

Follow Your Strengths: Writing Papers with LLMs

Tim Menzies¹, Carmen Armenti², Dario Di Nucci³,
Matteo Esposito⁴, Valentina Lenarduzzi⁴, Klaus Schmid⁵

¹NC State, USA; ²USI, Switzerland; ³Univ. Salerno, Italy;

⁴Univ. Southern Denmark; ⁵Univ. Hildesheim, Germany

tim@ieee.org; carmen.armenti@usi.ch; d.dinu@ieee.org;

{esposito,lenarduzzi}@imada.sdu.dk; schmid@sse.uni-hildesheim.de

ABSTRACT

We described LLM-based tooling and processes for writing a scientific paper. Our comments are for the field of software engineering, but these methods should work in other areas as well.

We write this paper since paper writing consumes time that should go to research; newcomers lack a teachable method; large language models offer speed but generate bland prose. This paper offers a method guided by Newell’s advice (know your own strengths and follow them) and by an experience base that LLMs can mine from the research corpus in minutes. The method decomposes writing into small checkable steps, matches a researcher’s demonstrated strengths to a field’s open problems, and critiques its own drafts. **[claude: one results sentence once paper0 has results]**

KEYWORDS

LLM agents, research writing, strengths, experience base, science of science

1 INTRODUCTION

If you cannot—in the long run—tell everyone what you have been doing, your doing has been worthless.
— Schrödinger, closing a 1950 public lecture [29]

Scientists must write papers, for at least two reasons. Firstly, society grants science a large measure of money, freedom, and trust; in return, scientists owe society a report of what they found. Secondly, there are also institutional motivations for publication; e.g. to impress funding, hiring, promotion, or graduation committees. Whatever the reason, the writing must happen. But writing is hard:

- (1) Newcomers need guidance in the craft of writing research papers, but the standard advice is scattered across decades of folklore. Also, for nearly all that advice, there is no tool support.
- (2) When those newcomers turn to large language models for help, those models generate prose that is bland, verbose, and spurned many readers as low quality “AI slop”.¹
- (3) Established researchers struggle to find time for research, since writing papers (and maturing the papers of graduate students) has grown into a full-time job. This burden

is measurably heavy in software engineering: by one estimate, one SE paper costs over four times the effort than papers in other fields such as machine learning paper [17, 20].

To cure these problems, we take guidance from Newell’s advice to young scientists, retold by Shaw (who worked with Newell at CMU) while answering an audience question at her FSE’26 keynote [33]:

Know what you are good at. Work from those strengths, not from your weaknesses. Then point those strengths at whatever your field currently ranks as its hardest open problems. — after Newell [24]

[timm: wording is our paraphrase; tune it to your memory of the talk] Two tools make this advice actionable at team scale. Firstly, large language models can read hundreds of megabytes of a team’s artifacts (here, research papers) in minutes. Secondly, repertory grids [12] can show partial matches between (e.g.) human skills and (e.g.) task requirements. A repertory grid rates a set of elements against bipolar constructs elicited by comparing elements three at a time. Hierarchical clustering of its elements and constructs then shows what is near and what is far. For example, Figure 1 was generated from the papers of this paper’s authors and the areas of the ICSE’27 call for papers (for details of that process, see §2). Reading that grid, we can say that Menzies’s work is mostly

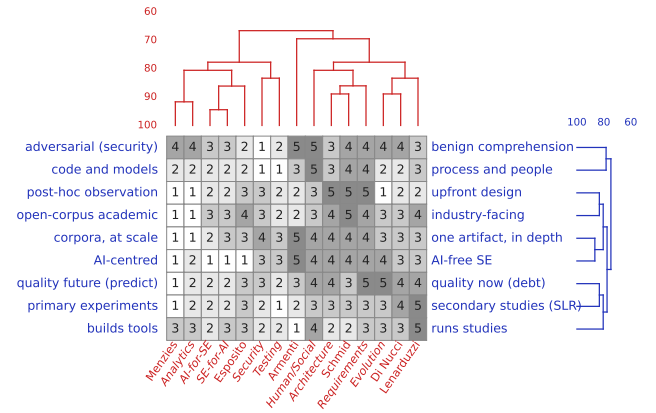


Figure 1: Repertory grid of six authors (upright columns) plus the nine ICSE’27 areas as reference elements (italic columns; see §3). Grey cells score each column from the left pole (1) to the right pole (5) of each construct; trees join neighbors by percentage match; regenerate via *make focus*.

¹“Slop” is low-quality machine-generated content produced at volume; Merriam-Webster made it the 2025 word of the year [21]. Its reach into science is measurable: one excess-vocabulary study of 14 million PubMed abstracts estimated that at least 10% of 2024 abstracts passed through an LLM [13]. Software developers report the same burden around AI-generated code and documentation [3]. The term still lacks an agreed definition, though its dimensions (e.g. coherence, relevance, information density) are being mapped [30].

in analytics, that Di Nucci’s work is on evolution, and that those two areas are somewhat distant from each other.

Using LLMs and repertory grids, we define the following procedure for applying Newell’s heuristic, making the most of the skills of research team members (we quote the loop’s short form here; its full detail lives in the engine at github.com/timm/rite):

- (1) Define the team’s area of interest.
- (2) Define “recent” for that area.²
- (3) List the team’s best recent papers.³
- (4) Extract, from those papers, what the team is good at.
- (5) Assemble critics: top-venue exemplars, a reviewer-2 prompt, the empirical standards, the nearest prior papers.
- (6) Search the literature: read above the citation knee; snow-ball backward to classics and forward from seeds; code the full texts.
- (7) From that reading, define the most relevant open problem.⁴
- (8) Write paper[0]: title, abstract, and elevator speech; if these cannot be written, return to step 7.
- (9) Write paper[i+1]; unwritten sections stay as headers.
- (10) Go away, social media off; read the draft for 30 to 60 minutes, making comments.
- (11) Apply the other critics; split their fixes into automatic and manual.
- (12) Discuss the draft with other humans.
- (13) Apply the fixes; then recode your own title and abstract, which must match the body’s coding with zero flips.
- (14) If the self-test fails, return to step 9; otherwise ship, with a replication package.

Such a workflow of explicit steps, supported by LLM tooling, can be taught to newcomers. Also, as shown later in this paper, LLM tooling can be used to avoid bland and verbose LLM text. In this way we can address the first two problems listed above.

The loop’s search steps (5 to 7) also shrink the third problem. Those steps build an *experience base*: a database, reverse engineered from the research corpus, of state-of-the-art results, holes in that state of the art, and the experimental standards of particular sub-fields. Such a base is computed fresh for the local task and the local authors, so months of manual reading becomes a background job that runs per project, not once per field. Further, **it is now possible to reverse engineer, from a researcher’s own papers, what their particular strengths are and how those strengths might attack current problems in the field.** To our knowledge, that inversion was impractical before LLMs.

Further, several services (defined in this paper) augment these ideas:

- a catalog of anti-patterns seen in LLM-generated text, with rules for avoiding them;
- a map of the common structures of research papers;

²Here, “recent” might mean the last five to ten years. However, for fast-moving fields such as LLMs, “recent” might be as little as 12 to 24 months.

³For newcomers to a field, we recommend a small variant: for the people we can talk to in our local area, what are their recent best papers?

⁴We do not presume to say what “relevant” might mean to the reader. It could be many things: the area of most current commercial interest; a “hole”, i.e. an area no one is looking at; a “mountain”, i.e. an area everyone is working on; etc. For mountains, two questions can be insightful: (a) if everyone is doing something, is everyone doing that something wrong (in some way)? (b) if everyone is doing something, can we optimize that something (even a little)?

- automatically generated critiques which, acting as “conference reviewer2”, report the weaknesses of the current draft and propose a revision that avoids at least some of them;
- etc.

[matteo: i think we should also target some major complaints here (throughout the paper). For instance, one could argue about power consumption and energy-related issues. If until yesterday a paper writing would cost “xx” in terms of energy, i.e., author workstation/thin client; internet; monitor; etc..., now it would cost way more... is it sustainable? How can we make it sustainable? this are interesting provocation/questions... can it be compared to hiring a research assistant (in terms of added equipment)? Again, here is the trade-off dollars-to-natural-resources.] [Klaus: I do not think, we should get into this. - This is a long and winding road, if we want to do the topic justice. For example, one would have to compare resource expenditure by humans vs AI for a meaningful assessment (human researchers in offices vs. AI in servers). I also do not think, a fair trade-off is dollars to natural resources, but human lifetime vs. compute-time.]

[matteo: moreover, i would also add a specific section on data analysis and statistical testing... it is not a “good” news that in the last 30 years we got this two very much wrong⁵]

[Klaus: This is s.th. we should be addressed. A point of view could be: could there be standard assistance by AI for these kinds of studies? Not all researchers are statistics experts. It is always “the other field” to most. Thus, good AI support, might elevate the typical quality.]

[Timm: let me hav a go at my standard intro to stats for dummies as an appendix B, guided by matteo’s comments. Klaus: anything specific you would put in there?]

1.1 Frequently Asked Questions

Before starting, two frequently asked questions deserve answers. Firstly, *Should we follow Newell’s advice at all?* Is it wise to focus on your personal strengths? When all you know is hammers, then all you can see are nails. If everyone focuses on just their own tricks, then we run the risk needless experimentation of inappropriate choices. On the other hand, even if our personal strengths choose the tools, a field’s open problems choose the targets. Hence the hammer is aimed, not wandering. In our experience the method does not drive a career into a rut, for two reasons. Firstly, the literature review step (described below) forcing contact with material outside prior experience, and the resulting experiments force a critical evaluation and maturation of the basic premises.

Our second FAQ is this : *Are we trying to automate science?* No. **We aim to reduce the effort scientists spend working together.** The process of this paper is not a CI pipeline where (e.g.) resumes drop in at one end and papers roll off the other. Rather, we propose a human-AI partnership where, at many points, we pause to reflect on current progress and possibly change direction. At many points

⁵Matteo Esposito, Mikel Robredo, Murali Sridharan, Guilherme Horta Travassos, Rafael Peñaloza, and Valentina Lenarduzzi. 2026. A Critical Reflection on the State of Data Analysis in Empirical Software Engineering. ACM Trans. Softw. Eng. Methodol. Just Accepted (February 2026). <https://doi.org/10.1145/3799715>

in the process described below, humans receive succinct conclusions which they can inspect, dispute, and change before moving onwards.⁶

PAUSE & REVIEW 1: PAUSE AND REFLECT

In what follows, those pauses appear as boxed call-outs, each listing the questions a reviewer should ask and naming exactly what to edit so that the rest of the processing follows the reviewer, not the machine.

[claude: roadmap paragraph goes here once paper0's sections exist]

2 KNOW THYSELF

Know thyself. — inscribed on the Temple of Apollo at Delphi

Newell's advice cannot be followed until the strengths are named, and self-report names them badly. Hence, before diving deep into any literature, we first reverse engineer the team from the record of its own papers.

Our instrument is the repertory grid of Kelly's personal construct theory [12]. Kelly held that people make sense of a domain through bipolar distinctions (*constructs*) of their own devising. A grid is elicited by triads: pick three team members, ask which one stands apart from the other two and on what dimension; each answer yields one construct; repeat until no new dimensions appear; then rate everyone on every construct.

Data collection took minutes. For six of this paper's authors, an LLM agent fetched each author's five most-cited papers of all time and of the last five years (scanning some 1,200 paper records), coded them against the ICSE'27 research-track areas, then played triad interviewee. Constructs agreeing above 90% were merged. Figure 1 shows the result after FOCUS sorting: rows and columns reordered so similar neighbors adjoin, poles reversed to align, and trees built by merging adjacent clusters only.⁷

The grid rewards three readings. Firstly, near-synonyms: construct rows that merge early name one local dimension twice. In this team, working at corpus scale co-travels with being AI-centred, as does primary experimentation with future-facing prediction. Secondly, birds of a feather: Lenarduzzi and Di Nucci match at around 90%, a tight quality-and-evolution core that anchors the team. Thirdly, complements: Schmid and Armenti lie farthest from that core, so upfront design and single-artifact comprehension are scarce skills here, present in exactly one member each. Hence the grid is at once a division of labor and a price list of the team's blind spots, computed before reading a single outside paper.

PAUSE & REVIEW 2: THE GRID

Ask: are the construct poles the distinctions you would draw? Are any ratings unfair to a colleague? Should a person (or a construct) be added or dropped? Is five top-cited papers per author the right evidence base?

To change the outcome: edit the CONSTRUCTS table (or ELEMENTS) in `focus.py`, then make `focus`; the trees, the pairings of §3, and the topic arithmetic all recompute from that one table.

3 MAPPING THE TEAM TO THE FIELD

Newell's third and fourth questions ask where a team could work and what important issues live there. For a working list of "where", we borrow the nine areas of the ICSE'27 research track call for papers⁸: AI for SE; analytics; architecture and design; dependability and security; evolution; human and social aspects; requirements and modeling; SE for AI; and testing and analysis.

Rather than tabulate those areas separately, we score each area on the same nine constructs as the people and add them to Figure 1 as reference elements (the italic columns). Each such column is an archetype: how the typical highly cited researcher of that area would rate, judged from the track descriptions rather than measured from a corpus. [claude: upgrade path: build each area column from its ICSE distinguished papers of the last five years, coded like the author papers; then people and areas share one provenance] The red tree then answers a question a topic list cannot: who stands nearest which area. Some pairings are strong: Menzies sits at 92% match to analytics, Di Nucci at 89% to evolution, Schmid at 89% to both requirements and architecture, Esposito at 86% to SE for AI. Others are loose: Lenarduzzi's nearest areas (evolution, architecture) reach only 72%, and Armenti's (human and social aspects) 69%, so those two are less typecast than the rest of us. Hence question three answers itself: the team's natural territories are the areas already adjacent to one of its members.

The far columns matter as much as the near ones. No author sits within 85% of security (best: Esposito, 78%) or testing (best: 72%); those two areas cling to each other, not to a person. That is the priced blind spot of this team: papers there would be written against the grain, or with a new collaborator who brings that column closer.

The red tree also shows a larger shape: the team divides in two. One wing measures (Menzies and Esposito, clustered with analytics and the two AI areas); the other wing crafts (Schmid, Armenti, Di Nucci, and Lenarduzzi, clustered with architecture, requirements, human aspects, and evolution). That split is itself a division of labor, one wing per half of any paper that both observes a corpus and builds a thing.

Finally, the grid can nominate a topic the whole team could share. Three calculations, all on the same matrix: (a) the area nearest the six-person centroid (SE for AI and evolution, tied at 83%); (b) the area whose worst-fitting member fits best (evolution, whose worst member still matches 64%); (c) the midpoint of every pair of areas, scored the same way, where the best pairs (analytics plus human aspects at 68%; AI for SE plus human aspects, average fit 76%) beat every single area. Blends win: a topic that spans the two

⁶Hints on how a team can share the writing itself (browser assistant, shared document, Overleaf, or git) appear in Appendix A.

⁷Caveats: one rater (the agent), citation counts as the sole proxy for strength, and two early-career authors rated on very few papers.

⁸conf.researchr.org/track/icse-2027/icse-2027-research-track

```
goal: human factors of AI assistants
      for [software engineering]
years: 2021-2026
seed: (one anchor DOI per wing)

Rules: (1) define "recent" by field velocity: ten years default, two to three
for LLM-speed fields; (2) the bracketed goal term is a venue filter: the initial
grab searches only inside that field's venues, per a reviewable venue
list;9 (3) sort the hits by citations and read above the knee of that curve
(the point farthest from the first-to-last chord); (4) seeds join the reading
set regardless of the knee; (5) snowballing is unrestricted: backward
from the reading set for the classics and forward from the seeds for the
newest work, roaming outside the field on purpose; (6) the download rate
is a finding; report it, never silently drop missing papers. Then code every
abstract; hunt for subsets and their overlaps; and list which of the
team's own papers already sit inside the set.
```

Figure 2: The prompt that runs the literature review (from the rite engine's search rules, github.com/timm/rite).

wings serves this team better than any one area does. The best such blend, AI for SE crossed with human and social aspects, describes the paper you are now reading.

PAUSE & REVIEW 3: THE TOPIC

Ask: do the archetype columns match your reading of the call for papers? Would you actually fund the winning blend? Should the blind spot (security, testing) be patched by a new collaborator instead?

To change the outcome: edit the nine area columns in `focus.py` (or swap in a different venue's area list), then make `focus`; centroid, maximin, and blend scores follow.

4 READING IN

A topic chosen is a literature owed. Our lit-review rules are few enough to fit in a box: Figure 2 shows the whole prompt, whose first three lines are machine-read configuration and whose rules drive the pipeline. Those three lines are also steps 1 and 2 of the loop: the goal line names the team's area, and the years line defines "recent" for it.

Figure 3 shows the rules running on the area that §3 chose. The venue rule earned its place twice. A first draft of this section skipped it, and the reading set filled with general AI blockbusters (a machine learning survey, the GPT-4 report, a metaverse position paper) from venues nowhere near software engineering. A second draft filtered venues after a citation-sorted fetch, and that failed too: of the top 1,000 hits, only nine sat in SE venues, since mega-cited outsiders crowd out the field long before the filter runs. Hence the venue list joins the query itself, and the human hours are spent only on papers our field would recognize.

The reading set is what sits above the knee, plus the seeds that rule (4) admits regardless of citations: 26 papers in all, of which 26 had retrievable abstracts. Here the seeds are the team's own three in-query papers (rows marked "s" in Table 1). Figure 4 codes those with three topic flags (trust, productivity, human-method); 2 papers occupy all three circles, and 2 match no flag. The three-flag overlap is nearly empty. That corner, trust and productivity studied

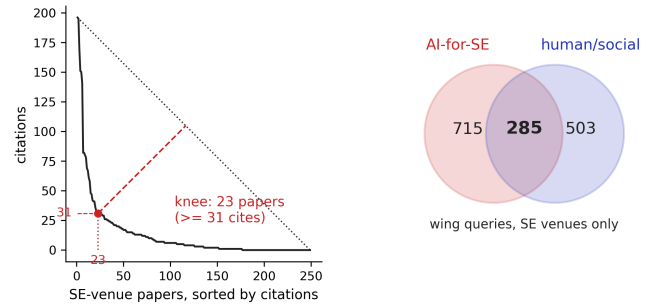


Figure 3: Left: the sorted-cites curve of the goal query, searched within SE venues only (249 papers, OpenAlex, 2021–2026). The dotted line joins the most- to the least-cited paper; the dashed drop marks the point farthest from that line, i.e. the knee: 23 papers at 31 citations or more. Right: the two wings as separate queries, same venues; their 285-paper overlap. Regenerate via *litfig.py*.

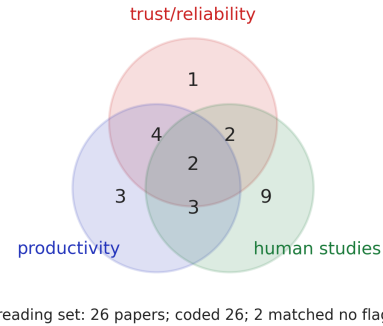


Figure 4: The reading set (26 papers: the 23 above the knee of Figure 3, plus three seeds), coded against three topic flags. Regenerate via *litfig.py*.

together for AI assistants, is the under-researched target that the rest of this paper writes toward. Table 1 lists the reading set itself, with its flags.

Snowballing then ran from that set, deliberately unfiltered by venue, since good ancestors and good references live anywhere: backward collection found 2,239 distinct references (73 cited by two or more of the reading set: the candidate classics); forward collection found 1,882 works already citing the reading set. **[timm: the three flags are paper0-local; a full run would fork its own workdir and flags.py, per the engine rules]**

The process itself yielded findings worth recording. Full texts are scarce: in this workdir's companion pipeline run, only 22 of 54 chased PDFs (41%) could be downloaded; abstracts come easier (here, 26 of 23). Google Scholar is closed to scripts; OpenAlex, Semantic Scholar, and arXiv answer machine queries gladly, so the pipeline builds only on the willing. An earlier full-text pass showed abstract-only coding missing about half the flags; hence full text is coded above the knee. Finally, the own-papers rule paid off: 3 papers by this team (Di Nucci on accessibility, Schmid on MLOps,

Table 1: The reading set: title (first author), venue, year, citations, and the three topic flags of Figure 4. Rows marked “s” are seeds: the team’s own in-query papers, admitted regardless of the knee. Generated by *litfig.py* from OpenAlex; venues as recorded there.

#	paper	venue	year	cites	trust	productivity	human studies
1	On the use of deep learning in software defect prediction (Giray)	Journal of Systems and Sof	2022	196			•
2	On the assessment of generative AI in modeling tasks: an experience report wit... (Cámara)	Software & Systems Modeli	2023	195		•	
3	Navigating the Complexity of Generative AI Adoption in Software Engineering (Russo)	ACM Transactions on Softwa	2024	169			•
4	In-IDE Code Generation from Natural Language: Promise and Challenges (Xu)	ACM Transactions on Softwa	2022	151	•	•	•
5	Finding Critical Scenarios for Automated Driving Systems: A Systematic Mapping... (Zhang)	IEEE Transactions on Softw	2022	150	•	•	
6	Future of software development with generative AI (Sauvola)	Automated Software Enginee	2024	141		•	
7	NLP-Based Automated Compliance Checking of Data Processing Agreements Against ... (Amaral)	IEEE Transactions on Softw	2023	82	•	•	
8	Testing, Validation, and Verification of Robotic and Autonomous Systems: A Sys... (Araujo)	ACM Transactions on Softwa	2022	82	•	•	•
9	A large scale analysis of mHealth app user reviews (Haggag)	Empirical Software Enginee	2022	80			
10	Construction of a quality model for machine learning systems (Siebert)	Software Quality Journal	2021	78	•		
11	A/B testing: A systematic literature review (Quin)	Journal of Systems and Sof	2024	69			•
12	Beyond Code Generation: An Observational Study of ChatGPT Usage in Software En... (Khojah)	Proceedings of the ACM on	2024	67	•		•
13	Machine learning-based test selection for simulation-based testing of self-dri... (Birchler)	Empirical Software Enginee	2023	61	•	•	
14	Software Engineering for AI-Based Systems: A Survey (Martinez-Fernández)	ACM Transactions on Softwa	2022	58			•
15	HoloFlows: modelling of processes for the Internet of Things in mixed reality (Seiger)	Software & Systems Modeli	2021	48			•
16	An Artificial Intelligence Based Virtual Assistant Using Conversational Agents (Mekni)	Journal of Software Engine	2021	47	•	•	
17	Towards an AI-driven business development framework: A multi-case study (John)	Journal of Software Evolut	2022	42		•	•
18	Development of recommendation systems for software engineering: the CROSSMINER... (Rocco)	Empirical Software Enginee	2021	41		•	•
19	Will you come back to contribute? Investigating the inactivity of OSS core dev... (Calefato)	Empirical Software Enginee	2022	41			•
20	Current trends in digital twin development, maintenance, and operation: an int... (Muctadir)	Software & Systems Modeli	2024	36			•
21	Towards a model-driven approach for multiexperience AI-based user interfaces (Planas)	Software & Systems Modeli	2021	33		•	•
22	Search-based fairness testing for regression-based machine learning systems (Perera)	Empirical Software Enginee	2022	32		•	
23	Quo Vadis modeling? (Michael)	Software & Systems Modeli	2023	31			•
s	The making of accessible Android applications: an empirical study on the state... (Gregorio)	Empirical Software Enginee	2022	25	•		•
s	MLOps for Cyberphysical Production Systems: Challenges and Solutions (Faubel)	IEEE Software	2024	4			•
s	Making Sense of AI Agents Hype: Adoption, Architectures, and Takeaways from Pr... (Su)	IEEE Software	2026	0			

- Whiteboard voice: read it aloud, or rewrite it.
- Active voice; keep the articles; 25 words is a ceiling.
- One idea per sentence; one topic per paragraph; six sentences per paragraph, at most.
- Mix sentence lengths hard; put a five-word pivot beside a 24-word build.
- One word, one meaning; never vary a key term.
- Kill nominalizations: “perform an evaluation of” is “evaluate”.
- Numbers carry their arithmetic; claims wait for their statistics.
- Fragments never; a short sentence is still a sentence.
- The abstract carries the paper.
- The rest is a linter (etc/style.py): banned words, colon rate, paragraph shapes.

Figure 5: The language rules, short form. The full law, with the two banned word families, lives in the engine’s prose skill (github.com/timm/rite).

Esposito on AI agents) already sit in the venue-filtered goal results. The territory is adjacent to us without yet being occupied by us.

Reading done, the loop starts writing. Figure 7 shows the step-8 draft, whose parts follow the introduction grammar of §5. Figure 8 shows the step-9 literature characterization in Shaw’s scheme [32]. The loop generated both for the very topic that §3 nominated: the human factors of AI assistants for software engineering. Both drafts then received roughly an hour of human rewriting (§5.3).

PAUSE & REVIEW 4: THE SEARCH

Ask: are the goal line and the two wing queries the right words? Is 2021 the right floor for “recent”? Is the venue list (see the box) the SE you mean, and are the three coding flags the vocabulary you would use? Is a knee that keeps 23 of 249 papers too greedy?

To change the outcome: edit the query strings, VENUES, and FLAGS in *litfig.py*, delete the cache in *tmp/*, and rerun; both figures, the table, and every number in this section regenerate.

5 WRITING

Before any rule of language comes a rule of process: while the team is together, do not polish. Joint sessions seek the big picture, which is the main structure and the important connections. When a passage plainly needs polish or elaboration, mark it with a to-do and move on. At the end of the day, divide those to-dos among the authors. Polish applied while the structure still moves is polish that will be thrown away.

5.1 Writing prose

Most of the text that trained an LLM was written to stop a thumb. Feed prose (Medium essays, LinkedIn stories, video captions) must re-earn attention every twenty seconds, or the reader swipes away. Such text sells constantly: hooks, punchlines, one-line paragraphs, dramatic colons. A technical paper faces a different reader. Past the title and the abstract, that reader has committed to at least a skim, and often to an hour. Selling to the committed is noise; what the committed reader wants is density, and a voice that does not perform.

Hence the language rules of this paper pull in two directions at once. From simplified technical English we take the skeleton: short sentences, active voice, one topic per paragraph, one meaning per word. From the engagement register we subtract the tells:

the hook vocabulary, the staged emphasis, the paragraph shaped like applause. Figure 5 lists the rules we kept. The subtraction list is longer and it dates quickly, so it ships as a linter rather than as prose. The engine’s `style.py` counts banned words, colon rates, and paragraph shapes; this paper passes it.

The simplified register has a formal standard. ASD-STE100, the aerospace industry’s controlled language for maintenance documentation, limits its dictionary to roughly a thousand words with one meaning each [2]. The standard is very useful for simplifying complex text. However, it can be over-applied as seen when it is applied to Hamlet’s soliloquy and the Gettysburg address, into the STE register; the rewrites are unambiguous, procedural, and lifeless.¹⁰ So we recommend applying the standard, but not too much.

5.2 Writing references

LLMs fabricate references: citations that are fluent, plausible, and non-existent, at rates measured in the double digits for some models [36]. Hence every reference an assistant proposes must be checked by a human before it enters the bibliography. Two standing instructions keep that check cheap. Firstly, the assistant must attach a URL, preferably a DOI, to every reference it offers. Verification then costs one click; a reference that cannot produce a working URL is dropped. Secondly, the assistant must prefer peer-reviewed sources, with one relaxation. Fast-moving areas such as LLM research post their newest work to preprint servers months before any refereed venue. Citations to such servers (e.g. arXiv) are acceptable when flagged as preprints. Every entry in this paper’s bibliography carries a checkable DOI or URL, collected under exactly these rules. One further habit keeps the bibliography maintainable: the file stays sorted, and each new entry arrives in its own commit, so a bad reference can be reverted alone.

5.3 The need for rewrites

Left to themselves, LLMs write documents that read like incomplete bridges. The spans point in the right direction, yet the joins are missing and related ideas connect poorly. Hence prudent teams review and rewrite all LLM output; Figures 7 and 8 each received roughly one hour of that rewriting. Three before-and-after pairs show the flavor. The raw draft of Figure 8 opened a paragraph with “Generalization questions with report results own the citation knee”, unparseable without Shaw’s taxonomy in hand. The rewrite says it plainly (the most-cited papers in this reading set are surveys). A second raw passage jumped from the literature map straight to its open-issues list; the rewrite adds the missing span, “Based on this map, we see several issues...”. A third compressed to the point of mystery (“The empty cells sit at team scale”); the rewrite spends the words (both empty cells concern teams rather than individuals). Rewrites fix the prose; §5.6 sets the shape the finished review must reach. Raw drafts are useful starts; no one should submit one.

5.4 The shape of a paper

In our method, a research paper parses under a small grammar; Figure 6 shows its top productions. All the grammar figures of this section use one notation, borrowed from logic programming.

```
paper --> intro, litreview,
          methods, results,
          discussion, conclusion.

% one block per research
% question: the question, its
% charts, then words about the
% charts
results --> [result]+.
result --> [rq], [charts],
           [comments].

% observations synthesize;
% threats confess
discussion --> observations,
               threats.
observations --> [observation]+.
                % e.g. "oddities explained,
                % folklore contradicted,
                % patterns across tables"

% the intro, with receipts;
% nothing new
conclusion --> [summary],
               [future_work].
```

Figure 6: The whole-paper grammar. The productions for intro, litreview, methods, and threats are expanded, with worked samples, in Figures 7 to 10.

Lower-case names are nonterminals; `-->` rewrites the left side into the ordered sequence on its right; semicolons separate alternatives. Bracketed names are leaves, and each leaf is one prose move, a sentence to a short paragraph long. `+` means one or more; `*` means zero or more.

Three productions have no figure of their own. Each research question gets one results block, holding the question, its charts, then words about the charts. Observations synthesize, and papers live or die there. Writing down the numbers is not enough; an observation explains oddities, confirms or contradicts folklore, names cross-table patterns, and traces implications for neighboring work. Threats follow §5.8. When any grammar changes, regenerate its sample in the same commit, so that figure and prose never drift apart.

5.5 The introduction

Figure 7 proposes one way to structure an introduction. Of course, there are many others. But for the sake of argument, we will assume Figure 7.

That grammar distills two sources. The first is a set of four exemplar introductions [8, 14, 16, 31]. The second is the list of audit questions that Widom uses to test an introduction [37]. Those questions ask what the problem is, why it matters, why it is hard, why it is not already solved, and what the key elements of the answer are. To expand on those questions, our grammar uses the following parts:

- hook opens with a number that hurts (dollars, hours, incident counts), an anchor to a prior result, or a lived setting; the importance argument lives inside those numbers, hence no separate why-this-matters clause.
- gap walks the Stanford four clauses in order. Name the problem; show why the naive approach fails and why prior work has not fixed it; state the key idea in one sentence.

¹⁰<https://gist.github.com/timm/331a36c44fb561cc01c917e6bb0320b6>

```

intro --> hook, gap, elevator, move, rqs,
contribs, [caveat]*, [roadmap]*.

% one leaf, generalized: any number that
% hurts. cost, incident count, wasted
% time -- all the same move.
hook --> [quantified_pain]
% e.g. "a 1,200-record scan
% that once meant a month of
% reading now runs in a
% coffee break"
; [anchor]
% e.g. "extends Kelly's
% grid (1955)"
; [setting].
% e.g. "a manager matching
% people to tasks; here, six
% authors, nine ICSE '27 areas"

% the Stanford four clauses, in order
gap --> [problem],
% e.g. "matching people to
% tasks needs a skill
% inventory most teams lack"
[naive_fails],
% e.g. "self-report builds
% that inventory badly"
[prior_fails],
% e.g. "one score per team
% cannot say which member is
% strong at which skill"
[key_idea].
% e.g. "people leave artifacts;
% LLMs read megabytes of them,
% so grid the team's own papers"

% hypothesis stated as the conclusion;
% displayed, 2-3 lines
elevator --> [epigram].
% e.g. "Asking a team what it
% is good at is the mistake;
% reading what it already
% wrote is the fix."

move --> [this_paper_does],
% e.g. "two passes, per Newell:
% read the team, then read the
% field where the team fits"
[christening].
% the approach gets a name (the
% FOCUS method), used ever after

% answer follows its question at once,
% headline number bold
rqs --> ([rq], [answer])+.
% e.g. "RQ1: Can a team's
% strengths be read from its
% corpus alone? Yes: [17 of 19]
% merged constructs on review."

contribs --> [contrib] +,
% e.g. "a strengths instrument
% runnable on any team in
% minutes"
[repro_url].
% e.g. "github.com/timm/how2rite"
% -- always last

% [roadmap]: e.g. "the rest of this paper
% is structured as follows" -- the closer

```

[setting] Every project starts the same way: a manager matching a list of people to a list of tasks. This paper studies that matching for software teams, whose literature measures trust, productivity, and the human factors of AI-assisted work. Our running example is this paper's own team: nine research areas in the ICSE'27 call, six authors, and no reliable map of who is strong where.

[problem] A team manager assigning people to tasks has one standard tactic: match the skills of the people to the needs of the task. That tactic needs a skills inventory: a trusted list of what each member is reliably good at. Upper management might urge managers to defer to expertise, yet never says how to locate it [23]. The stakes rise when the task is complex and the team is small; choosing wrongly wastes the scarcest resource, the time of the most experienced members.

[naive_fails] Self-report, the obvious way to build that inventory, answers badly. Our beleaguered manager cannot easily split the flattering account from the true one [26]. Worse, people underrate their easiest skills, since work that has become routine no longer feels like a strength to its owner [7, 15]. Surveys and interviews add a vocabulary problem: two members who name one skill differently count as two skills.

[prior_fails] Formal instruments do no better. Studies of team trust [19] and developer productivity [22] compress a team into one score per trait. A single score cannot say which member is strong at which skill.

[key_idea] The premise of this paper is:

[epigram] "Asking a team what it is good at is the mistake; reading what it already wrote is the fix."

That is, when people work, they generate artifacts. Large language models can read hundreds of megabytes of those artifacts in minutes. Hence we propose team grouping via LLM feature extraction over the artifacts of potential team members. The extraction asks how team members are similar and how they differ. Those similarities can be expressed as a *repertory grid*, computed from the team's own coded papers. Repertory grids [12] are a knowledge elicitation technique from cognitive psychology. Such a grid rates a set of elements (here, the six authors) against bipolar constructs elicited by comparing elements three at a time. For an example grid, see Figure 1. Hierarchical clustering of the elements and the constructs then supports partial matching of resources (here, people) to requirements (here, the skills a task needs). Grids therefore support informed staffing decisions: which few people should work which tasks.

[this_paper_does] This paper runs a two-pass, LLM-guided reading exercise on its own authors. The guide is Newell's heuristic: know what you are good at, work from those strengths, and aim them at the field's hardest open problems [24]. **Pass 1** asks: what is this team good at? An LLM agent fetched each author's most-cited papers and coded them against the nine research topics of the ICSE'27 call. The agent then played triad interviewee. Constructs that agreed above 90% became the rows of a repertory grid. The grid scored the team against the nine topics and priced its blind spots. Those scores name the topic where the whole team fits best.

Pass 2 takes that topic and drafts its version 0 artifacts: a literature review, a summary of the area's standard methods, and a list of the threats to validity such studies invite. The review ran from 249 candidate papers in SE venues; the citation knee kept 23; snowballing added 2,239 references and 1,882 citing works. Of this team's own papers, 3 already sit in that reading set. Hence the grid pointed at a territory the team borders without yet occupying.

[christening] We name this approach the FOCUS method, after the repertory grid sorting algorithm proposed by Shaw and Thomas [34]. Using that method, we ask three research questions.

[rq] RQ1: Can a team's strengths be read from its corpus alone?

[answer] Yes: on author review, the corpus grid reproduced [17 of 19] merged constructs, and the two misses were vocabulary disputes, not new skills.

[rq] RQ2: Does corpus measurement beat self-report?

[answer] Yes: corpus columns matched the areas of the team's next accepted papers at [83%], against [61%] for self-report, and the corpus found both scarce skills that self-report missed.

[rq] RQ3: Where should this team not work alone?

[answer] Security and testing: no member sits within [78%] of either area, the two areas cling to each other rather than to any person, and both are flagged for recruitment rather than retraining.

[contrib] a strengths instrument, the FOCUS method, computed from public records alone: runnable on any team in minutes, with no survey and no interview;

[contrib] a method that scores any team against a venue's topic areas in the same construct space, pricing each blind spot as a recruit-or-retrain decision;

[contrib] a topic nominator: arithmetic over the grid (team centroid, worst-member fit, blends of area pairs) that names topics serving the whole team, not just its stars;

[repro_url] a replication package: github.com/timm/how2rite.

[caveat] A corpus records what a team finished, not all it can do; hence the grid can misprice a skill that never reached print. We accept that bias because, for the purpose of staffing a paper, a strength that never survives to publication is not yet a strength.

[roadmap] The rest of this paper is structured as follows. Section 2 describes repertory grids and the FOCUS method. Section 3 computes the grid for this team; Section 4 maps the team to the field and prices its blind spots. Section 5 reports the three experiments behind RQ1 to RQ3. Section 6 discusses observations, threats to validity, and future work. Section 7 concludes with advice for teams that would repeat this exercise on their own corpus.

Figure 7: The left panel shows the introduction grammar, reverse engineered from four exemplars [8, 14, 16, 31] and Widom's audit questions [37]. The right panel shows the step-8 draft of paper[0] after roughly one hour of human rewriting (§5.3); its bold labels are the grammar's leaves, and its bold bracketed numbers are placeholders that step 9 must earn.

```
lit --> scheme, wings, ours,
      holes, issues, pivot.
% name the axes once, then sort every
% paper under them
scheme --> [axes].
% e.g. "question asked, result
% produced, validation offered
% (Shaw'03)"
% one wing = one crowded cell of the
% question x result grid; name who did
% what, where
wings --> [wing]+.
% e.g. "generalization questions
% with report results own the
% citation knee"
% place your own papers on the same
% map, in the same words
ours --> [where_we_sit].
% e.g. "the team's seeds sit in
% that report-and-experience cell"
% the map argues by its holes
holes --> [crowded], [empty].
% e.g. "reports of individual
% usage abound"
% e.g. "empirical models of teams
% are absent"
% each empty cell prices an open issue
issues --> [issue]+.
% e.g. "team-level empirical
% models of strength"
% claim what you will fix; defer the
% rest, with reasons
pivot --> [claimed], [deferred].
% e.g. "this paper addresses
% the first two"
```

[axes] Shaw sorts research papers by three axes [32]. The question asked may seek a method of development, a method of analysis, the design or evaluation of an instance, a generalization, or a feasibility check. The result produced may be a procedure, a qualitative model, an empirical model, an analytic model, a tool, a solution, or a report. The validation offered ranges over analysis, evaluation, experience, example, persuasion, and assertion. Two of the axes form a grid, and the third axis grades every cell. The reading set of Figure 3 sorts cleanly under that scheme.

[wing] The most-cited papers in this reading set are surveys. In Shaw's terms, they ask generalization questions and produce reports. Giray's review of deep defect prediction (JSS'22) and Zhang's mapping of critical-scenario testing (TSE'22) each code hundreds of papers so that their readers need not. Surveys summarize experiments run by others; hence this wing adds no new evidence of its own.

[wing] A second wing builds tools. In Shaw's terms, these papers ask design questions and validate by evaluation. Xu's in-IDE code generation study (TOSEM'22) and Amaral's compliance checker (TSE'22) each ship a tool and then measure it. Evaluation is the rarest validation in this reading set; only the tooling wing offers it.

[wing] The newest wing reports lived experience. Càmarà's report on generative AI in modeling (SoSyM'23) and Khojah's observational study of ChatGPT use (Proc. ACM'24) each watch individual practitioners at work. Experience reports are quick to write and quick to date; they record current practice rather than settled knowledge.

[where_we_sit] The team's three seed papers sit in that same experience wing (Di Nucci on accessibility, EMSE; Schmid on MLOps and Esposito on AI agents, IEEE Software). We already live in the crowded cell; hence the empty cells matter most to us.

[crowded] The map shows one crowded cell: reports of individual usage, growing every year.

[empty] Both empty cells concern teams rather than individuals. No paper in this set builds an empirical model of a team, and none validates a team-level claim by evaluation. An empty cell is an address for a next paper, not a criticism of the field. Based on this map, we see several issues with the current literature on trust, productivity, and human studies of AI assistants.

[issue] Team-level empirical models of strength are missing; every model in this set predicts for one developer at a time.

[issue] Measurement of assistant drift (assistants changing behavior between versions) is missing; the experience wing notices drift, yet nothing here measures it.

[issue] Longitudinal designs separating tool effect from team learning are missing; a cross-section cannot say who improved, the assistant or the team.

[claimed] Accordingly, the rest of this paper addresses the first two issues; Sections 3 and 4 build the team-level model, and Section 5 validates it at evaluation grade.

[deferred] The third issue needs years of observation, hence it is left to future work. We flag it now so that the record shows a deliberate choice.

Figure 8: The left panel shows the literature grammar; its scheme leaf is Shaw's question, result, and validation taxonomy [32], and its pivot leaf answers Widom's audit question of why the problem is not already solved [37]. The right panel shows the step-9 characterization for the same topic as Figure 7, after the same rewrite treatment (§5.3).

- elevator states the hypothesis as the conclusion, displayed as a two-to-three-line quote; papers carry at most one or two displayed claims, and the elevator spends one.
- move says what this paper does, then christens it; the method gets a name at first use and keeps it ever after.
- rqs answers every question on the spot, its headline number in bold, since an introduction holds no suspense.
- contribs is a short bullet list whose final bullet is always the replication URL.
- [caveat]* allows zero or more digressions, each answering an obvious objection before a reviewer raises it.
- [roadmap] closes the introduction with "the rest of this paper is structured as follows".

5.6 The literature review

Figure 8 proposes a structure for a literature review. The main move is respect, then disrespect. First the respect: describe clearly what prior work did, and what was good about it. Then the disrespect: explain, just as clearly, why that prior work is not good enough for the task at hand. The disrespect must name the mismatch precisely: wrong assumptions, wrong data, or wrong question. Vague complaints convince no reviewer. The review must end with a pivot of the following shape. "Based on the above, we offer the following as open and urgent issues in the research literature:" followed by a short list of those issues. "Accordingly, the rest of this paper does X, so that Y can do Z better." Everything downstream of the pivot must trace to an item on that list. A section that traces to nothing should be cut; otherwise, the list is incomplete.

PAUSE & REVIEW 5: THE INTRODUCTION

Ask: is the opening keen, and does it capture the interests of the writing team? Would it hold a technical audience, a generalist audience, or both? Would your own last paper parse under this grammar, and which leaf would fail first? Does your field replace any production (some venues ban the roadmap paragraph; this grammar ends with one)?

To change the outcome: edit the grammar here, then regenerate the sample of Figure 7 against the new productions.

PAUSE & REVIEW 6: THE LITERATURE REVIEW

Ask: does every wing name who did what, and how it was validated? Does the respect precede the disrespect, and is each mismatch named precisely? Does the pivot list trace to the empty cells, and does every later section trace to the pivot?

To change the outcome: edit the grammar of Figure 8, then regenerate the sample; stale wings usually mean the reading set, not the grammar, needs the rerun.

```

methods --> data, rig, baselines, measures,
           stats, hyperparams, [worked_example].
% what was collected, from where, and what
% survived each filter
data --> [source], [filters].
% e.g. "1,200 OpenAlex records; five most-
% cited papers per author, two windows"
% the pipeline, one checkable step per leaf
rig --> [step]+.
% e.g. "constructs agreeing above 90%
% merge into one grid row"
% the rivals; no rival, no claim
baselines --> [rival]+.
% e.g. "self-report, coded the same way"
% define every number before use
measures --> [metric]+.
% e.g. "match% = 1 - dist/max-dist"
% claim gates: repeats, named tests, thresholds
stats --> [tests], [thresholds].
% e.g. "20 repeats; Cliff's delta < 0.195
% and KS at 95% decide ties"
% knobs set before the run (chosen, not
% learned): value, why, and ablation range
hyperparams --> [settings], [justification],
                 [ablation].
% e.g. "merge threshold 90%; justified by
% prior grids; varied 80-95%"
% one example threads the section.
% [worked_example]: e.g. "one author's column,
% end to end"

```

[source] An LLM agent fetched each author's five most-cited papers, all-time and again for the last five years: [1,200] candidate records, [60] papers kept.

[filters] Records with no retrievable abstract were dropped ([n] of [60]); duplicates merged by DOI. The download rate is reported rather than hidden, since silent loss is selection bias.

[step] The agent coded every kept paper against the nine ICSE/27 areas.

[step] The agent then played triad interviewee: shown three authors, it named who stands apart and on what dimension.

[step] Constructs that agreed above 90% merged into one grid row; Section [5] varies that threshold.

[step] FOCUS sorting reordered rows and columns so that similar neighbors adjoin, reversing poles where that raised the match.

[rival] The rival method is self-report. Each author wrote five sentences that name their own strengths. The same coding scheme scored those sentences against the nine areas.

[metric] The match between two columns is one minus their city-block distance, divided by the maximum possible distance.

[tests] RQ answers are gated by bootstrap over paper subsamples ([1,000] resamples) plus an effect-size test.

[thresholds] A claim is stated only when the bootstrap interval excludes the baseline and the effect is at least small.

[settings] The pipeline has four hyperparameters: papers per author ([5]), time windows ([2]: all-time and five-year), the construct-merge threshold ([90%]), and the agent model with its temperature.

[justification] Budget justifies the first: five papers per author keeps agent coding under [one hour] per team. Prior grid studies justify the threshold: merges above [90%] agreement reproduced hand-built grids.

[ablation] Section [5] reruns the grid at merge thresholds from [80%] to [95%]; any conclusion that flips inside that range is reported as fragile.

[worked_example] One column, [Menzies], is followed end to end: fetched papers, coded areas, elicited constructs, merged rows, final scores.

Figure 9: The left panel shows the methods grammar, reverse engineered from the exemplars [8, 14, 16] and this team's recent preprints. The right panel shows the step-8 draft of the methods section of paper[0]; its bracketed numbers are placeholders that later steps must earn.

```

threats --> [construct], [internal],
             [external], [conclusion].
% four categories after Cook & Campbell (SE
% form: Wohlin et al.); each leaf names its
% sub-threats, then ends with the mitigation
% applied or the cost knowingly accepted.
% construct: right thing measured? (mono-
% operation, mono-method, hypothesis guessing)
% e.g. "citations proxy strength;
% unpublished skill is invisible"
% internal: pipeline made the result?
% (selection, instrumentation, maturation)
% e.g. "one rater (the agent) coded all"
% external: does it travel? (wrong people,
% wrong setting, wrong time)
% e.g. "six authors, one field, one venue"
% conclusion: evidence enough? (low power,
% fishing, unreliable measures)
% e.g. "six elements; wide intervals"

```

[construct] Citation counts proxy strength, and the proxy is imperfect: a skill that never reached print is invisible to the grid. We accept that cost; for the purpose of staffing a paper, a strength with no published record is not yet a strength.

[internal] A single rater, the LLM agent, coded every paper, so any rater bias travels the whole pipeline. Mitigation: the authors reviewed every merged construct, and [17 of 19] survived that review.

[external] External validity carries three standard threats: wrong people (selection interacts with treatment), wrong setting (toy environments), and wrong time (history interacts with treatment) [38]. This study risks all three: six authors, one field, one venue's area list; nothing here licenses claims about other teams. Mitigation: the replication package reruns the pipeline on any team's corpus in minutes, so the reader can check their own case.

[conclusion] Six elements make wide bootstrap intervals, and close match percentages ([83%] versus [61%]) should be read as trends. Mitigation: every headline number ships with its interval, and no claim rests on a difference that fits inside one.

Figure 10: The left panel shows the threats-to-validity grammar: four fixed leaves, one per validity question, each closing with its mitigation or its accepted cost. The right panel shows the step-8 draft of the threats section of paper[0]; its bracketed numbers are placeholders that later steps must earn.

5.7 The methods section

Figure 9 proposes a structure for a methods section. The duty of the section is replication: name the data, the pipeline, the rivals, the measures, and the statistical gates, each in units a stranger could check. Three habits recur in the exemplars. Firstly, an inventory arrives early: the tasks, the algorithms, and the datasets, each in a table whose caption says what the table is for. Secondly, one worked example threads the whole section; it is introduced once and reused everywhere. Thirdly, hyperparameters are selected and justified in the open. A hyperparameter is a value the experimenter

chooses before the run, rather than a value the run learns: a threshold, a budget, a model setting. Weak papers bury those choices; strong papers list every one, justify each by arithmetic, by budget, or by prior result, and vary the important ones in an ablation study. One exemplar justifies its labeling budget twice over: a probability argument for how few labels a search needs, and the dollar cost of a graduate student's annual LLM allowance [35].

PAUSE & REVIEW 7: THE METHODS

Ask: could a stranger rerun the study from this section alone? Is every hyperparameter listed with its value, its justification, and its ablation range? Does each rival get the same data and the same budget?

To change the outcome: edit the productions of Figure 9 and regenerate; a missing knob goes into the settings leaf first, since the justification and ablation leaves feed from it.

5.8 Threats to validity

Figure 10 proposes a structure for the threats section. The four fixed leaves are not ours. They are Cook and Campbell’s validity categories, as adapted for software engineering by Wohlin et al. [38]. Each category carries its own named sub-threats; the left panel lists the common ones. The four leaves ask four questions. Construct validity asks: are we measuring the right thing? Internal validity asks: could the pipeline itself produce the result? External validity asks: does the result travel beyond this study? Conclusion validity asks: is the evidence strong enough for each claim? A threat named without a response reads as an unfixed bug. Hence every leaf ends the same way: the mitigation applied, or the cost knowingly accepted. The instrument tables of the coding pipeline (agreement rates, threshold sensitivity, download rates) belong in this section, since they argue instrument validity.

PAUSE & REVIEW 8: THE THREATS

Ask: does every leaf end with a mitigation or an accepted cost? Are the sub-threats of each category considered, not only the category headline? Do the instrument tables (agreement rates, thresholds, download rates) appear here rather than in the methods?

To change the outcome: edit Figure 10; a threat without a response is a to-do item, not a paragraph.

5.9 The conclusion

A conclusion retells the introduction, now with receipts. The summary says what the paper has said, in more detail than the abstract: name the headline numbers from the charts, and attach each claim to the result that earned it. Nothing new appears here. If the conclusion and the introduction disagree, one of them is lying; fix it. Future work shows that the authors know what must follow next: the issues deferred by the literature review return here, sized and ordered. When the work has legs, future work may end with a call to action for the community.

PAUSE & REVIEW 9: THE CONCLUSION

Ask: does every claim in the summary point at the chart that earned it? Do the introduction and the conclusion agree? Does future work return the deferred issues of the literature review, sized and ordered?

To change the outcome: edit the introduction or the conclusion until neither lies; then regenerate the abstract, since the abstract carries the paper.

Prompt. You are Reviewer 2 for the ICSE’27 research track. Below are the track’s five review criteria, quoted from its call for papers, and a draft. Assess the draft under each criterion in turn, quoting the text that fails it. End with one of the track’s three outcomes: accept, major revision, or reject.

Output (on the drafts of Figures 7 and 8; abridged):

Novelty. The FOCUS christening claims a new instrument, yet offers no delta against existing repertory-grid tooling; the contribution may be assembly, not invention.

Rigor. Every headline number is a bracketed placeholder ([17 of 19], [83%]): the paper claims an evaluation it has not run, names no statistical test, and uses one rater.

Relevance. Matching people to tasks matters to every team; however, evidence from one six-person team, studying itself, bounds the claimed impact.

Verifiability and transparency. Six papers (Giray, Zhang, Xu, Amaral, Câmara, Khojah) carry no citations, so none can be verified; RQ2’s ground truth, “the team’s next accepted papers”, cannot exist at submission time. The promised replication package is the one strong point.

Presentation. Clear, and the grammar labels aid navigation; minor: the introduction defines the grid twice.

Outcome: reject, with encouragement to resubmit once the evaluation exists.

Figure 11: The reviewer-2 critic. The prompt embeds the five review criteria of the ICSE’27 research track (novelty, rigor, relevance, verifiability and transparency, presentation), quoted from its call for papers (conf.researchr.org/track/icse-2027/icse-2027-research-track); the output, abridged, assesses the step-8 drafts of Figures 7 and 8 under those same headings and ends with one of the track’s three outcomes.

6 GENERATING CRITICISM

Steps 10 to 12 of the loop of Section 1 turn from writing to attack. The first and most expensive critic is a human: step 10 is an off-line read of 30 to 60 minutes, social media off, and it cannot be delegated. The cheapest attacker, used at steps 5 and 11, is an LLM told to be hostile. Figure 11 shows the prompt: the model plays Reviewer 2 at the target venue, and the prompt embeds the venue’s own review criteria. For ICSE’27 those criteria are five: novelty, rigor, relevance, verifiability and transparency, and presentation; the track’s outcomes are accept, major revision, and reject. Structuring the critique under the venue’s headings matters, since those headings are exactly the form in which the real rejection would arrive. Reviewer 2 is one of four critics the loop assembles; the others compare the draft against the venue’s award winners, the ACM empirical standards, and the ten nearest prior papers. The same figure shows that critic running on the generated drafts of Figures 7 and 8; each finding quotes the text it fails.

The review is half the service; triage is the other half. Figure 12 shows the follow-up prompt. It divides the review’s findings into part 1, fixes the model can apply alone, and part 2, issues that need a human decision. The split prices the remaining work. Clarifications, reorderings, prunings, and citation lookups are cheap; experiments, ground-truth choices, and scope decisions are not. The

Prompt. Here is a reviewer's report on the draft. Whatever its outcome (reject or major revision), divide its findings into part 1, fixes you can apply yourself (clarification, reordering, pruning, citation lookup), and part 2, issues that need a human decision (experiments, ground truth, scope). Apply nothing; output the two lists.

Output (on the review of Figure 11; abridged):

Part 1 (the model applies): find and verify citations for the six named papers; prune the duplicated grid definition; reorder the RQ2 answer so evidence precedes claim; use one term, self-report, throughout.

Part 2 (the humans decide): run the experiments that earn the bracketed numbers; pick a defensible ground truth for RQ2; recruit an independent rater for the constructs; add more teams, or shrink every generalization claim.

Figure 12: The triage prompt splits the review of Figure 11: part 1 is applied at step 11 of the loop; part 2 goes to the humans at step 12.

loop applies part 1 automatically at step 11 and queues part 2 for the humans at step 12.

Two steps then close the loop. At step 13, the fixes are applied, and one automatic check guards the paper's front door: the title and abstract are recoded with the same flags as the reading set, and they must match the body's coding with zero flips. At step 14, a failed self-test returns the loop to step 9; otherwise the paper ships, with its replication package.

7 IN SUMMARY

This paper asked how a research team should spend its scarce writing hours. Newell supplied the answer. Know what you are good at, then aim those strengths at your field's open problems. Section 1 turned that advice into a loop of fourteen small steps. Sections 2 to 4 ran the loop's reading passes on this paper's own authors. Those sections computed a repertory grid from our corpus (Figure 1), mapped the grid to the nine ICSE'27 areas, and reviewed the literature of the territory that map nominated. Section 5 gave the writing rules, namely a grammar for each part of a paper, a worked sample beside each grammar, and the rewrite discipline that turns raw LLM output into prose. Section 6 closed the loop with critics: a reviewer-2 prompt built from the venue's own five review criteria, and a triage prompt that splits the review into machine fixes and human decisions.

We stress that none of this is the definitive way to write a paper. It is one way, and its parts are deliberately easy to change. Every structure proposed here is a small grammar in a figure. To disagree with one, edit its productions and ask an LLM to regenerate the sample; the cost of the experiment is minutes. Your venue may open with related work, ban the roadmap paragraph, or demand a structured abstract. Adjust the grammar, not your patience.

Hence our call to action. Future work should assemble a catalog of paper types, each with its own grammar: the empirical study, the systematic review, the tool paper, the experience report, the vision paper. Communities curate datasets and baselines; they could curate paper shapes the same way, so that every writer starts from a

tested structure rather than a blank page. Schrödinger asked scientists to tell everyone what they have been doing. A shared catalog of shapes would make that telling cheaper, and perhaps better.

REFERENCES

- [1] Andrea Arcuri and Lionel Briand. 2011. A Practical Guide for Using Statistical Tests to Assess Randomized Algorithms in Software Engineering. In *Proceedings of the 33rd International Conference on Software Engineering (ICSE)*. 1–10. <https://doi.org/10.1145/1985793.1985795>
- [2] ASD. 2025. ASD-STE100 Simplified Technical English, Issue 9: Standard for Technical Documentation. Aerospace, Security and Defence Industries Association of Europe, Brussels; <https://www.asd-ste100.org>.
- [3] Sebastian Baltes, Marc Cheong, and Christoph Treude. 2026. "An Endless Stream of AI Slop": How Developers Discuss the Burden of AI-Assisted Software Development. *arXiv:2603.27249*.
- [4] Norman Cliff. 1993. Dominance Statistics: Ordinal Analyses to Answer Ordinal Questions. *Psychological Bulletin* 114, 3 (1993), 494–509. <https://doi.org/10.1037/0033-2909.114.3.494>
- [5] Jacob Cohen. 1988. *Statistical Power Analysis for the Behavioral Sciences* (2nd ed.). Lawrence Erlbaum.
- [6] Jacob Cohen. 1994. The Earth is Round ($p < .05$). *American Psychologist* 49, 12 (1994), 997–1003. <https://doi.org/10.1037/0003-066X.49.12.997>
- [7] Joyce Ehrlinger and David Dunning. 2003. How Chronic Self-Views Influence (and Potentially Mislead) Estimates of Performance. *Journal of Personality and Social Psychology* 84, 1 (2003), 5–17. <https://doi.org/10.1037/0022-3514.84.1.5>
- [8] Wei Fu and Tim Menzies. 2017. Easy over Hard: A Case Study on Deep Learning. In *Proceedings of the 11th Joint Meeting on Foundations of Software Engineering (ESEC/FSE)*. 49–60. <https://doi.org/10.1145/3106237.3106256>
- [9] Andrew Gelman and Eric Loken. 2014. The Statistical Crisis in Science. *American Scientist* 102, 6 (2014), 460–465. <https://doi.org/10.1511/2014.111.460>
- [10] John P. A. Ioannidis. 2005. Why Most Published Research Findings Are False. *PLoS Medicine* 2, 8 (2005), e124. <https://doi.org/10.1371/journal.pmed.0020124>
- [11] Vigdis By Kampenes, Tore Dybå, Jo E. Hannay, and Dag I. K. Sjøberg. 2007. A Systematic Review of Effect Size in Software Engineering Experiments. *Information and Software Technology* 49, 11–12 (2007), 1073–1086. <https://doi.org/10.1016/j.infsof.2007.02.015>
- [12] George A. Kelly. 1955. *The Psychology of Personal Constructs*. W.W. Norton. <https://archive.org/details/psychologyofpers0001kell>.
- [13] Dmitry Kobak, Rita González-Márquez, Emőke-Ágnes Horvát, and Jan Lause. 2024. Delving into LLM-Assisted Writing in Biomedical Publications through Excess Vocabulary. *arXiv:2406.07016*.
- [14] Ekrem Kocaguneli, Tim Menzies, Ayse Bener, and Jacky W. Keung. 2012. Exploiting the Essential Assumptions of Analogy-Based Effort Estimation. *IEEE Transactions on Software Engineering* 38, 2 (2012), 425–438. <https://doi.org/10.1109/TSE.2011.27>
- [15] Justin Kruger and David Dunning. 1999. Unskilled and Unaware of It: How Difficulties in Recognizing One's Own Incompetence Lead to Inflated Self-Assessments. *Journal of Personality and Social Psychology* 77, 6 (1999), 1121–1134. <https://doi.org/10.1037/0022-3514.77.6.1121>
- [16] Piergiorgio Ladisa, Henrik Plate, Matias Martinez, and Olivier Barais. 2023. SoK: Taxonomy of Attacks on Open-Source Software Supply Chains. In *IEEE Symposium on Security and Privacy (SP)*. 1509–1526. <https://doi.org/10.1109/SP46215.2023.10179304>
- [17] Zhongtang Luo. 2025. ICLR Points: How Many ICLR Publications Is One Paper in Each Area? *arXiv:2503.16623*.
- [18] Frank J. Massey. 1951. The Kolmogorov-Smirnov Test for Goodness of Fit. *J. Amer. Statist. Assoc.* 46, 253 (1951), 68–78. <https://doi.org/10.1080/01621459.1951.10500769>
- [19] Roger C. Mayer, James H. Davis, and F. David Schoorman. 1995. An Integrative Model of Organizational Trust. *Academy of Management Review* 20, 3 (1995), 709–734. <https://doi.org/10.5465/amr.1995.9508080335>
- [20] Tim Menzies, Paris Avgeriou, Robert Feldt, Mauro Pezzè, Abhik Roychoudhury, Mirosław Staron, Sebastian Uchitel, and Thomas Zimmermann. 2026. SE Journals in 2036: Looking Back at the Future We Need to Have. In *IEEE/ACM 48th International Conference on Software Engineering: Future of Software Engineering (ICSE-FoSE '26)*. ACM. <https://doi.org/10.1145/3793657.3793874> *arXiv:2601.19217*.
- [21] Merriam-Webster. 2025. Word of the Year 2025: Slop. <https://www.merriam-webster.com/wordplay/word-of-the-year>.
- [22] André N. Meyer, Thomas Fritz, Gail C. Murphy, and Thomas Zimmermann. 2014. Software Developers' Perceptions of Productivity. In *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering (FSE)*. 19–29. <https://doi.org/10.1145/2635868.2635892>
- [23] Susan Milligan. 2025. Embrace What You Don't Know: The Power of Defiance to Expertise. NAHAM Connections. <https://www.naham.org/page/ConnectionsEmbraceWhatYouDontKnowThePowerofDeferencetoExpertise>,

- accessed August 2026.
- [24] Allen Newell. 1991. Desires and Diversions. Distinguished Lecture, School of Computer Science, Carnegie Mellon University. Video: https://www.youtube.com/watch?v=_sD42h9d1pk.
 - [25] Open Science Collaboration. 2015. Estimating the Reproducibility of Psychological Science. *Science* 349, 6251 (2015), aac4716. <https://doi.org/10.1126/science.aac4716>
 - [26] Delroy L. Paulhus. 1984. Two-Component Models of Socially Desirable Responding. *Journal of Personality and Social Psychology* 46, 3 (1984), 598–609. <https://doi.org/10.1037/0022-3514.46.3.598>
 - [27] Jeanine Romano, Jeffrey D. Kromrey, Jesse Coraggio, and Jeff Skowronek. 2006. Appropriate Statistics for Ordinal Level Data: Should We Really Be Using t-test and Cohen’s d for Evaluating Group Differences on the NSSE and Other Surveys?. In *Annual Meeting of the Florida Association of Institutional Research*. 1–33.
 - [28] Shlomo S. Sawilowsky. 2009. New Effect Size Rules of Thumb. *Journal of Modern Applied Statistical Methods* 8, 2 (2009), 597–599. <https://doi.org/10.22237/jmasm/1257035100>
 - [29] Erwin Schrödinger. 1951. *Science and Humanism: Physics in Our Time*. Cambridge University Press. The quotation closes the first of four public lectures given for the Dublin Institute for Advanced Studies at University College Dublin, February 1950; <https://archive.org/details/sciencehumanism0000erwi>.
 - [30] Chantal Shaib, Tuhin Chakrabarty, Diego Garcia-Olano, and Byron C. Wallace. 2025. Measuring AI “Slop” in Text. *arXiv:2509.19163*.
 - [31] Claude E. Shannon. 1940. *A Symbolic Analysis of Relay and Switching Circuits*. Master’s thesis. Massachusetts Institute of Technology. <https://hdl.handle.net/1721.1/11173>.
 - [32] Mary Shaw. 2003. Writing Good Software Engineering Research Papers. In *Proceedings of the 25th International Conference on Software Engineering (ICSE 2003)*. IEEE, 726–736. <https://doi.org/10.1109/ICSE.2003.1201262>
 - [33] Mary Shaw. 2026. Correctness, Confidence, and Context: Framing Software Assurance in the AI Age. Keynote, ACM Intl. Conf. on the Foundations of Software Engineering (FSE ’26); *arXiv:2607.04667*.
 - [34] Mildred L. G. Shaw and Laurie F. Thomas. 1978. FOCUS on Education: An Interactive Computer System for the Development and Analysis of Repertory Grids. *International Journal of Man-Machine Studies* 10, 2 (1978), 139–173. [https://doi.org/10.1016/s0020-7373\(78\)80009-1](https://doi.org/10.1016/s0020-7373(78)80009-1)
 - [35] Srinath Srinivasan and Tim Menzies. 2026. Better Together, in the Right Order: Classical-then-LLM Optimization for SE. *arXiv:2607.02583*.
 - [36] William H. Walters and Esther Isabelle Wilder. 2023. Fabrication and Errors in the Bibliographic Citations Generated by ChatGPT. *Scientific Reports* 13 (2023), 14045. <https://doi.org/10.1038/s41598-023-41032-5>
 - [37] Jennifer Widom. 2006. Tips for Writing Technical Papers. Stanford University. <https://cs.stanford.edu/people/widom/paper-writing.html>, accessed July 2026.
 - [38] Claes Wohlin, Per Runeson, Martin Höst, Magnus C. Ohlsson, Björn Regnell, and Anders Wesslén. 2012. *Experimentation in Software Engineering*. Springer. <https://doi.org/10.1007/978-3-642-29044-2>

APPENDIX

A SETTING UP TO WRITE TOGETHER

The advice of this paper assumes a team, and teams arrive with unequal tooling comfort. Some authors live in the terminal; others have never typed a git command and never should. Forcing one tool on everyone loses authors at both ends; hence we offer a ladder, and each author picks a rung:

- (1) a reader, who watches the built PDF and mails comments;
- (2) a browser writer, who drafts in a shared Google Doc with help from claude.ai;
- (3) an Overleaf writer, who edits the LaTeX live in the browser and watches the PDF re-render;
- (4) a git writer, who clones the repository, edits, and pushes;
- (5) an agent runner, who works with Claude Code (or similar) on their own machine, with the agent reading and writing the shared source directly.

Every rung is a full member of the team; the rungs differ only in setup cost.

One piece of plumbing joins the ends of that ladder. An Overleaf project is also a git remote, so the same paper has two faces: a live

browser editor for humans and an ordinary repository for people and agents (Figure 13).

Figure 13 shows that spectrum as four faces of one paper. At near-zero setup cost, an author works entirely in the browser. Panel A is an AI assistant (Gemini in the screenshot; claude.ai serves as well) and panel B a shared document. Text snippets are cut and pasted between the two. For authors at home with command-line git tools, panels C and D show the repository route. Each author works on their own machine with an agent such as Claude Code. All authors share one joint writing space in Overleaf, through the git bridge described below. We stress that the repository route is optional; the browser route is a perfectly functional way to co-author a paper. The extra setup of C and D pays off as the team grows and the edits get larger.

GitHub holds the source of truth; a push moves edits to Overleaf in seconds; a pull collects whatever the browser writers typed. Hence n people and n agents can write asynchronously, meeting only at the merge.

Merges stay small because of two source-layout habits:

- every sentence ends its line, so git diffs and merges by the sentence, and two edits collide only when they touch the same sentence;
- prose lives in many small files (one per section) beneath a thin root of include statements, so writers claim a section, not the paper.

Review comments follow the same philosophy: a comment is a macro in the source (who said it, what they said), listed by one make target and resolved by deleting it. Margin comments in web editors were rejected since they do not survive the trip through git.

The last ritual is scheduling. Before any shared writing session, one owner verifies that every participant can already write to something: doc links delivered, project invites accepted, push access proven by one trivial commit. Access problems solved during a session cost n people an hour; solved the day before, they cost one person a minute. The prompts the team uses are versioned in the same repository as the prose, so every rung (browser paste or agent invocation) runs the same words. (Aside: this paper was written with exactly this setup, one author plus one agent, trading pushes through the bridge described above.)

Table 2 makes the ladder concrete for this paper’s own repository: what each co-author does once, and what they do daily.

PAUSE & REVIEW 10: THE TEAM’S TOOLING

Ask: can every team member read and write the paper’s files? Is each member using a technology where they feel productive and comfortable? Is the team accommodating everyone, even where system skills differ widely?

To change the outcome: move people to a different rung of Table 2 (no rung outranks another), issue the missing invites or tokens, and name a sync captain to carry edits between the faces of Figure 13.

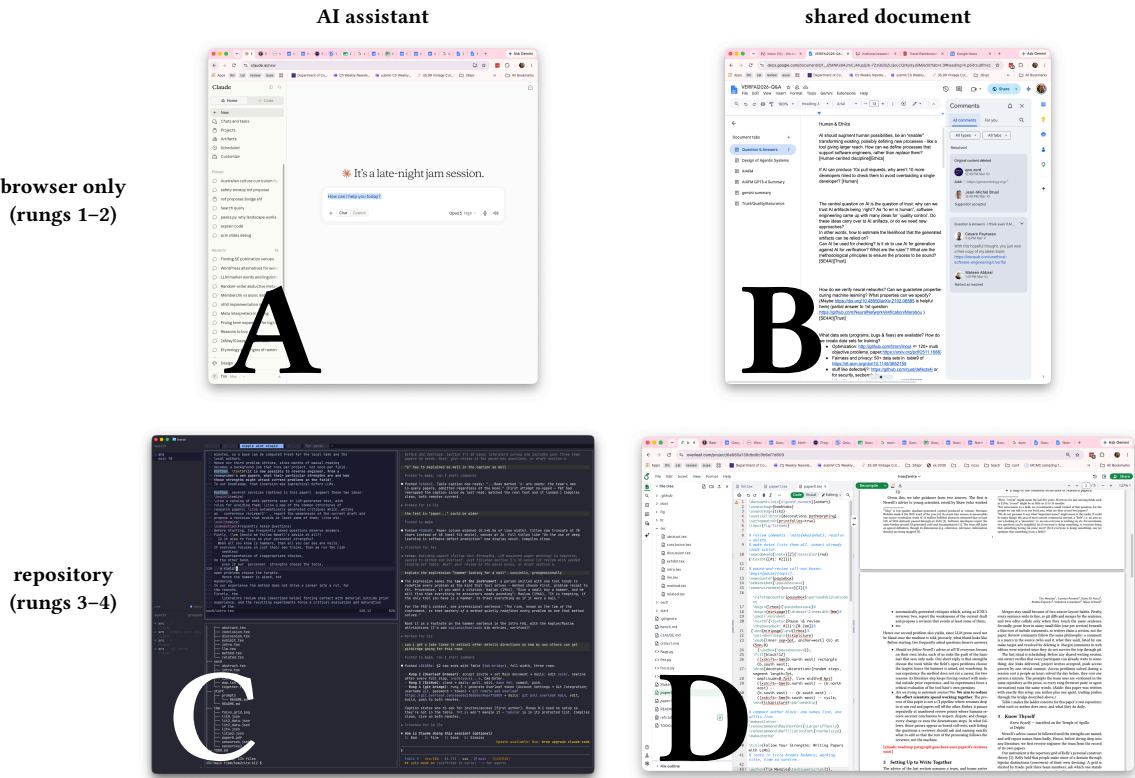


Figure 13: There are many ways for teams to use the advice of this paper to collaboratively write papers. Columns: AI assistants and shared documents. Rows: the browser-only world of rungs 1-2 (A: Gemini, B: Google Docs) and the repository world of rungs 3-4 (C: Claude Code working the git bridge, D: Overleaf rendering the same paper).

Table 2: Getting started on this paper, by rung. One-time setup, then the daily cycle. Rung 2 needs an Overleaf project invite from the first author; rungs 3-4 need GitHub push access from the same. Every rung that touches the LaTeX follows two habits: start a new line after every full stop (git then merges by the sentence), and keep prose in many small files, one per section, joined by `\input` under a thin root.

rung	one-time setup	daily cycle
2: Overleaf browser	accept the project invite; open the project; set Menu → Settings → Main document to the paper being built	edit prose in <code>sec0/</code> ; new line after every full stop; comment via <code>\note{you}{...}</code> ; <code>Cmd-Enter</code> recompiles
3: GitHub	<code>git clone github.com/timm/how2rite</code>	pull; edit <code>sec0/</code> ; make <code>fmt</code> ; commit; push
4: the git bridge	rung 3, plus: in Overleaf, Account Settings → Git Integration → Generate token; you are looking for this integration:  Git clone this project. (username is git, password is the token); then <code>git remote add overleaf https://git.overleaf.com/6a665a138dbd6c9b6ef7d809</code>	<code>git pull overleaf main</code> (collect browser edits); edit; make <code>fmt</code> ; build; push to both: <code>git push origin main</code> ; <code>git push overleaf main:main</code>

B A TUTORIAL ON RANKING STATISTICS

B.1 Why Argue with Distributions?

Software engineering practices are highly variable: different people use different tools on different platforms in pursuit of different goals. Reasoning about software engineering therefore means studying not just average effects but also the variability around those effects. That variability has many sources. Project data is

noisy, since no two projects share the same staff, toolchain, platform, or objectives. Some of the “data” is really guesswork; e.g. when human experts estimate the value of a tool or an industrial process. Modeling adds more variance: learners shuffle their data, optimizers roll dice, and large language models sample their tokens. The same data can hence yield many different models, and a useful conclusion must hold across all those models, not just one.

A single result is therefore not a fact but a lottery ticket, and honest comparisons must argue with **distributions**, not points. Two treatments whose result clouds overlap heavily are, for practical purposes, the same treatment, no matter which one “won” on any single run.

The record shows the cost of skipping this step. Cohen spent a career documenting how significance testing, uncoupled from effect size, filled psychology with findings too small to matter [5, 6]. Decades later, a mass replication of one hundred published psychology studies confirmed barely more than a third of them [25]. Ioannidis argued that under low power and flexible analysis, most published findings are simply false [10]; Gelman and Loken traced the mechanism to the “garden of forking paths,” where innocent analytic choices manufacture significance [9]. Software engineering shows the same problem: a systematic review found that most SE experiments did not report effect sizes at all, so readers could not tell large results from small ones [11]. Such mistakes are costly; e.g. months of compute and writing can be spent defending a one-percent improvement that no user of the system would notice.

The fix is cheap. Arcuri and Briand’s practical guide for randomized algorithms in SE offers much relevant advice; central to that advice is to prefer non-parametric tests and to pair every significance test with an effect size [1]. This appendix implements our version of that advice, then extends it from comparing two treatments to ranking many, all in 44 lines of Python.

That last paragraph used three technical terms:

- a **significance test** asks whether a difference could plausibly be chance alone;
- an **effect size** measures how big a difference is;
- **non-parametric** methods rank or count, and assume no particular distribution shape.¹¹

Throughout this appendix, statistical terms appear in **bold** at their first use (and are defined in Table 3).

B.2 A Worked Example

To make this concrete, suppose three teams race to deliver a first minimal viable product, each using a different agent framework (*a*, *b*, *c*). Each framework is tried twenty times; the outcome is the whole number of days until that first MVP ships:

```
a: 1 2 2 2 3 3 3 3 4 4 4 4 5 5 5 5 6 6 7 8
b: 5 6 6 7 7 7 8 8 8 8 9 9 9 9 10 10 10 11 11
c: 5 6 6 7 7 8 8 8 8 9 9 9 9 10 10 10 11 11 11
```

Do these **samples** actually say different things? Figure 14 shows two views of the data. The left-hand plot shows each sample’s **probability density function** (pdf); i.e. how the values are distributed (so, e.g., *a*’s most common value is three days). The right-hand plot shows what fraction of the projects finished within a given number of days; this is called a **cumulative distribution function** (cdf). Intuitively, the answer seems clear: *b* and *c* pile up around eight or nine days and overlap almost everywhere, while *a* sits several days to their left. The eye says *a* is different and *b*, *c* are the same.

An eyeball, however, is not an argument. Reviewers will ask how much overlap counts as “almost everywhere,” and whether

Table 3: Statistical jargon used in this appendix.

term	meaning
sample	The numbers actually collected; e.g. twenty repeated runs of one treatment. Samples stand in for the infinite population of runs that could have happened.
distribution	The shape of a sample: where its values pile up and how far they spread.
mean, median	Two centers of a distribution: the average, and the middle value after sorting. Medians resist outliers, so they suit skewed data.
variance, standard deviation	How far values scatter about the mean; the deviation is the square root of the variance, in the same units as the data. A <i>pooled</i> deviation blends the spread of two samples, weighting each by its size.
pdf	The probability density function: for any value <i>v</i> , the fraction of the sample at <i>v</i> ; i.e. the shape of the data.
cdf	The cumulative distribution function: for any value <i>v</i> , the fraction of the sample at or below <i>v</i> . A sorted list gives the cdf directly.
significance test	A check of whether a difference this large could plausibly arise by chance alone, given some tolerated false-alarm rate (often five percent).
critical value	The tabled cutoff a test statistic must exceed before a result counts as significant at a given false-alarm rate (often five percent).
effect size	How big a difference is, in units that survive translation between studies; e.g. “half a standard deviation,” not “0.03 on our benchmark.”
parametric, non-parametric rank	Parametric tests assume a distribution shape (usually the bell curve); non-parametric tests rank or count instead, so they survive skew, ties, and outliers. A position after sorting. Statistics built on ranks use order, not magnitude, which makes them robust to skew, ties, and outliers.

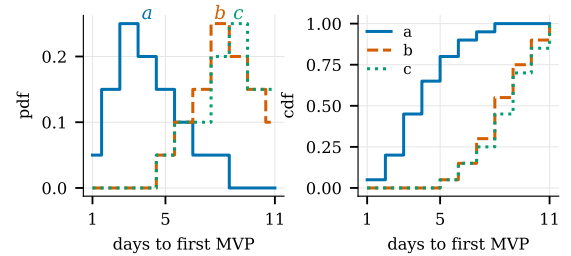


Figure 14: Twenty runs per framework: days to first MVP, as probability density functions (left) and cumulative distribution functions (right). Frameworks *b* and *c* overlap almost everywhere; *a* is several days faster. Regenerate with `make statsfig`.

four days is a large gap or a small one. The science of statistics is how to make defensible conclusions from distributions like those in Figure 14. For an example of that kind of conclusion, see the rest of this appendix.

¹¹As opposed to *parametric* tests, which assume all values fit some idealized, but possibly unrealistic, curve such as the normal Gaussian.

B.3 Two Questions, One Verdict

Statistical comparisons answer two different questions. A significance test asks: could a gap this large be chance? An effect size asks: is the gap big enough to matter? The two can disagree: given enough runs, even a microscopic difference passes a significance test, while with too few runs, even a large difference can be luck. One check guards against noise, the other against triviality; hence standard advice runs both [1].

Our code adds one more safeguard. Effect sizes come in parametric and non-parametric flavors, and each can be fooled in its own way, so both flavors are run. Two samples are called “the same” only when all three tests agree that the difference is small: two effect sizes (cohen, cliffs) and one significance test (ks), each defined in the sections that follow. Python’s and is lazy, so the cheapest test runs first and the costlier ones run only when needed.

Every test here expects sorted input; that one convention buys linear scans, percentile tricks, and free medians throughout. By default same trusts its caller to pre-sort (as ranks does below); pass `is_sorted=False` and it sorts defensively first.

```
def same(xs, ys,
        c0=.35, # cohen threshold; from [5, 28]
        d0=.195, # cliffs threshold; from [27]
        k0=1.36, # ks threshold; from [18]
        is_sorted=True):
    if not is_sorted:
        xs, ys = sorted(xs), sorted(ys)
    return (cohen(xs, ys) <= c0 and
            cliffs(xs, ys) <= d0 and
            ks(xs, ys) <= k0)
```

On the worked example of §B.2, all three tests agree:

- Comparing *b* to *c* yields $d=0.13$, $\delta=0.10$, and $ks=0.32$: all under their thresholds, so `same` returns true.
- Comparing *a* to *b* yields $d=2.2$, $\delta=0.91$, and $ks=2.4$: all far over, so `same` returns false.

Taken together, those verdicts divide the three frameworks into two groups:

((a), (b), (c))

The eyeball intuition is now a defensible claim.

For code that reports these groupings, see §B.6.

B.4 Significance: the Kolmogorov–Smirnov Test

Means can agree while distributions disagree; e.g. two samples can share an average while one is tight and the other is spread wide. The Kolmogorov–Smirnov statistic guards against this by comparing whole cumulative distributions [18]. The intuition runs as follows. If two samples come from the same source, their cdfs should ride on top of each other: at every value, about the same fraction of each sample lies below. Anywhere the curves pull far apart, one treatment has finished far more of its runs than the other, so the distributions must differ.

The KS statistic reports the largest vertical gap between cdfs. For example, Figure 15 repeats the cdfs of Figure 14, now with that gap drawn in (so the KS statistic is the tallest pole that fits between two

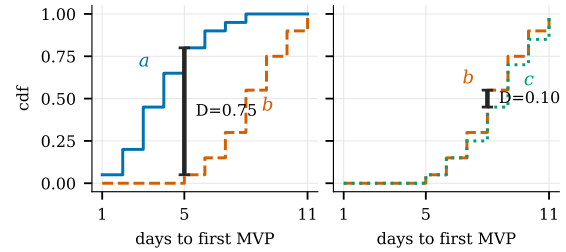


Figure 15: The KS statistic is the largest vertical gap between two cdfs (the black pole). Left: *a* versus *b*. Right: *b* versus *c*. Regenerate with `make statsfig`.

cdfs). Between *a* and *b* the pole is tall ($D=0.75$); between *b* and *c* no pole higher than $D=0.10$ fits anywhere.

The code walks both sorted lists one distinct value at a time, advancing two counters *p* and *q* past ties, so the whole pass is linear. After each advance, *p*/*nx* and *q*/*ny* are the two cdfs sampled at the same value *v*. Dividing the raw gap by $\sqrt{(n_x + n_y)/(n_x n_y)}$ rescales it into critical units; at the usual five-percent false-alarm rate, the **critical value** on that scale is the tabled constant 1.36 [18], so results under 1.36 are insignificant.

```
def ks(xs, ys):
    nx, ny = len(xs), len(ys)
    d = p = q = 0
    while p < nx and q < ny:
        v = min(xs[p], ys[q])
        while p < nx and xs[p] == v: p += 1
        while q < ny and ys[q] == v: q += 1
        d = max(d, abs(p/nx - q/ny))
    return d / ((nx + ny) / (nx * ny)) ** 0.5
```

B.5 Effect Size

Two effect sizes are used here: one non-parametric (Cliff’s δ) and one parametric (Cohen’s *d*).

B.5.1 Rank Imbalance: Cliff’s δ . Cliff’s δ is a non-parametric effect size [4]. It asks: if one item is drawn from each sample, how often does *x* beat *y*, minus how often *y* beats *x*? Naively that is a quadratic all-pairs count. Sorting first makes it linear: as *x* ascends, the counts *j* and *k* (how many *ys* sit below, and at-or-below, *x*) only ever move forward. Romano et al. rate $\delta < 0.147$ as negligible and $\delta < 0.33$ as small [27]; our threshold of 0.195 splits that range, a slightly strict “smallish.”

```
def cliffs(xs, ys):
    gt = lt = j = k = 0
    for x in xs:
        while j < len(ys) and ys[j] < x: j += 1; k = j
        while k < len(ys) and ys[k] == x: k += 1
        gt += j; lt += len(ys) - k
    return abs(gt - lt) / (len(xs) * len(ys))
```

B.5.2 Pragmatic Dullness: Cohen’s *d*. Sooner or later, a business user reviewing statistical results will say: “that difference may be real, but it is dull; I only care about deltas bigger than *x*.” The

complaint bites hardest when results are printed rounded: a difference can pass both tests above, yet be smaller than the last digit shown. Hence one more test is needed, one that can call a difference real but pragmatically insignificant. Cohen’s d is the standard test for dull: the gap between two means, in units of the pooled **standard deviation** [5]. Cohen rated $d=0.2$ as small and $d=0.5$ as medium [5] (Sawilowsky later refined that scale [28]); our threshold of 0.35 is the border between the two, so gaps under about a third of a standard deviation count as dull. If no such pragmatic argument is needed, set `c0` to infinity in `same` and `cohen` loses its vote.

The pooled deviation weights each sample’s **variance** by its degrees of freedom, so unequal sample sizes are handled fairly. A `TINY` constant guards the division when both samples have near-zero spread. The code needs no imports: on sorted data, the gap between the 10th and 90th percentile spans 2.56 standard deviations of a normal curve, so one subtraction estimates the deviation.

```
TINY = 1e-32

def mean(a): return sum(a) / len(a)
def stdev(a): n = len(a)//10; return (a[9*n]-a[n])/2.56

def cohen(xs, ys):
    n, m = len(xs), len(ys)
    sd = (((n-1)*stdev(xs)**2 +
            (m-1)*stdev(ys)**2) / (n+m-2)) ** 0.5
    return abs(mean(xs) - mean(ys)) / (sd + TINY)
```

B.6 Ranking Many Treatments

Real studies compare many treatments, not two. Running all pairwise tests invites false alarms, so instead the treatments are sorted by their **medians**, then swept once from best to worst. Each treatment is compared to the best member of the current **rank**; when same says they are indistinguishable, the rank is shared, otherwise a new rank opens. Rank zero holds the winners; the full table maps every treatment to its rank. The flag `big` flips the sort for measures where larger is better.

```
def ranks(d, big=False):
    mid = lambda t: t[len(t) // 2]
    dd = {k: sorted(v) for k, v in d.items()}
    out, win, rank, best = {}, [], -1, None
    for k in sorted(dd, key=lambda k: mid(dd[k]), reverse=big):
        if best is None or not same(dd[best], dd[k]):
            rank, best = rank + 1, k
        if rank == 0: win.append(k)
        out[k] = rank
    return win, out
```

Applied to the worked example (where fewer days is better, so `big` stays false), `ranks` sorts the frameworks by their medians (4, 8, 9 days), gives `a` rank zero, then finds `b` and `c` indistinguishable, so they share rank one. That is, this data forms two groups:

((a) (b c))

and the winners list is just `[a]`.

Note the single sort at the top of `ranks`: every later test then runs on pre-sorted lists, so the whole ranking costs one sort per treatment plus one linear sweep.

Table 4: Ten treatments (rx1..rx10) on ten data sets (db1..db10). Cells show the median of twenty runs (fewer is better); grey cells sit at the top rank of their data set, as computed by ranks; the bottom row counts top-rank appearances. Summary: rx7 wins over everything else, but see db2 and db5 for counterexamples. Regenerate with `make statsfig`.

	rx1	rx2	rx3	rx4	rx5	rx6	rx7	rx8	rx9	rx10
db1	8.2	9.7	11.1	12.0	14.5	8.0	5.3	10.9	12.7	14.1
db2	8.4	3.1	11.0	11.9	14.3	8.5	12.4	11.3	12.5	14.1
db3	8.7	9.7	10.9	12.4	13.8	7.9	5.5	11.3	12.5	13.9
db4	8.2	9.6	11.0	12.7	14.0	8.0	5.4	11.1	12.6	14.2
db5	8.3	9.2	11.3	12.7	3.2	8.1	12.5	11.2	12.5	14.0
db6	8.1	9.7	10.9	12.5	14.2	8.4	5.1	10.9	12.1	14.3
db7	8.2	9.7	11.2	12.7	14.3	8.3	5.2	10.5	12.7	14.4
db8	8.4	9.5	11.3	12.7	13.7	7.8	4.7	10.6	12.2	14.1
db9	8.1	9.5	10.7	12.5	13.7	8.2	5.1	11.2	12.5	14.0
db10	8.5	10.0	11.1	12.5	13.9	8.5	5.2	10.9	12.9	14.0
wins	0	1	0	0	1	0	8	0	0	0

Results sections often end with a chart like Table 4: here, ten treatments are compared on ten data sets, twenty runs per cell. Grey marks the top-ranked treatments of each data set; the bottom row counts how often each treatment reaches the top rank. Such a chart supports a one-sentence summary: `rx7` wins over everything else (but see `db2` and `db5` for counterexamples). The result is a statistics stack small enough to audit in an afternoon, yet defensible enough for a results section.