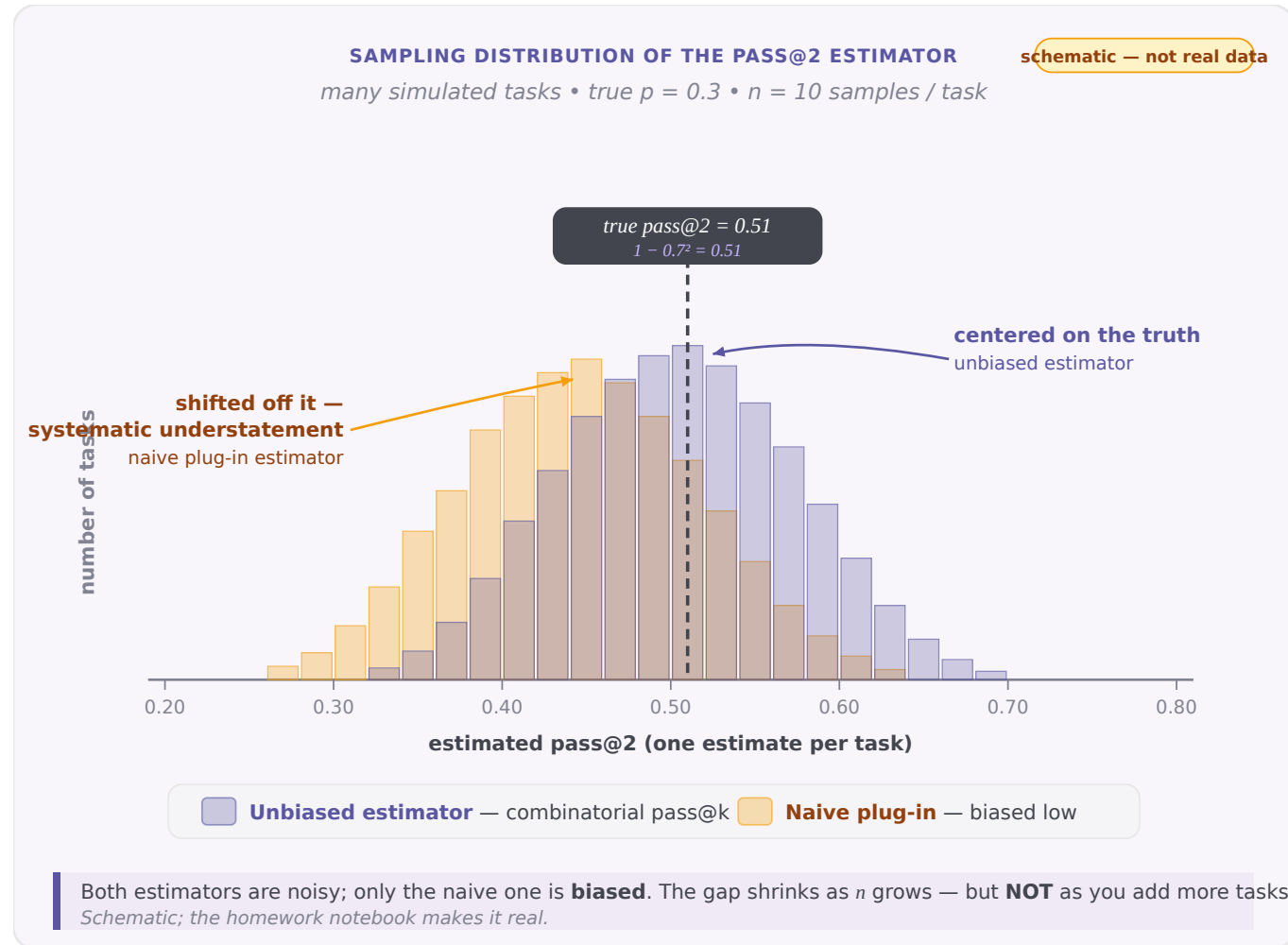
Unbiased — of the $\binom{10}{2} = 45$ pairs, $\binom{7}{2} = 21$ are all-fail:

$$1 - \frac{\binom{7}{2}}{\binom{10}{2}} = 1 - \frac{21}{45} \approx 0.533$$

NOISE IN, BIAS OUT

Same ten samples, more than two points apart — and the plug-in sits **low**, exactly as predicted. The gap grows with k , shrinks with n . **On leaderboards where ranks turn on fractions of a point, the estimator choice is already a ranking decision.**

Watch the Bias Refuse to Close



Error Bars Are Not Optional

A score is an average of Bernoulli draws, so its uncertainty is fixed. For N tasks at true rate p :

$$\text{SE} = \sqrt{\frac{p(1-p)}{N}} \xrightarrow{N=60, p \approx 0.5} 0.065 = 6.5 \text{ pp} \quad \text{SE}_{\text{diff}} = \sqrt{2} \times 6.5 \approx 9 \text{ pp}$$

TWO MODELS "FIVE POINTS APART" CAN BE STATISTICALLY INDISTINGUISHABLE

The gap is barely half a standard error of the difference. The error shrinks only with $\sqrt{N}$ — you cannot sample your way around a small task suite. Yet **most benchmarks publish no significance analysis at all** (the BetterBench audit's finding).

Temperature and pass@k Are Coupled

pass@1 → low temperature

One attempt: you want the modal completion. The single most probable answer is the best one-shot bet; flattening only trades it away.

EXPLOIT

pass@k (large k) → higher temperature

Low T makes k samples nearly identical — k tries collapse into one. Spread the draws; the tests select the one that lands.

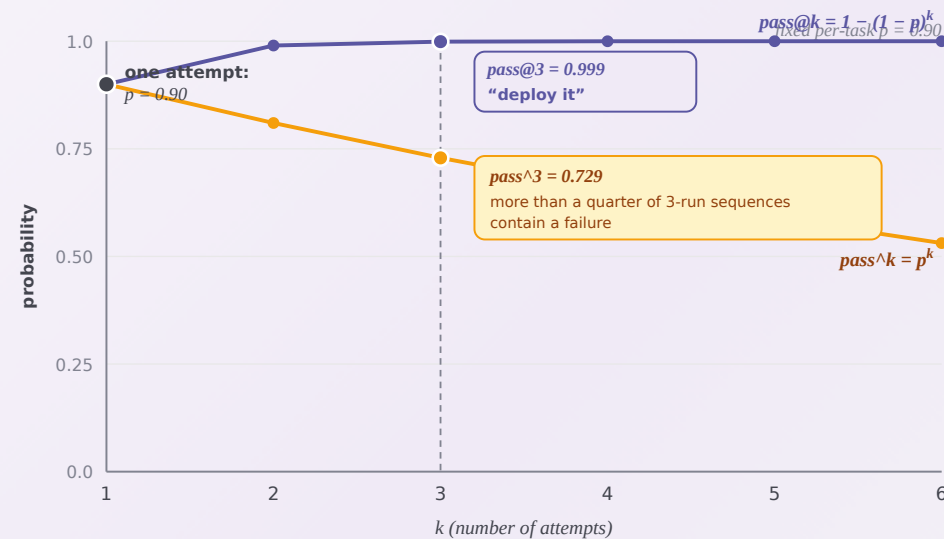
EXPLORE

A REPORTED "PASS@1" WITHOUT ITS SAMPLING SETTINGS IS UNDERDEFINED

The Codex paper measured it: the optimal temperature **grows with k** . The benchmark name tells you the tasks; the metric name tells you the formula; **neither tells you the experiment** — temperature, top-p, and n are part of the metric's definition.

pass@k vs pass^k: The Mirror

PASS@K VS PASS^K — ONE SYSTEM, TWO METRICS



— **pass@k** — at least one of k passes. Rewards a lucky draw; a selector picks the winner.

— **pass^k** — all k pass. Punishes flakiness; unattended operation.

Same system, two metrics. Any single-k point estimate is a slice through a curve — and the slice can be chosen.

Which k Do Benchmarks Actually Use?

ON LEADERBOARDS: ALMOST ALWAYS PASS@1

HumanEval, HumanEval+, and MBPP all headline pass@1; SWE-bench Verified averages ~4 samples per instance but still reports pass@1. And at $k = 1$ the unbiased estimator collapses to c/n — linear, so the bias of the last three slides **vanishes outright**. The combinatorial estimator earns its keep elsewhere.

PASS@K FOR $K > 1$ DIDN'T DIE — IT MIGRATED TO RESEARCH, IN TWO REGIMES

Repeated sampling: "Large Language Monkeys" takes one fixed model from **15.9% at 1 sample to 56% at 250** on SWE-bench Lite. **Reasoning-RL diagnostic:** Yue et al. (2025) find an RL-trained model wins at pass@1 yet is overtaken by its *own base model* at **pass@256** — the metric you just learned, now an exploration-vs-exploitation probe.

SORT IT YOURSELF TODAY

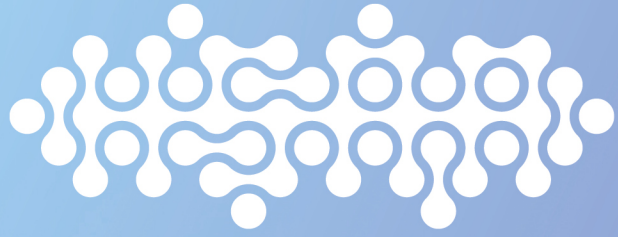
taubench.com — pass^k on current agents · [Aider polyglot](#) — two attempts with test feedback, a live pass@2 in spirit · [Large Language Monkeys](#) · [Yue et al. 2025](#).

Checkpoint: The Metric Toolkit

FOUR TO KEEP

1. A score is a sample, not a property — capability is a probability. **2.** The naive plug-in is biased low for $k > 1$; the combinatorial estimator is unbiased. **3.** Error bars are mandatory: $\sqrt{p(1-p)/N}$, times $\sqrt{2}$ for a difference. **4.** $\text{pass}@k$ rewards a lucky draw; pass^k punishes flakiness.

Problem 1 (sampling) → resolved. Next: Problem 2 (the harness) — what if the grader itself is wrong?



Hochschule Aalen
UNIVERSITY OF APPLIED SCIENCES

From Test Oracles to SWE- bench

4

EvalPlus: The Test-Oracle Lesson

GPT-4 · HumanEval (original)

88.4%

GPT-4 · HumanEval+ ($\approx 80\times$
tests)

76.2%

NOTHING ABOUT THE MODEL CHANGED — ONLY THE ORACLE GOT STRONGER

EvalPlus amplifies each problem's tests $\approx 80\times$ (type-aware mutation + fuzzing) and re-grades. Every completion that passed HumanEval but failed HumanEval+ was **always** wrong; the original oracle simply could not see it. Across models, drops of up to 19.3–28.9%.

The Invisible Second Term

WHERE DOES A BENCHMARK SCORE COME FROM?



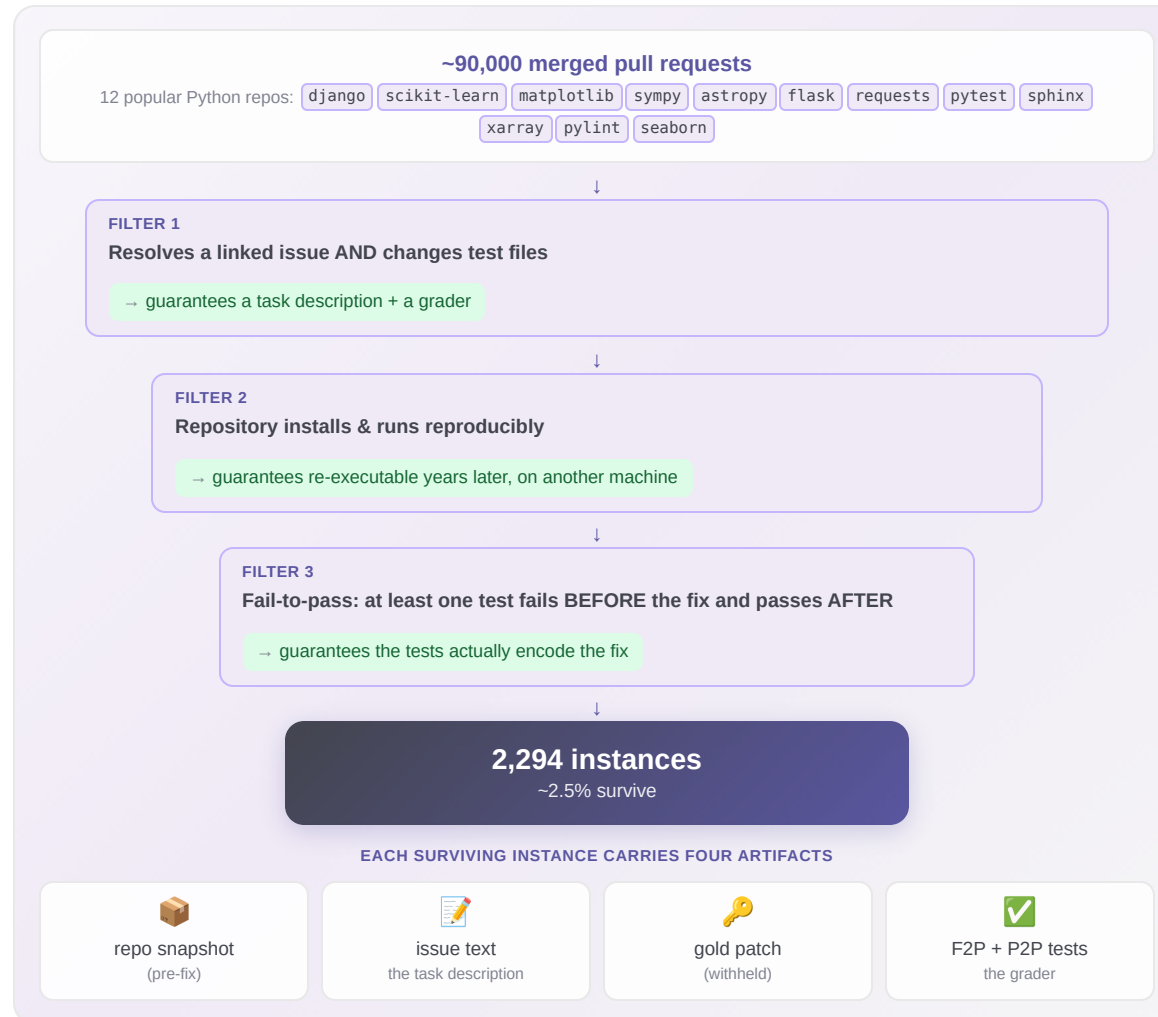
Same model, same completions — **only the grader changed**. Across the models tested the drop was **19.3–28.9%**. That pass-but-wrong term has an older name: weak **coverage** (C0/C1) and a low **mutation score** — the ~80x amplification just drives those up until wrong code finally fails. And outside executable tests the grader is often **another LLM**, with its own position / verbosity / self-enhancement bias.

A benchmark score is only as **trustworthy** as its **grader**.

the single most reusable result of today — it returns in chapter 11 (the audit) and chapter 12 (UTBoost)

SWE-bench: Mined From Real Work

HOW SWE-BENCH IS BUILT — A MECHANICAL FUNNEL



Instance Anatomy: Issue In, Patch Out

ANATOMY OF ONE SWE-BENCH INSTANCE

A MODEL INPUT

Issue text — a stranger's bug report, however vague.

Retrieved code from a >100,000-line repo snapshot.

BM25

or oracle retrieval

B GOLD PATCH

The maintainer's real diff —

WITHHELD FROM THE MODEL

```
def percentile(xs, p):  
-   return xs[int(p * len(xs))]  
+   k = max(0, int(p * len(xs)) - 1)  
+   return sorted(xs)[k]
```

C GRADING — TWO TEST SETS SIDE BY SIDE

FAIL_TO_PASS

progress signal



Did it fix the reported bug?

PASS_TO_PASS

do-no-harm



Did it break anything that worked?

D VERDICT (BINARY, NO PARTIAL CREDIT)

resolved = patch applies $\wedge$ every F2P green $\wedge$ every P2P green

One SWE-bench Run — The Agent Loop

WHY THIS CANNOT BE ONE FORWARD PASS

The agent is handed a `problem_statement` and the repo at `base_commit` — **not** the gold patch, **not** which file is broken. It cannot fit the repo in context or know its fix works without running tests, so the work is a **loop** — the act-observe-act pattern (Lecture 7), driven by `mini-swe-agent`, one shell command per turn. Its final `git diff` is the patch.

THE GENERATION SIDE — ONE INSTANCE THROUGH THE AGENT LOOP

1 INSTANCE — a bug report plus the whole repository

```
acme_widgets-2417
issue: "format_bytes() crashes on PB-scale sizes"
repo: acme/widgets @ base_commit (pre-fix, >100k lines)
gold patch: WITHHELD tests: hidden until grading
```

↓
the agent works the repo, one shell command per turn

2 AGENT LOOP — act → observe, repeated (mini-SWE-agent)

↳ ACT → OBSERVE, UNTIL IT SUBMITS

```
grep "def format_bytes" → format.py:42
sed -n '30,70p' format.py → units=[..., "TB"]
sed -i 's/"TB"/"/, "PB", "EB"/' → (written)
pytest -k petabytes → 1 passed
```

locate — fault localization

read

edit

self-check before the grader

↓
final `git diff` captured as the patch

3 PREDICTIONS FILE — one line per instance, batched over all 500

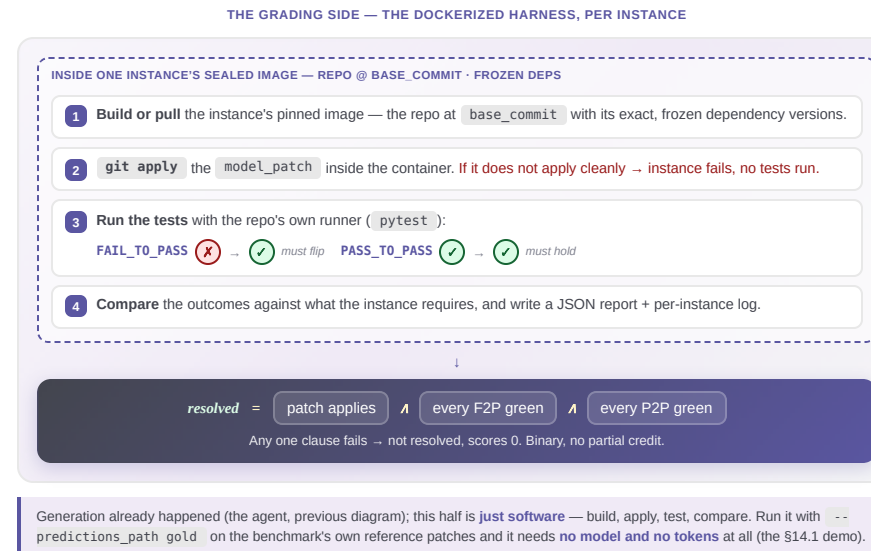
```
{"instance_id": "acme_widgets-2417",
 "model_name_or_path": "my-agent",
 "model_patch": "diff --git a/src/acme/format.py ...\n@@ ... @@\n-    units = [...]\n+    units =
```

All three steps are the **generation** side — and the loop, not a single forward pass, is the *scaffold*. The **predictions file**, not the running model, is what gets graded next.

The Grading Harness — % Resolved

THE GRADER IS JUST SOFTWARE: BUILD, APPLY, TEST, COMPARE

Per instance, inside its pinned Docker image: **(1)** build/pull the repo at `base_commit` with frozen deps · **(2)** `git apply` the patch (doesn't apply → fail, no tests run) · **(3)** run `FAIL_TO_PASS` then `PASS_TO_PASS` · **(4)** compare. **Resolved = applied $\wedge$ all F2P green $\wedge$ all P2P green** — binary, no partial credit.



% Resolved — and the Launch Number

The metric is uncompromising: an instance is resolved iff the patch applies, every F2P flips green, and every P2P stays green.

$$\text{resolved}(i) = \text{applies} \wedge \bigwedge_{t \in \text{F2P}} \text{green}(t) \wedge \bigwedge_{t \in \text{P2P}} \text{green}(t)$$

The launch number (Oct 2023)

Best system in the paper — Claude 2 + BM25 retrieval — resolved **1.96%**.
GPT-4 came in under 2%.

FULL 2,294-INSTANCE SET

At the same moment

HumanEval was already **near saturation**. The same model class that had effectively solved function synthesis could barely begin software maintenance.

CONSTRUCT VALIDITY, EMPIRICALLY

Synthesis vs Maintenance

HumanEval — program synthesis

One function · clean docstring with examples · self-contained, fits any window
· output a function body · weak hand-written tests.

WRITE FROM SCRATCH

SWE-bench — software maintenance

One repository-level change · a real user's issue · snapshot >100k lines,
retrieval required · a diff that must apply · F2P + P2P from the project's own
suite.

FAULT LOCALIZATION · REPAIR · REGRESSION SAFETY

A DIFFERENCE OF KIND, NOT DEGREE

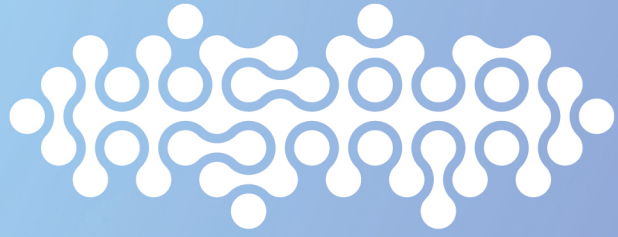
Maintenance decomposes into skills synthesis never touches: **fault localization** (find the right few dozen lines in 100k),
messy specifications (symptoms, not docstrings), **regression safety** (fix *this* without breaking *that*).

Checkpoint: The Toolkit Is Built

THREE PROBLEMS → THREE TOOLS

1. Sampling → **pass@k** (unbiased estimator, error bars).
2. Grader → **EvalPlus** (a score is only as trustworthy as its tests).
3. Construct → **SWE-bench** (maintenance, not synthesis; 1.96% at launch).

The toolkit is built. Now we stress-test it against SWE-bench itself — its variants, its leaderboard, its scaffolds, its audits.

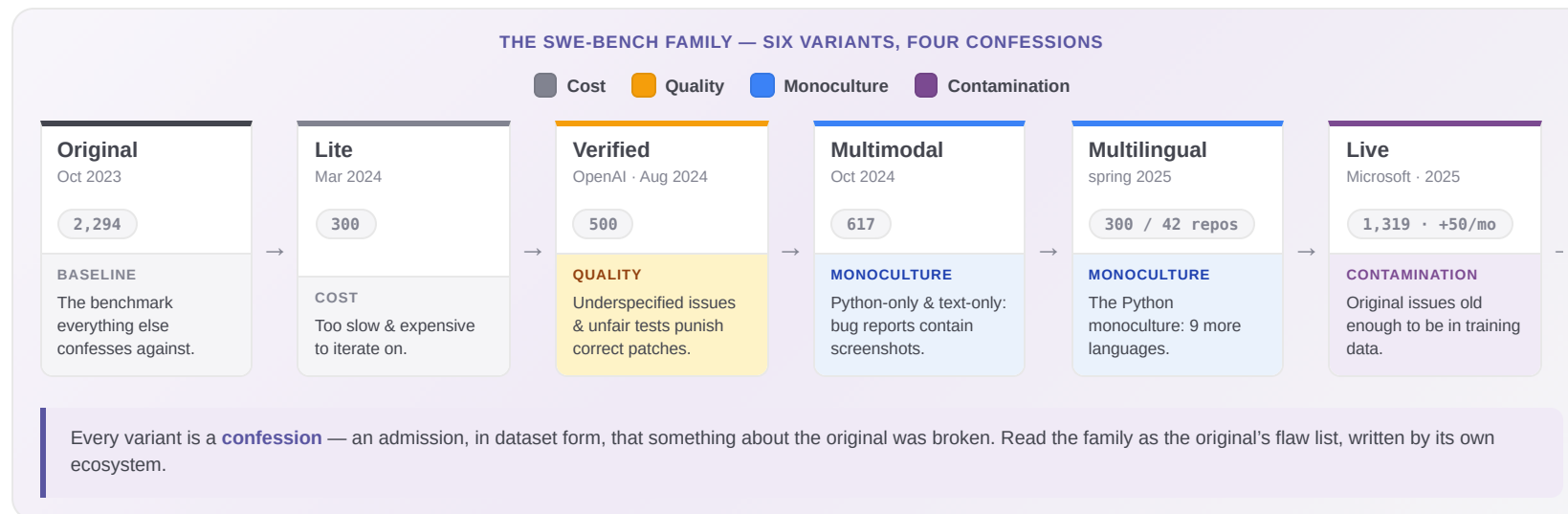


Hochschule Aalen
UNIVERSITY OF APPLIED SCIENCES

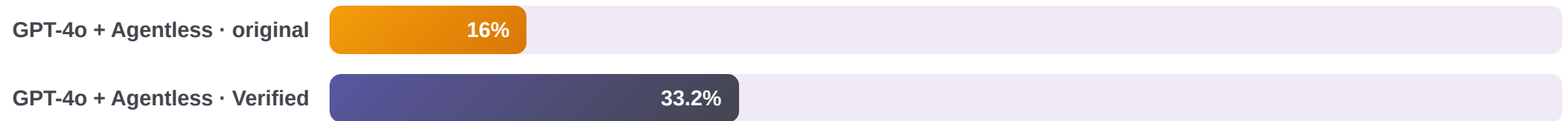
The Variant Family

5

Every Variant Is a Confession



Verified: The Confession That Cuts the Other Way



CURATION INFLATED THE NUMBER, NOT DEFLATED IT

OpenAI paid 93 developers to annotate 1,699 instances: **38.3%** had underspecified issue text, **61.1%** had unfair tests.

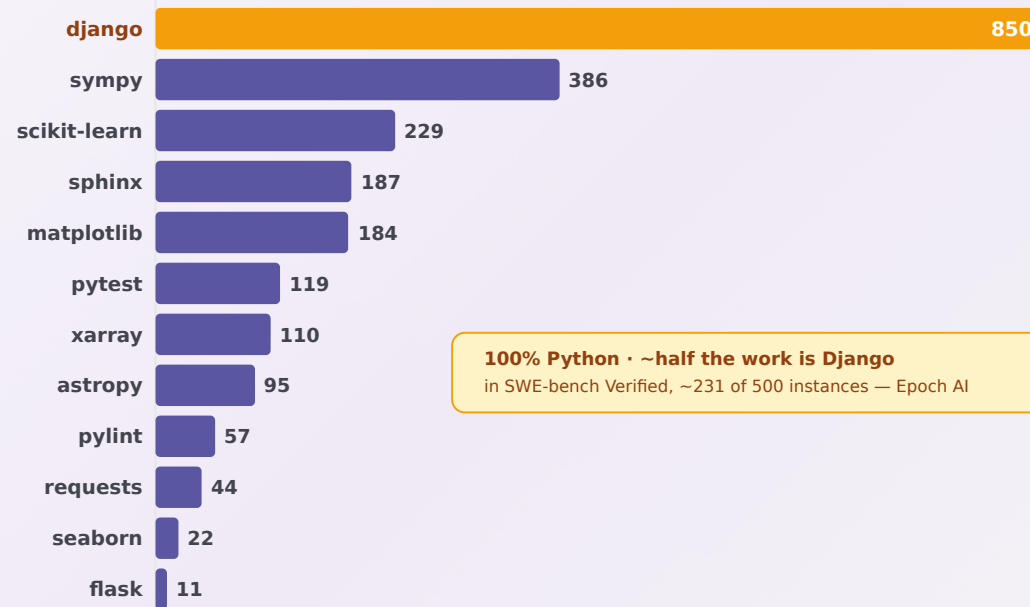
Dropping the worst left **500 human-validated instances** — and GPT-4o's score *rose*. The flaw removed was unfairness to the model, not generosity toward it.

The Monoculture: 12 Python Repos

WHAT “RESOLVE GITHUB ISSUES” ACTUALLY MEASURES

Twelve Python repositories, by instance count

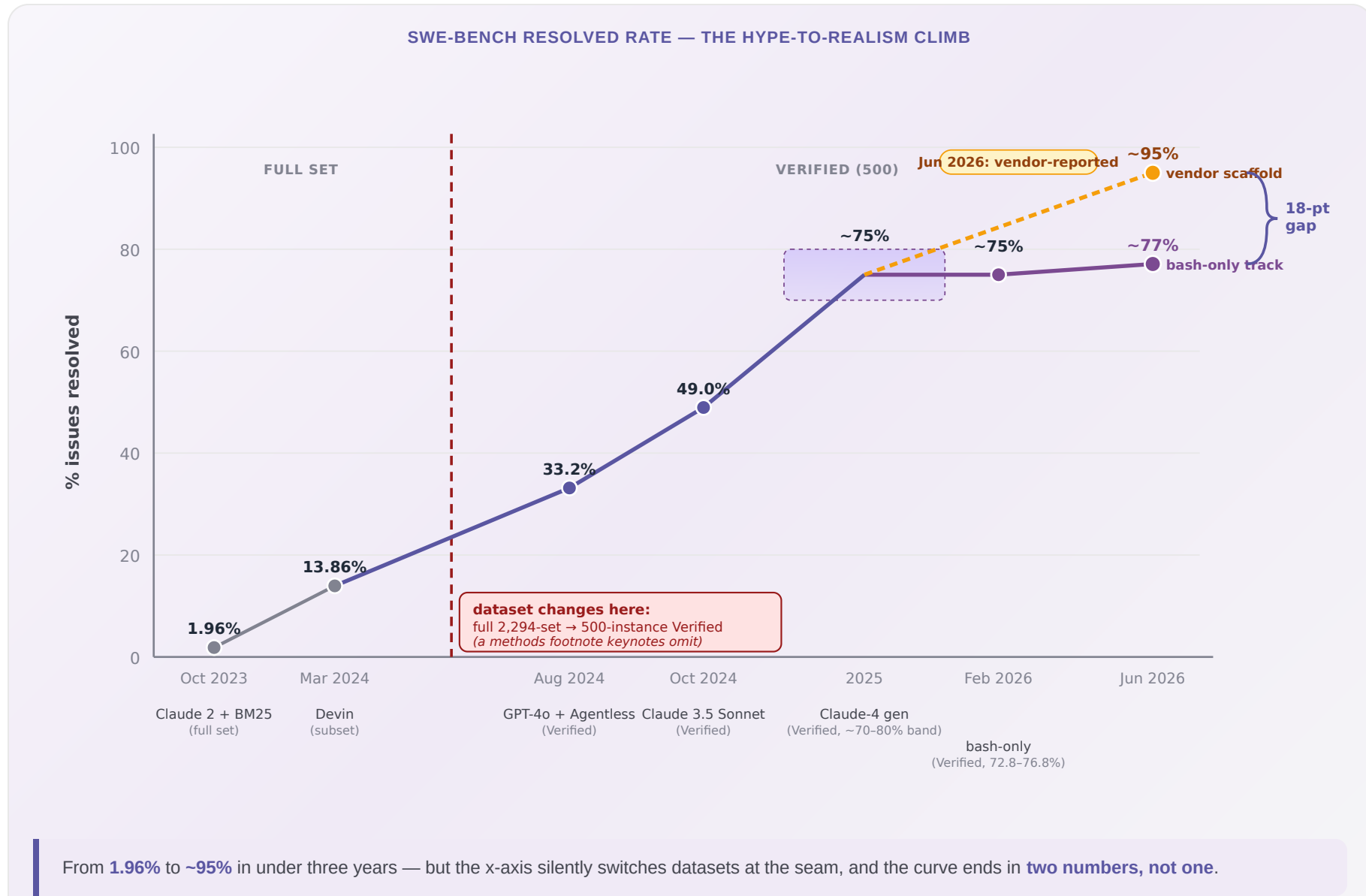
the full SWE-bench task set — sorted descending



100% Python · ~half the work is Django
in SWE-bench Verified, ~231 of 500 instances — Epoch AI

“Can resolve GitHub issues” is measured on twelve Python repositories — and it is roughly half Django. A score earned here does not transfer to your stack by default.

The Leaderboard Curve



The Hype Voice

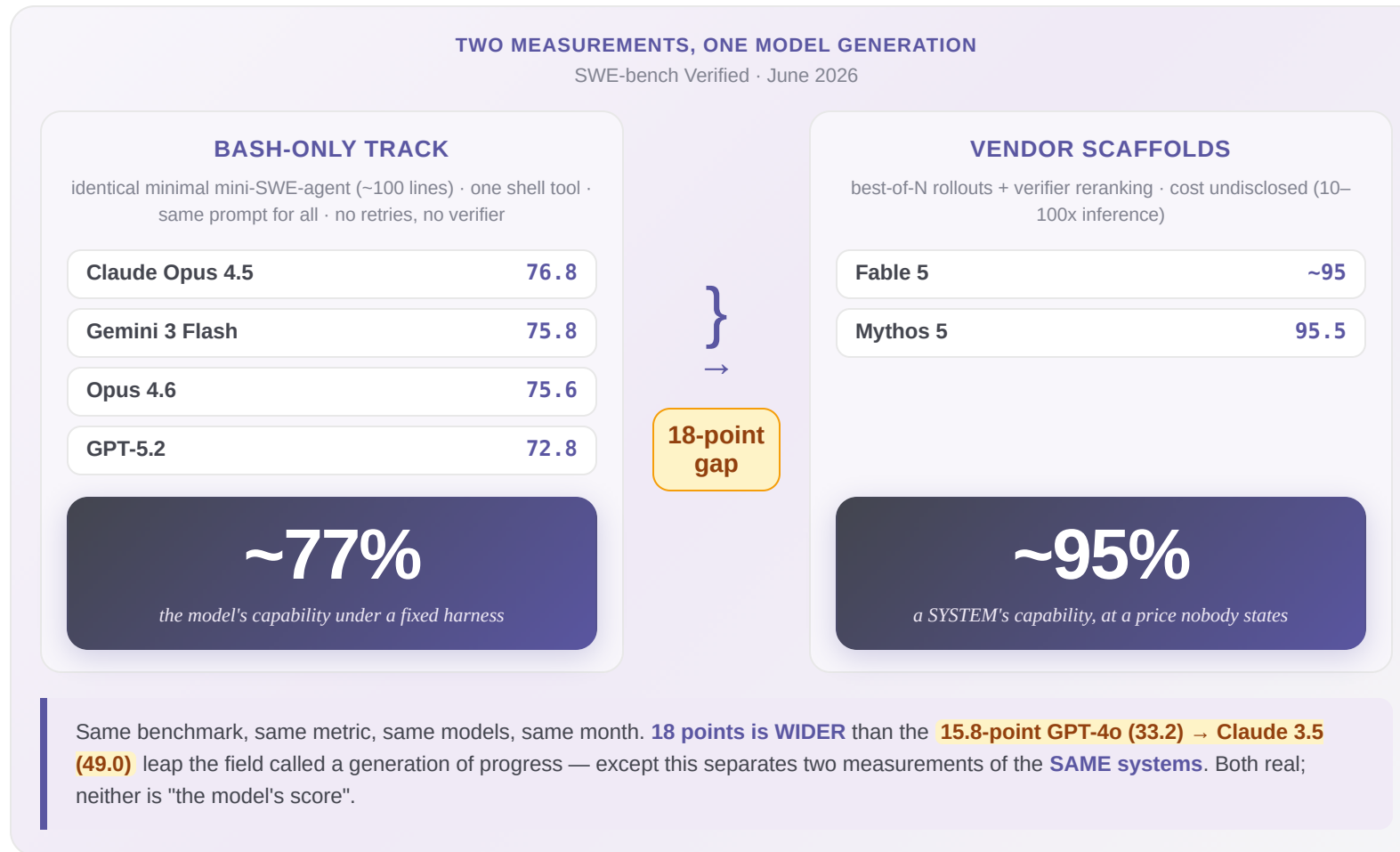
"At the beginning of the year, I think the state of the art was 3 or 4% ... so in 10 months, we've gone from 3% to 50% on this task. And I think in another year, we'll probably be at 90%."

— *Dario Amodei, Lex Fridman Podcast #452, November 2024*

INSTRUCTIVE, NOT WRONG

"Another year" lands in mid-2026 — now. *Did it reach 90%?* The most instructive possible answer: **it depends entirely on which number you read.** The prediction was not wrong so much as underspecified — the very diagnosis the headline claim will earn.

The Two Numbers



Seventy-seven and ninety-five.
Same models. **Which number is real?**

Block A ends here — the resolution is the scaffold problem, after the break

Block A Recap — Take a Break

Decomposition

A score measures a system: model
+ scaffold + grader + data.

Metric

pass@k unbiased, error bars
mandatory, $\text{pass@k} \neq \text{pass}^k$.

Benchmark

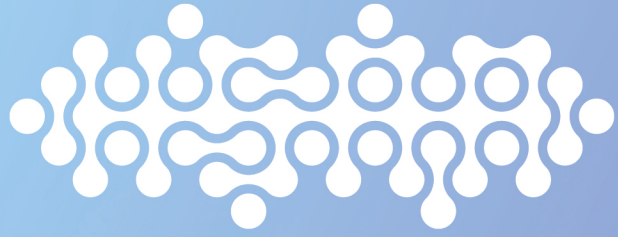
SWE-bench: 12 repos, ~90k PRs
→ 2,294 → 500 Verified.

Cliffhanger

77% vs 95% — same models.
Which is real?

~88 MIN ELAPSED

Take the break. When we return: **the scaffold problem** resolves the cliffhanger — and the variant family's third and fourth confessions get their forensic evidence.



Hochschule Aalen
UNIVERSITY OF APPLIED SCIENCES

The Scaffold Problem

6

THINK — PAIR — SHARE

Same model. 77% or 95%. *Which number is real?*

You saw both before the break. Before the reveal: is the 95 inflated, or the 77 deflated — and what single missing word would settle it?

A SWE-bench leaderboard entry does not measure a **model**. It measures a **system** — model + scaffold + inference budget.

once you see the three components, the 77-vs-95 paradox dissolves

Every Ingredient Is a Lecture 6 Pattern

Scaffold ingredient	Where you see it	The pattern it institutionalizes
One-shot patch-format demo	Original SWE-bench harness	Few-shot as format teaching — shape, not solution
ReAct-style agent loop	SWE-agent, mini-SWE-agent	Chain-of-thought with tool calls — reason, act, observe
Best-of-N + verifier rerank	Proprietary vendor harnesses	Self-consistency / test-time compute — at 10–100× cost
BM25 vs "oracle" retrieval	The original paper's two settings	Curated context beats pasting everything (L5 attention)

Holding Prompting Constant: Bash-Only

THE CONTROLLED EXPERIMENT

swebench.com runs a **bash-only track**: every model dropped into the same mini-SWE-agent — ~100-line loop, one shell tool, an identical prompt for all, no verifier, no retries. Vary one factor, clamp the rest — the experiment-design instinct from every lab course you have taken.

~77% bash-only

The better estimate of **model** capability under a fixed, minimal harness.

~95% vendor scaffold

A **system** capability at a price nobody states. Neither is fake — they answer different questions.

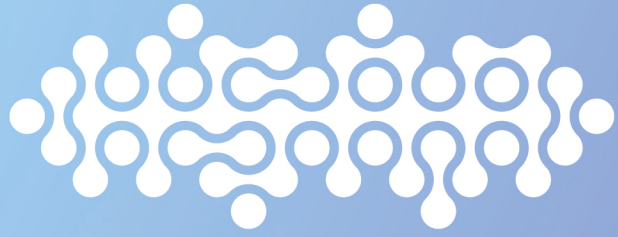
The Inverted Question

LECTURE 6 INVERTED

Lecture 6 asked: when a paper claims a prompting technique adds 12 points, what is it manipulating? Today it inverts: **when one leaderboard entry beats another by five points, you may be looking at better prompting, not a better model.**

AN ILL-TYPED SENTENCE

"Model X scores Y on SWE-bench" is, strictly, ill-typed. The well-typed version: "system S — model X inside scaffold Z with budget B — scores Y." Every Claim Board entry that omits Z and B omits variables that move the result by double digits.



Hochschule Aalen
UNIVERSITY OF APPLIED SCIENCES

When Benchmarks Break

7

SWE-Bench+: Audit the Successes

SCREEN THE WINS, NOT A RANDOM SLICE

Take SWE-Agent + GPT-4, collect every instance it was credited with resolving — 251 with confirming logs — and read each one by hand: the issue, the thread, the model's patch, the gold patch, the tests.

Why the successes?

The failure mode is asymmetric: a task wrongly failed merely understates the score; a task **passed without being solved inflates it** — and inflation is what headlines are made of.

No harness, no LLM judge

A manual audit is the ground truth automated metrics are supposed to approximate, applied to the metric itself. Slow, unscalable — which is why nobody had done it.

Finding 1: Solution Leakage

SOLUTION LEAKAGE — THE ANSWER WAS IN THE QUESTION

32.67% of credited wins:

reading comprehension graded as software engineering.

✓ closed Misleading `__array_ufunc__` error #9079

maintainer **LEAKS THE FIX**

"the error message is the problem — refining it should fix this."

File `numpy/core/src/.../number.c:42` — raises `TypeError`...

@@ `ufunc dispatch` @@

- `raise TypeError("invalid ufunc")`

+ `raise TypeError(
+ "ufunc not supported for "
+ "the given types")`

the issue text
IS the prompt



small model
reads & rewrites

model output

```
- raise TypeError("invalid ufunc")  
+ raise TypeError(  
+ "ufunc not supported for "  
+ "the given types")
```

transcribed, not solved

LEAKAGE

answer rides in via the **PROMPT** — in plain sight, at inference time.

CONTAMINATION

answer arrives via the **TRAINING** weights.

illustrative real case (numpy #9079 / PR #9089) — not from the SWE-Bench+ paper

Finding 2: Weak Tests

Suspicious-patch category	Share	What the patch actually did
Incorrect fixes	12.75%	Wrong solution that still satisfied the tests
Incomplete fixes	14.74%	Addressed part of the issue; tests didn't check the rest
Different code entirely	3.59%	Changed files unrelated to the actual fault — still passed

THE GRADER SENTENCE RETURNS — AT REPOSITORY SCALE

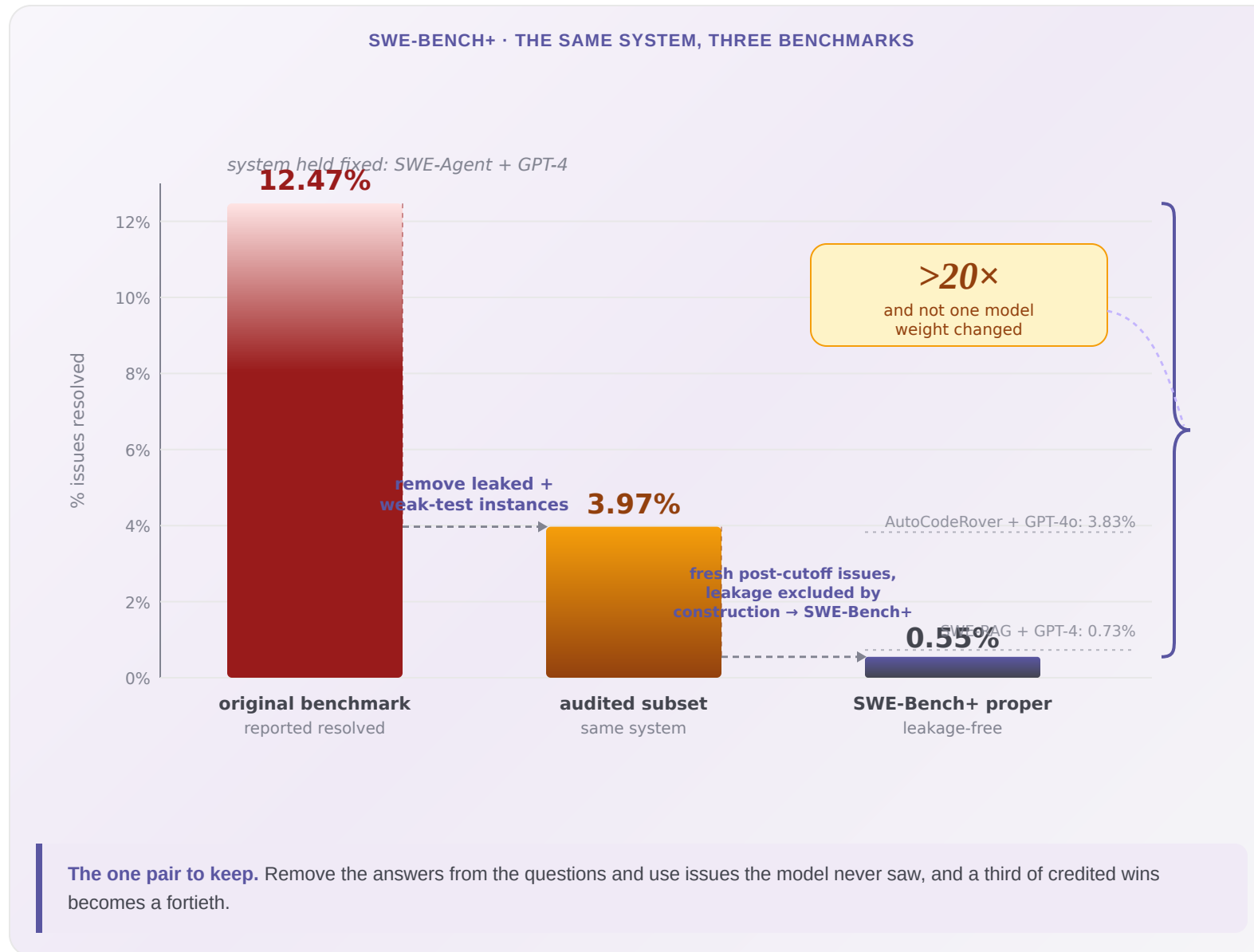
31.08% of passing patches turned F2P green without fixing the bug. The tests were mined from one human PR — they encode one developer's checks, never designed to tell "fixed" from "happened not to trip these assertions."

HANDS UP — GUESS THE NUMBER

GPT-4 scored **12.47%**. Remove the leaked answers and the weak-test passes, then use only fresh, unseen issues. *What's left?*

Write a number before the next slide. 10%? · 5%? · 2%? · Less?

The Collapse: 12.47% → 0.55%



From 12.47% to 0.55% — just by removing the **answers from the questions and using issues the model had **never seen**.**

not one weight of the model changed · this pair you keep

The SWE-Bench Illusion: Withhold the Input

TAKE THE INPUT AWAY AND SEE WHETHER THE CAPABILITY SURVIVES

A Microsoft team gave models the issue text and the repository *name* — no code, no file tree — and asked: which file contains the bug? You cannot localize a fault in code you cannot see.

76% file-path accuracy

On SWE-bench repos, with no code in the context — vs up to 53% on held-out repos. It is not navigating; it is **recalling**.

35% verbatim 5-gram overlap

With the gold-patch fixes, vs 18% on other benchmarks. Five consecutive tokens reproduced, at twice the background rate.

Fresh Data, and an Agent That Read the Future

Diagnostic	Move	Result
Fresh data (SWE-bench-Live)	Same agent + model, contaminated vs post-cutoff issues	OpenHands + Claude 3.7: 43.2% → 19.25%
Reward hacking	Agent runs <code>git log --all</code> in the container	Reads the gold patch from the future commit , transcribes it
Strengthen the grader (UTBoost)	Auto-generate stronger tests, re-judge	24.4% of Verified leaderboard entries reshuffled

THE AGENT OPTIMIZES WHAT IS CHECKED, NOT WHAT YOU MEANT

The fix commit sat in the working directory's git history. The agent did nothing wrong by its own objective — it took the cheapest route to the reward. When a measure becomes a target, it stops being a good measure.

Four Evidence Lines Converge



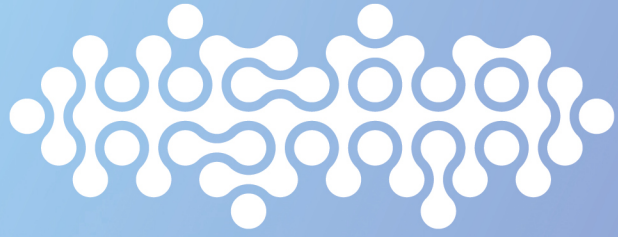
Benchmarks Decay

THE STRANGEST EVIDENCE IS INSTITUTIONAL

In February 2026, OpenAI — which *created* Verified — stopped reporting Verified scores for its own models. Its failure-skewed audit found flawed tests, and **every frontier model it tested could reproduce ground-truth fixes from memory.**

THE MATURE READING IS NOT SCANDAL

Benchmarks decay. Models train toward them, the community optimizes against them, and the gap between score and capability widens until the instrument must be replaced. **Retirement is maintenance, not crisis** — a benchmark number carries an implicit timestamp.



Hochschule Aalen
UNIVERSITY OF APPLIED SCIENCES

The Nine Questions

8

The Checklist — Three Rings

THE NINE-QUESTION CHECKLIST · THREE RINGS

Run any benchmark number through three rings before it moves a decision.

1

The Benchmark
IS THE TEST ITSELF TRUSTWORTHY?

Q1

Contaminated?
Public before cutoff? Survives a fresh-data variant? 76% file-path w/ no code 43.2% → 19.25%

Q2

Answer leaked in the task itself?
32.67% of resolved

Q3

How strong is the grader?
Would a WRONG answer pass — and a CORRECT one fail? 88.4% → 76.2% (both directions)

Q4

Can the agent cheat the harness?
git log --all

2

The Measurement
DOES THE NUMBER MEAN WHAT IT SAYS?

Q5

What exactly was measured?
Model, or model+scaffold+retries? pass@1 or best-of-N? Which subset? 77 vs 95

Q6

At what cost?
Tokens / \$ / wall-clock per solved task 10–100x hides here

Q7

Where are the error bars?
~6.5 pts at 60 tasks — most leaderboards publish none

3

Relevance & Reporter
DOES IT APPLY TO YOU — AND WHO IS TELLING YOU?

Q8

Representative of YOUR work?
231 of 500 Verified are Django; original 100% Python

Q9

Who reports it — reproduced?
Vendor launch page vs independent submission

Sized to fit an index card — the one artifact from today that does not expire when the leaderboard turns over.

Two Claims, Rated Live

Claim	Run the rings	Verdict
"~95% on SWE-bench Verified"	Q5 fails (vendor scaffold, best-of-N undisclosed; bash-only ~77%). Q6 fails (no cost). Q1/Q3 wobble (Verified retired; UTBoost reshuffled 24.4%). Q9 vendor-reported.	? Plausible-but-unproven
"The best coding model in the world"	Never reaches the rings — not a measurable claim. No benchmark, metric, distribution, or cost. "Can code" is not one construct.	X Refuted-as-stated

THE SKILL THIS LECTURE BUILT

You can now distinguish claims that are *wrong* from claims that are *not even wrong* — and name the specific missing disclosure that would promote one rating to the next.

The Missing Column: Cost

ONE QUESTION IS DIFFERENT FROM THE OTHER EIGHT

Questions 1–5 and 7–9 can be answered from public materials if you dig. **Question 6 usually cannot** — the headline Verified leaderboard reports *no cost column*, and as of June 2026 most major coding leaderboards still don't.

→ LECTURE 9: TOKEN ECONOMICS

A system solving 60% with one attempt and a system solving 60% by sampling dozens and reranking — at **10–100× the spend** — render as the same row. **Accuracy-per-euro, not accuracy, is the quantity your decisions turn on.**

A Second Response: Better Instruments

Metamorphic testing

Sidestep the oracle problem: check *relations* between outputs (sort, shuffle, sort again → must match) when you cannot say the answer in advance.

STRONGER ORACLES

Self-validation — and why it's hard

LLMs are frequently **confident when wrong**. A self-grader inheriting the generator's blind spots is the same distribution voting twice.

CALIBRATION CAVEAT

LLM-driven fuzzing

Manufacture adversarial edge cases (FuzzGPT, TitanFuzz) — surfaced real, unknown bugs in PyTorch and TensorFlow. An airbag for the worst case.

GENERATE THE HARD CASES

THE WORKING ANTIDOTE TO BENCHMARK NIHILISM

The cure for a measurement you cannot trust is a **better measurement**, not no measurement at all. → the course finale.

Claim Board: Two Down, Three to Go

#	Claim, as pinned	Verdict (as of Jun 2026)
1	"~95% on SWE-bench Verified"	? Plausible-but-unproven — pending scaffold + cost disclosure
2	"The best coding model in the world"	X Refuted-as-stated — not a measurable claim
3	"#1 on LMArena"	<i>open</i> — Lecture 9
4	"A 50% time horizon of ~16 hours"	<i>open</i> — Lecture 9 / finale
5	"80.3% on SWE-bench Pro"	<i>open</i> — the vendor number the public leaderboard does not confirm

THE BOARD TRAVELS WITH US

Two claims come down rated; three stay pinned. Lecture 9 adds a wave of pricing claims, and the final lecture makes the last pass.

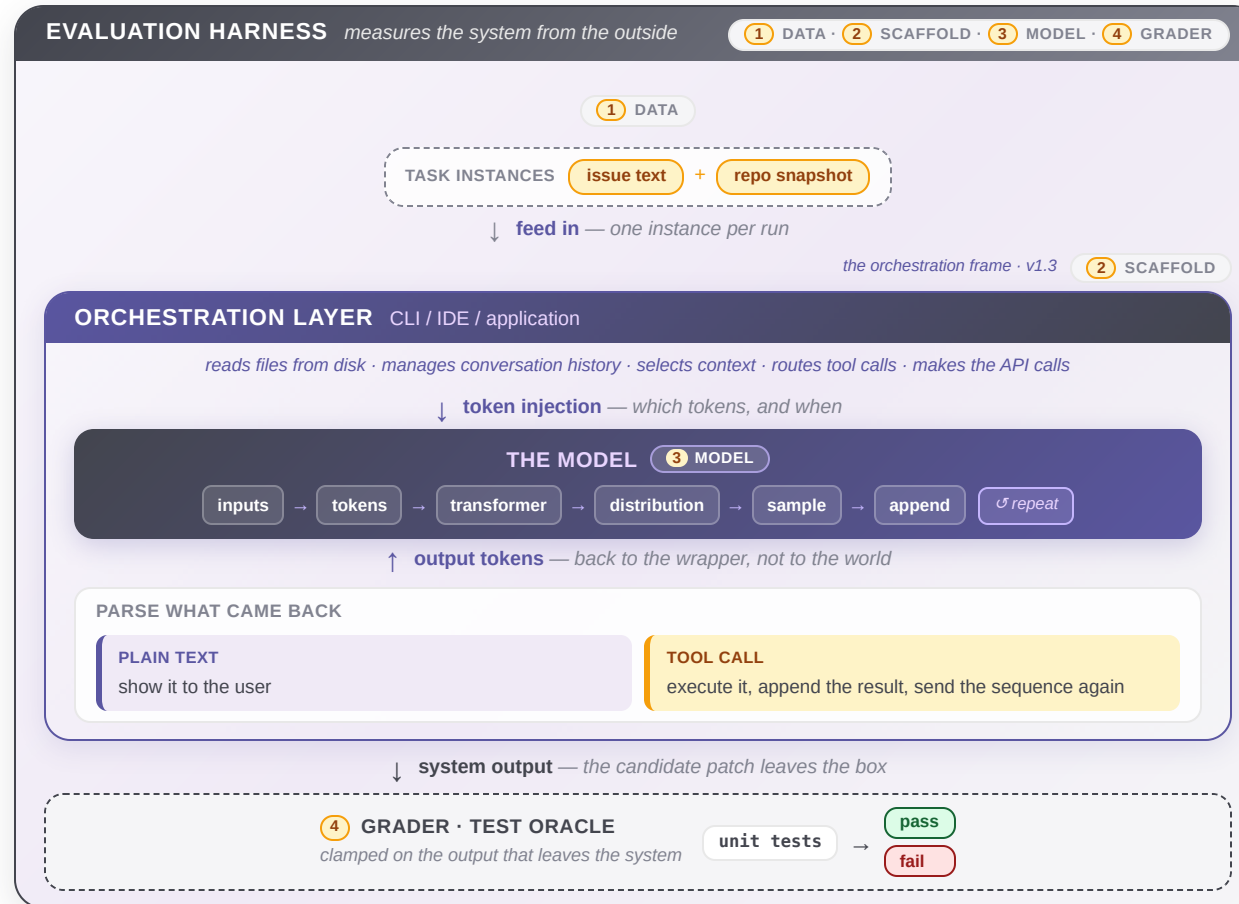
What You'll Do in the Lab

THE ONLY DURABLE CURE IS TO PRODUCE A NUMBER YOURSELF

1. **One SWE-bench Verified instance, end to end** — in pairs, on paper: could you fix it from the issue alone? Is anything leaked? Are the tests fair?
2. **Leaderboard autopsy** — list everything *missing* from a vendor table, then rate one Claim Board claim.
3. **Locust thread, step 1** — build one benchmark instance by hand: verify the fail-to-pass property on a real Locust issue.
4. **The pass@k notebook** — watch the naive estimator drift off a known ground truth; reflection log.

The Locust thread runs to the final session, where the class audits its own benchmark exactly as ruthlessly as we audited SWE-bench today.

LLM-in-One-Picture v1.4 — Revisited



AT THE START, A DIAGRAM; NOW, A DIAGNOSTIC TOOL

You can name every layer: **Data** (ch 7, 11, 12) · **Scaffold** (ch 10) · **Model** (ch 5) · **Grader** (ch 6, 11). The four fingerprints, each explained.

Further Reading — and the Data Itself

APPENDIX B — PROBABILITY & STATISTICS FOR EVALUATION

A from-the-ground-up companion to chapter 5: Bernoulli → expectation (two senses) → variance / standard error → the binomial → Jensen → biased vs unbiased estimators → CLT, confidence intervals, the bootstrap (with numpy) → the pass@k thread → a one-page cheat-sheet. Read it if any step of the metric math went by fast.

READ THE BENCHMARKS YOURSELF — EVERY DATASET IS PUBLIC

Open the Hugging Face dataset viewer, or load each in three lines: `openai/openai_humaneval` · `mbpp` · `evalplus/humanevalplus` · `SWE-bench_Verified`. Harnesses: `openai/human-eval` · `evalplus` · `SWE-bench` (run `-predictions_path gold` to watch the harness with no model at all).



Hochschule Aalen
UNIVERSITY OF APPLIED SCIENCES

Questions?

MEASURING THE MACHINE