

Peer Review

Review of: "Challenges and Paths Towards AI for Software Engineering"

Evgenii Arsentev¹

1. Independent researcher

1. What the work claims

This is a position paper with a large survey attached to it. The authors argue that AI for software engineering has advanced far enough to be a genuine success story of generative AI, and that the field is nonetheless stuck well short of the automation level it could reach. They make three moves. First (Sec. 2), a taxonomy of tasks: code generation, code transformation (refactoring, migration and translation, optimization), testing and program analysis, maintenance (documentation, PR review, code understanding), scaffolding and meta-code, and formal verification. Each task is meant to be placed along three measures defined at the top of Sec. 2: **scope** (function-level, self-contained unit, project-level), **logical complexity** (low, medium, high), and **level of human intervention** (low, medium, high, distilled from the six-level autonomy scale of [25] per footnote 1). Second (Sec. 3), a list of cross-cutting challenges: evaluation and benchmarks, tool usage, human-AI collaboration, long-horizon planning, large scope and long contexts, semantic understanding of codebases, low-resource languages, library and API version updates, and high logical complexity with out-of-distribution domains. Third (Sec. 4), research directions grouped into data collection, training, and inference-time approaches, with Figure 1 promising a mapping from directions back to challenges.

Appendix A then runs a per-task literature survey that is, by volume, the densest part of the paper: the reference list on Qeios runs to 356 entries. Sec. 5 lists four limitations in the authors' own words, including that the proposed directions "lack strong and concrete evidence" and that the paper is "written by people primarily in the academic community, who may not know the details of cutting-edge methods employed in frontier industry labs." Sec. 6 calls the work a position paper, while the Qeios header tags it REVIEW ARTICLE.

I want to say up front what I think this paper gets right, because the rest of my review is a list of things to repair. The insistence in Sec. 2 that software engineering is much wider than code generation is correct and worth repeating. The construct-validity paragraph in Sec. 3.1 names the real problem with SWE-Bench-style scores more crisply than most papers manage: benchmark gains and user experience have come apart, and many desiderata "cannot be described cleanly via automated unit testing." The Example blocks are the best writing in the paper. The Coverity report in Sec. 2.6, with churn and the 30% false-positive threshold past which engineers stop reading the tool, is exactly the kind of practitioner detail that academic surveys usually drop. And Sec. 5 is unusually honest about the paper's own speculative footing. Those are real strengths.

2. Whether the conclusion holds on the data

There is no dataset here, no experiment, no baseline, and no test statistic, so the usual questions do not apply directly. I applied a different standard: does each claim in the paper have an identifiable warrant, and are the paper's own internal artifacts consistent with each other? On that standard, the paper holds up in its argument and fails in several checkable places.

2.1 The taxonomy is defined but never applied as a structure. Sec. 2 announces three measures and says "we strive for AI to be capable across the board for all three measures." Nowhere in the Qeios version is there a table that places the tasks of Sec. 2.1 through Sec. 2.6 on those three axes. Instead, the assignments are dropped into prose, one at a time: Sec. 2.2.1 says refactoring "often has a low logical complexity" but "can be laborious in practice due to scope"; Sec. 2.2.2 says code migration has "very high scope" and "high logical complexity"; Sec. 2.4.2 says PR review requires "low logical complexity"; Sec. 2.5 says scaffolding and meta-code "often are small in scope and have low logical complexity"; Sec. 2.6 says full functional verification "generally has a high scope and medium logical complexity." Those five statements are the taxonomy's only output. A reader cannot recover the rule that produced them, cannot check whether tasks with no stated coordinates (Sec. 2.3.2 program analysis, Sec. 2.4.1 documentation, Sec. 2.4.3 code question answering) were omitted deliberately, and cannot reuse the scheme on a task the paper does not cover. For a contribution the abstract calls "a structured taxonomy," that is the central gap.

2.2 The logical-complexity scale has no operational boundary. Sec. 2 puts "most LeetCode problems" and "finding inputs to fuzz simple programs" at medium, and "difficult competition programming problems" at high. The separator between the two categories is the adjective "difficult." Similarly, low is defined as "little to no reasoning, such as writing CRUD applications or using APIs," which would classify

a good deal of genuinely hard integration work as low. If the measure is meant to be used by anyone other than the authors, each level needs a decision procedure, even a rough one: number of interacting components, presence of an algorithmic invention step, whether a correct solution is checkable in polynomial time, something.

2.3 The paper contradicts itself on PR review. Sec. 2.4.2 concludes that "reviewing PRs is a task requiring low logical complexity and can be automated relatively easily." The same section, two sentences earlier, lists what a reviewer must do: check style conventions, confirm no new bugs, verify that program invariants and guarantees still hold, and inspect whether tests are adequate. Verifying invariants is the task Sec. 2.3.2 describes as requiring abstract reasoning "across many different potential inputs and execution paths," and invariant inference gets its own paragraph in Appendix A.3 as a long-standing hard problem. Sec. 3.3 (Implicit constraints) argues that models infer repository conventions incorrectly, citing [74]. And Appendix A.4 reports that ByteDance's BitsAI-CR [301] works off "a manually crafted taxonomy of review rules," which is not what easy automation looks like. The "relatively easily" verdict should go.

2.4 The challenge count does not match. Sec. 1 says "in Sec. 3, we highlight nine challenges in the field that today's models face." The first paragraph of Sec. 3 says "we discuss ten key challenges in AI for code." The body has nine: Sec. 3.1 through Sec. 3.9. Nine is correct. This is small, but it sits in the two sentences most likely to be quoted.

2.5 Figure 1 carries argumentative weight I cannot check. Sec. 1 says Figure 1 summarizes "which directions correspond to each challenge," and that mapping is one of the paper's three advertised contributions. As rendered on Qeios, Figure 1 is present only as a caption; the diagram content is not legible as text. In the body, exactly one subsection of Sec. 4 carries an explicit back-reference in the "Challenges addressed: 3.3" form, namely Sec. 4.3.4. Every other subsection of Sec. 4 links back only through scattered inline mentions. So the claimed mapping lives entirely inside an image, and a reader working from the Qeios rendering cannot reconstruct it.

2.6 The empirical claims are single instances, and Sec. 3 states them as general properties. I counted the concrete evidence in Sec. 3: `astropy-#14181`, where models implemented only the read half of a serializer-deserializer pair (Sec. 3.3); `chartjs_Chart.js-7951`, where BM25 returned three wrong files (Sec. 3.5); the `test_case_convert_camel_to_snake` completion failure from [96] (Sec. 3.5); the "NextRouter was not mounted" session with Claude 3.7 (Sec. 3.8); and an `o3-mini` Lean 4 generation that emits Lean 3 begin syntax (Sec. 3.8). That is $n = 1$ in each case. I am content with anecdotes doing illustrative work in a position paper, and these are well-chosen anecdotes. The problem is grammatical rather than evidential:

the surrounding sentences generalize without hedging. Sec. 3.8 says "LLMs often write code that mixes and matches API constructs from different versions of the same library" and then supports it with one Next.js session and one Lean session. Either soften those sentences to claims about existence, or cite the benchmark results (CodeUpdateArena [106], GitChameleon [107]) that Sec. 3.8 already mentions and that would actually carry the general claim.

2.7 The one hard number in the challenges section is sourced to a moving target. Sec. 3.7 states that "On HumanEval, Qwen 2.5 Coder Instruct (32B)[103] has an accuracy of 83% in Python but only 27% in D." Footnote 3 attributes this to "the BigCode Models Leaderboard on the MultiPL-E benchmark[206]." Two problems. The sentence names HumanEval while the source names MultiPL-E, and although MultiPL-E is built by translating HumanEval, the paper should say so rather than let the reader reconcile it. More seriously, a live leaderboard cited with no access date and no revision identifier is not a reproducible source; the 83/27 split may well be right, but nobody can confirm it against the same snapshot the authors saw.

2.8 Two citation defects that I verified against the Qeios reference list. In Sec. 4.1.1 the text reads "KodCode [57] also showed that fine-tuning on verified synthetic data also leads to significant improvements." Entry [57] in the reference list is Xu C, Guan S, Greene D, Kechadi M, et al., "Benchmark data contamination of large language models: A survey," arXiv:2406.04244, 2024. That is a contamination survey, not KodCode, and the same [57] is cited correctly for contamination back in Sec. 3.1. So Sec. 4.1.1 points at the wrong work. Separately, AlphaDev is cited as [112] in Sec. 3.9 and as [239] in Appendix A.2; both entries resolve to the same paper, Mankowitz et al., "Faster sorting algorithms discovered using deep reinforcement learning," Nature 618(7964):257-263, 2023. The reference list carries it twice.

2.9 Industry figures are passed through without provenance. Sec. 2.2.2 states that COBOL "powers 80% of in-person financial services transactions and 95% of ATM swipes while processing \$3 trillion in commerce a day, with over 220 billion lines of COBOL code in production[29]." Sec. 2.2.3 credits V8 with "a 20x performance improvement," concurrent garbage collection that "reduced bloat by 100x," TurboFan at "5-10%," and background parsing that "reduced compile time by 20%," all against [38]. Sec. 2.3.1 reports that Meta engineers "accepted 73% of its tests" from ACH [42]. Sec. 2.2.2 reports a "3X latency drop" for Twitter, sourced to an InfoQ news item in footnote a. None of these are implausible, and I am not asking the authors to re-measure them. I am asking for the measurement year to appear next to each one, because a 2017 COBOL estimate and a 2025 COBOL estimate mean different things in a paper arguing about what to automate next.

2.10 Appendix A has no stated selection method. The appendix opens by saying the works surveyed "are by no means complete." That is a fair disclaimer, but it does not substitute for a method. There is no search strategy, no venue or database list, no inclusion criterion, and no cutoff date for a survey running to 356 references. Readers three years from now will want to know what window this covers.

3. Reproducibility

There is nothing to re-run here, so I read this section of my own review as: could someone else reconstruct the paper's claims from what is printed? Partly, and less than they should be able to.

No availability statement. I searched the Qeios text for any data, code, or artifact statement and found none. There is no repository for the taxonomy assignments, the reference collection, the prompts used in the model anecdotes, or the figure source. For a paper whose main output is an organizing scheme, releasing the scheme as a machine-readable table with the tasks scored on three axes would cost the authors almost nothing and would be the single most useful artifact they could publish.

Model anecdotes carry no version or parameter information. This is the failure mode I expect in LLM papers, and it is present here. Sec. 3.8 says "Claude 3.7 tries various solutions" with no dated snapshot, no date of the session, no prompt, no transcript, and no Next.js patch version beyond "13." Sec. 3.8 also reports prompting o3-mini with a Lean 4 problem and points to "Listing 3.8" for the output; as rendered on Qeios, no Listing 3.8 appears anywhere in the text, so the example cannot be inspected. Sec. 4.2.3 reports that SWE-Agent [66] powered by GPT-4o fails pydata_xarray-4750 on the issue description alone and succeeds when the issue discussion is included, referring the reader to Listing 2; on Qeios, Listing 2 appears as a caption ("Listing 2. SWE-Agent improves when incorporating issue discussions") with no content beneath it. No GPT-4o snapshot, no SWE-Agent commit, no temperature, no seed, no number of attempts. Since the xarray example is the paper's evidence that discussion improves issue resolution, and since SWE-Agent is stochastic at default settings, a single paired run is thin ground even for an illustration.

The retrieval example is not reconstructible. Sec. 3.5 reports that for chartjs_Chart.js-7951 from SWE-bench Multimodal [95], BM25 over the issue description returns `src/scales/scale.radialLinear.js`, `src/scales/scale.linearbase.js`, and `src/helpers/helpers.canvas.js` in the top three, while the needed edits are in `src/elements/element.bar.js` and `src/controllers/controller.bar.js`. BM25 results depend on the tokenizer, the `k1` and `b` parameters, whether the index is per-file or per-

chunk, and whether identifiers are split on camel case. None of that is given, so the result cannot be regenerated or contested.

The taxonomy assignments have no coding protocol. Placing a task at "medium logical complexity" is a judgment call. The paper reports no coding manual, no second annotator, and no agreement measure. Given that Sec. 2's assignments are asserted rather than derived, some statement about how the authors reached them belongs in the text, even if the honest answer is that they were settled by discussion among the co-authors.

The paper has no snapshot date for its own claims. Sec. 5 concedes that "a reader reading this paper a few months down the line will find that several of the mentioned challenges will have been partially or entirely resolved," which is the right instinct. Then act on it: state the date at which the state of the art was assessed, so a later reader can tell which model generation the Sec. 3 claims were written against. The Qeios posting date of April 4, 2025, is not the same thing as the date the authors last checked the models.

One loose thread in the Acknowledgements. The authors thank contributors "for bringing up illustrative examples mentioned in this paper," and the list includes "Chuyue Sun (incomplete specification in Verus)" and "Songlin Yang (syntax error in Triton)." I could not find either example in the Qeios text. Triton does not appear at all; Verus appears in Appendix A.6 only as one of several verification languages, not as an incomplete-specification example. Either two examples were cut and the acknowledgement was not updated, or two examples live in figures or listings that did not survive the Qeios rendering. Worth checking against the source.

4. What specifically to fix

1. **Sec. 2, new table.** Add a single table with one row per task from Sec. 2.1 to Sec. 2.6 and three columns for scope, logical complexity, and level of human intervention, with a fourth column giving the range rather than a point where a task genuinely spans levels (refactoring is the obvious case: Sec. 2.2.1 already argues it is low complexity but potentially project-scope). This is the artifact the abstract promises, and the paper currently does not deliver.
2. **Sec. 2, logical complexity definition.** Give each of the three levels a test a reader can apply. Right now, "most LeetCode problems" sits at medium and "difficult competition programming problems" at high, and the only thing separating them is the word difficult.

3. **Sec. 2.4.2, final sentence.** Strike "requiring low logical complexity and can be automated relatively easily." It is contradicted by the invariant-checking requirement listed in that same paragraph, by Sec. 3.3 on implicit constraints, and by the hand-built rule taxonomy behind BitsAI-CR [301] in Appendix A.4.
4. **Sec. 1, paragraph 4 and Sec. 3, opening paragraph.** One says nine challenges, the other says ten. The body has nine (Sec. 3.1 to Sec. 3.9). Fix the ten.
5. **Figure 1 and Sec. 4.** Two things. Make Figure 1 legible in the Qeios rendering, and mirror its content in the text by giving every subsection of Sec. 4 the "Challenges addressed: 3.x" tag that currently appears only at Sec. 4.3.4. The mapping should not be readable only from an image.
6. **Sec. 4.1.1, sentence beginning "KodCode [57]".** Wrong reference. [57] is Xu et al., the contamination survey, which is cited correctly in Sec. 3.1. Add the KodCode paper to the reference list and point at it.
7. **References [112] and [239].** Both are Mankowitz et al., Nature 618(7964):257-263, 2023. Merge them and renumber, so that Sec. 3.9 and Appendix A.2 cite AlphaDev by the same number.
8. **Sec. 3.7 and footnote 3.** The text says HumanEval, the footnote says MultiPL-E. Say which, note that MultiPL-E is a translation of HumanEval if that is the intent, and add the date on which the BigCode leaderboard was read. Better still, cite a fixed model card or the MultiPL-E paper's own numbers instead of a live leaderboard.
9. **Sec. 3.8 and Sec. 4.2.3.** Listing 3.8 (the o3-mini Lean 4 output) is referenced but absent from the Qeios version, and Listing 2 (the SWE-Agent comparison) appears as a caption with no body. Both need to render. Listing 1, if it exists, is also not visible.
10. **A metadata line under each model anecdote.** For Sec. 3.8 (Claude 3.7, Next.js), Sec. 3.8 (o3-mini, Lean 4), and Sec. 4.2.3 (SWE-Agent with GPT-4o, pydata_xarray-4750): give the dated model snapshot, the date of the session, the sampling settings, and how many attempts were made. One line each. For the xarray comparison, say whether the paired result was observed once or repeated.
11. **Sec. 3.5, BM25 example.** Add the retrieval configuration: tokenizer, identifier splitting, k1 and b, and index granularity. Without them, the top-3 file list is an assertion.
12. **Appendix A, opening paragraph.** Two or three sentences on method: where the references were gathered, what the inclusion bar was, and the cutoff date. The existing "by no means complete" disclaimer is honest but tells a reader nothing about coverage.
13. **Acknowledgements.** Reconcile the thanks for the Verus incomplete-specification example and the Triton syntax-error example against the body, where I could find neither.

14. **Copy-editing, specific instances.** Sec. 2.2.2, "early versions of COBOL". Sec. 2.6, "they has not yet become mainstream". Sec. 3.3, "corporations may enforces codes of conduct". Sec. 3.4, Library learning paragraph, "Designing APIs and libraries are designed with useful abstractions often leads to more code reuse" (the sentence has two predicates fighting each other). Sec. 3.8, "In order to to successfully deal with version changes". Appendix A.2, "Section A.3may help". Appendix A.3, "Specificatly, OSS-Fuzz-Gen[253]". Appendix A.6, "Recently, the ability of LLMs to write formal verification code." has no predicate at all.
15. **Sec. 5 and Sec. 6, genre.** Sec. 6 calls this a position paper; the Qeios header tags it REVIEW ARTICLE; Appendix A is a survey longer than several of the main sections. Add a sentence in Sec. 1 saying plainly that the main text is a position piece and the appendix is a companion survey, so readers know which standard to hold each part to.

5. Recommendation

Minor revisions.

The argument is sound, and the paper is useful. Its core claim, that the field has over-indexed on code generation at the function scope and under-invested in the rest of software engineering, is well made and supported by the breadth of Sec. 2 and Appendix A. I would expect this to be read and cited by people planning benchmark work, and the construct-validity discussion in Sec. 3.1 deserves the attention it will get. None of my objections attack the thesis.

What I am asking for is mostly bookkeeping, and it is the kind of bookkeeping that determines whether a taxonomy gets used or admired and forgotten. The three measures need a table and a decision procedure (items 1 and 2), the challenge count and the challenge-to-direction mapping need to be internally consistent and readable outside the figure (items 4 and 5), the two citation errors need correcting (items 6 and 7), and the model anecdotes need the version and parameter lines that would let someone else check them (items 9, 10, and 11). The PR-review sentence in Sec. 2.4.2 (item 3) is the one substantive claim I think is wrong rather than merely underspecified, and the paper's own material is enough to fix it. None of this requires new experiments, which is why I do not ask for major revisions.

Declarations

Potential competing interests: No potential competing interests to declare.