Code Review Strategies in AI-Assisted Programming

Job Schepens

February 4, 2026

Overview

This talk:

- Explore code-review strategies in AI-assisted programming
- not an endorsement of AI use
- Goal: think about incorporating critical thinking when using AI

What is AI?

Here: Focus on LLMs, preferably open models such as Mistral 24B via chat.kiconnect.nrw, etc. that generate code based on prompts, used via a terminal, IDE, software integration, or browser. Most models are closed source (not transparent), not EU-based, dependent on company support, etc, see also KI-Whitelist der Universität zu Köln.

Example use-cases:

- Conversation-like ‘sounding board’ for identifying problems and suggesting improvements, reflects on code structure, judges quality
- Implementation assistant that generates code from scratch, corrects mistakes, etc.

Many guidelines, laws, regulations:

- **EU AI Act**
  - puts duties on AI providers and on organisations that use AI
- **UzK KI-Richtlinie**
  - PDF (27.1.2026)
- **ACL (2023)**
  - Focus on transparency and accountability
  - Responsible NLP Research/
- **Wikipedia**
  - Focus on verification
  - Do not generate articles from scratch
  - Only in training modules: Never copy-paste from GenAI
- **DFG (German Research Foundation)**
  - Focus on transparency and existing standards for research integrity
  - wissenschaftliche-integritaet.de

AI Risks and (Un)critical AI Literacy?

AI Act (as based on FOI compliance checker):

- "No risk":
  - the AI system is intended to perform a narrow procedural task;
  - the AI system is intended to improve the result of a previously completed human activity;
  - and more
- "General Purpose AI models with Systemic Risk"
  - includes GPT 4 OSS through Academic Cloud / nrw.connect?
- "Prohibited systems"
  - e.g. exploiting vulnerabilities, manipulation, emotion recognition, etc.
- "AI Literacy obligations"

> "universities need to take their role seriously to safeguard higher education, critical thinking, expertise, academic freedom, and scientific integrity" (Guest et al., 2025)

Critical AI literacy criticism includes (e.g. Guest et al., 2025):

- social and environmental harms
- unaligned with university values of critical thinking and scientific integrity
- deskilling of researchers
- 'critical washing' — encouraging AI use while being 'aware of the risks'
- lower literacy–greater receptivity (Tully et al., 2025)
- deskilling (Anthropic, 2024, Bouchard, 2025)
- less security, e.g. Simon Willison's lethal trifecta
- see also the list of 80+ recent books on the website of Prof. dr. Dagmar Monett

> "While automation may give the false impression of rigor and efficiency, it leads to conceptual and scientific deskilling, deteriorates reflexive theorizing…" Van Rooij & Guest (2025)

Critical review of generated code

Critical review as a way to reduce risks, and address KI-Richtlinie §2.4 (human control) and §9.1 (output verification)?

- Verification, e.g. is x correct?
- Explaining, e.g. where does x come from?
- Questioning, e.g. is x necessary?
- Resisting “use it or lose it”, does using AI harm comprehension?
- Also, does it corrupt my work? does it give away my data? is it harmful to my personal communication, does it remove space and time to learn?

Idea: start from classic good practice rules of software engineering

Practice	Idea
KISS	Keep it simple
DRY	Don’t repeat yourself
YAGNI	You ain’t gonna need it
SoC	Separation of concerns

Clausemate Project

- German pronoun-clause mate extraction, see GitHub repo
- ~2,000+ lines Python code
- Evolved: linear script → modular → feature expansion → quality enforcement

Phase 1 → 2: From Script to Modular Design

- Initial script-based approach was working fine, but lacked features
- Refactored: abstract base classes, typed dataclasses
- **Problem:** Could have foreseen this earlier, reactive not proactive

Phase 2 → 3: Adding more Features

- Added multi-file processing
- Created 3 separate file parsers: 1,470 total lines (60% of project!)
- **Problem:** A lot of overlap between parsers.

Phase 3 → Adding Quality Checks

- Adding mypy, pre-commit hooks, testing
- **Problem:** Reactive measures instead of preventing issues

Problems with classic good practices:

1. **YAGNI:** Many quality checks
2. **DRY:** Repeated token parsing
3. **SoC:** Main.py handles format detection + parsing + streaming + analysis + export
4. **YAGNI:** `benchmark.py`, unused dependencies (spacy, scikit-learn, …)

Some additional critical developments:

- Increasing fear that changes might break something
- Things become too complicated to explain

Fast Iteration vs. Slow Questioning

“Fast” Iteration

```
Generate (AI) → Review (Human) →
Refine → Integrate (one-way)
```

Advantages:

- ✓ Fast

Problems:

- ⚠️ AI output may be completely wrong
- ⚠️ Review becomes less sophisticated
- ⚠️ Not easy to review 10 new features at once
- ⚠️ Can quickly become too passive

Clausemate Example:

Three working parsers, but with much overlap. Never asked “why 3?”

“Slow” Questioning**

```
AI                   Pro-active 'Review'
───────────          ──────────────────────
Suggests x      ↔    Question necessity - YAGNI
Verbose code    ↔    Demand simplicity - KISS
Duplication     ↔    Enforce reuse - DRY
Mixed concerns  ↔    Separate clearly - SoC
```

Advantages:

- ✓ More pro-active
- ✓ Mandated by UzK Richtlinie

Problem:

- ⚠️ Requires expertise and time. “Critical thinking”

Clausemate Example:

Better understand text structure first before developing three different parsers.”

Summary

Fast vs slow

1. Short-term efficiency vs. long-term skill development
2. Slowing down can identify problems, make decisions explicit
3. Slow review better aligns with UzK KI-Richtlinie

Some principles for slow review

1. Comprehension: Explain x (aloud) before accepting
2. Complexity: Not just add or restructure for the sake of it
3. Necessity: What breaks without x? YAGNI
4. Collaboration

Warning signs

1. Wrote x 2 weeks ago, but don't remember
2. Adding new things instead of fixing issues
3. Just run and see
4. Messy documentation (relates to UzK KI-Richtlinie §12.4's documentation obligations)

Conclusions

1. Classic best practices may help to prevent problems
   - KISS/DRY/YAGNI/SoC
2. AI harms understanding when not slowed down
   - Critical reviewing necessary
3. Complexity and messiness accumulate quickly
   - System becomes intransparent, difficult to verify

Take-Home Message:

AI builds faster than we can understand easily. Train your critical thinking skills and safeguard research integrity.

> "Für jede KI-gestützte Anwendung ist eine namentlich benannte Rolle mit menschlicher Letztverantwortung festzulegen." (UzK KI Richtlinie §2.4b)

References

Some References:

- Guest, O., et al. “Against Uncritical Adoption of ‘AI’ in Academia” https://philpapers.org/rec/GUEATU
- Anthropic: “AI-Assisted Coding Effects” https://www.anthropic.com/research/AI-assistance-coding-skills
- Veldhuis, Annemiek, Priscilla Y. Lo, Sadhbh Kenny, and Alissa N. Antle. “Critical Artificial Intelligence Literacy: A Scoping Review and Framework Synthesis.” International Journal of Child-Computer Interaction 43 (March 2025): 100708. https://doi.org/10.1016/j.ijcci.2024.100708.
- Tully, Stephanie M., Chiara Longoni, and Gil Appel. “Lower Artificial Intelligence Literacy Predicts Greater AI Receptivity.” Journal of Marketing 89, no. 5 (2025): 1–20. https://doi.org/10.1177/00222429251314491.
- Bouchard, Jeremie. “ChatGPT and the Separation between Knowledge and Knower.” Education and Information Technologies 30, no. 8 (2025): 10091–110. https://doi.org/10.1007/s10639-024-13249-y.