CISUC

Safety-Critical Systems:
Human Factors Are Back in Full Force

Henrique Madeira

CISUC, University of Coimbra, Portugal

87th IFIP WG 10.4 Meeting, Salvador, Brazil, February 7th-10th, 2025

Example 1: breast cancer diagnosis

Class II medical device: poses a moderate to high risk to patients

https://developer.nvidia.com/blog/new-ai-breast-cancer-model-is-the-first-to-show-diagnostic-process/

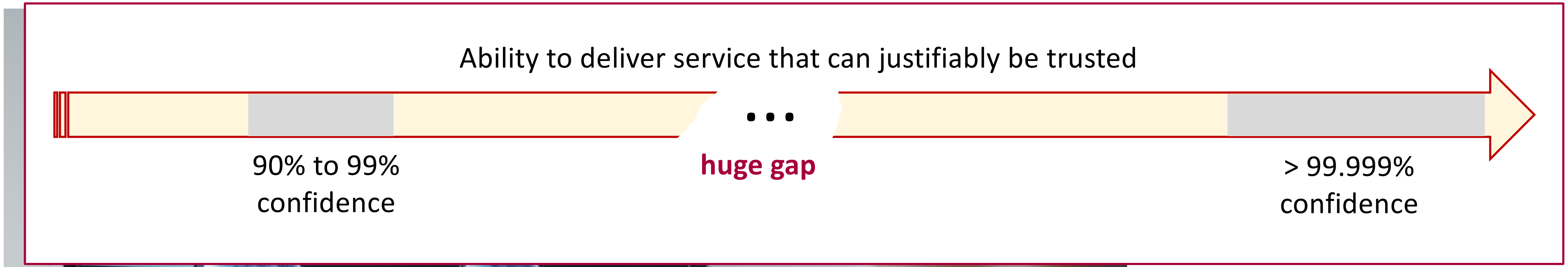
Goal:

To assist radiologists and oncologists, not replace them.

Reality:

If the system is good enough, doctors may blindly trust its diagnoses.

Example 1: breast cancer diagnosis



Ability to deliver service that can justifiably be trusted

90% to 99% confidence

huge gap

> 99.999% confidence

https://developer.nvidia.com/blog/new-ai-breast-cancer-model-is-the-first-to-show-diagnostic-process/

Reality:

If the system is good enough, doctors may blindly trust its diagnoses.

Example 2: software development

https://fyclabs.com/blog/the-role-of-artificial-intelligence-in-software-development/

Developer: prompts a (crude) description of the code he/she wants

Tool: provides the code

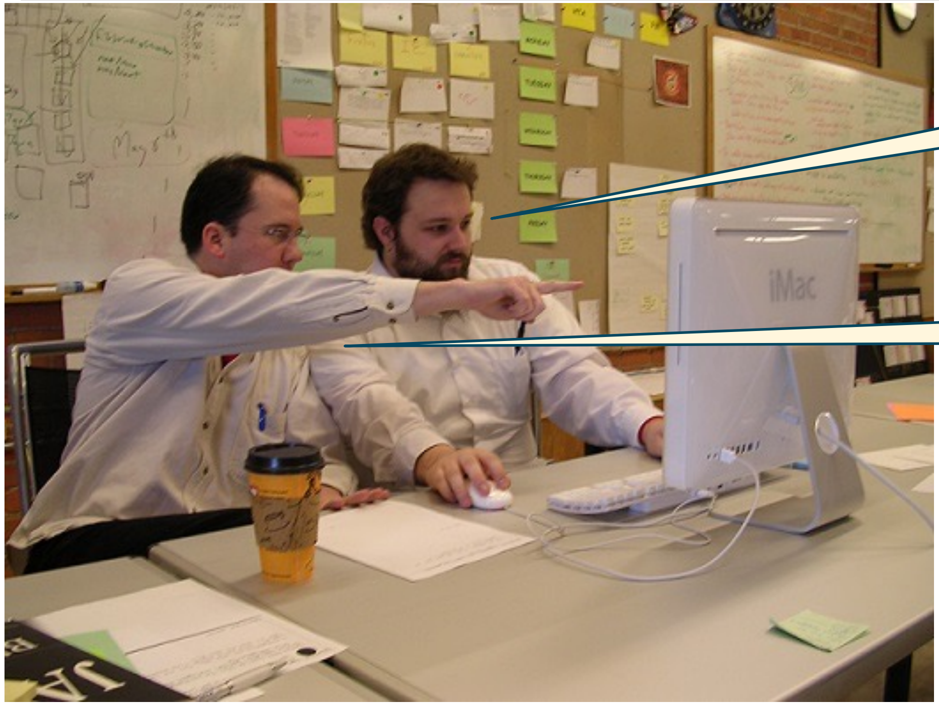
Developer: checks if it compiles OK and corrects easy mistakes

Developer: pastes error messages or problematic code into the tool

Tool: identifies bugs, logic errors, or inefficiencies and suggests fixes

Developer: declares “code complete” ☺ ☺ ☺

Is old pair programming back?



He is playing the role of the AI tool

He is the programmer

https://codingjourneyman.com/2015/05/11/extreme-programming-pair-programming/

Unfortunately, the reality is far more complex than this optimistic view...

Code developed with AI assistants

- Fails for complex requirements that fall outside of the LLM training space
- Has bugs, even for relatively simple code (80% of the respondents often or always experience bugs when using LLMs to generate code)
- AI-generated code can be difficult for humans to understand

F. Tambon, A. M. Dakhel, A. Nikanjam, F. Khomh, M. C. Desmarais, and G. Antoniol, "Bugs in large language models generated code," arXiv preprint arXiv:2403.08937, 2024

Code developed with AI assistants

- Fails for complex requirements that fall outside of the LLM training space
- Has bugs, even for relatively simple code (80% of the respondents often or always experience bugs when using LLMs to generate code)
- AI-generated code can be difficult for humans to understand
- Introduces new types of bugs
- Has security vulnerabilities

FN. Perry, M. Srivastava, D. Kumar, and D. Boneh, "Do users write more insecure code with AI assistants?," in Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security, pp. 2785–2799, 2023.

Code developed with AI assistants

- Fails for complex requirements that fall outside of the LLM training space
- Has bugs, even for relatively simple code (80% of the respondents often or always experience bugs when using LLMs to generate code)
- AI-generated code can be difficult for humans to understand
- Introduces new types of bugs
- Has security vulnerabilities
- Bug fixing with AI is not particularly reliable

Nan Jiang, Yi Wu, "RepairCAT: Applying Large Language Model to Fix Bugs in AI-Generated Programs" APR '24: Proceedings of the 5th ACM/IEEE International Workshop on Automated Program Repair, ICSE, 2024.

Code developed with AI assistants

- Fails for complex requirements that fall outside of the LLM training space

K. Liu, Y. Liu, Z. Chen, J. M. Zhang, Y. Han, Y. Ma, G. Li, and G. Huang, "Llm-powered test case generation for detecting tricky bugs," arXiv preprint arXiv:2404.10304, 2024.

Code developed with AI assistants

- Fails for complex requirements that fall outside of the LLM training space
- Has bugs, even for relatively simple code (80% of the respondents often or always experience bugs when using LLMs to generate code)
- AI-generated code can be difficult for humans to understand
- Introduces new types of bugs
- Has security vulnerabilities
- Bug fixing with AI is not particularly reliable

How can code developed with AI assistants be improved?

The mainstream research focuses on the LLM and AI aspects of the pair.

Most likely, the key is in the programmer.

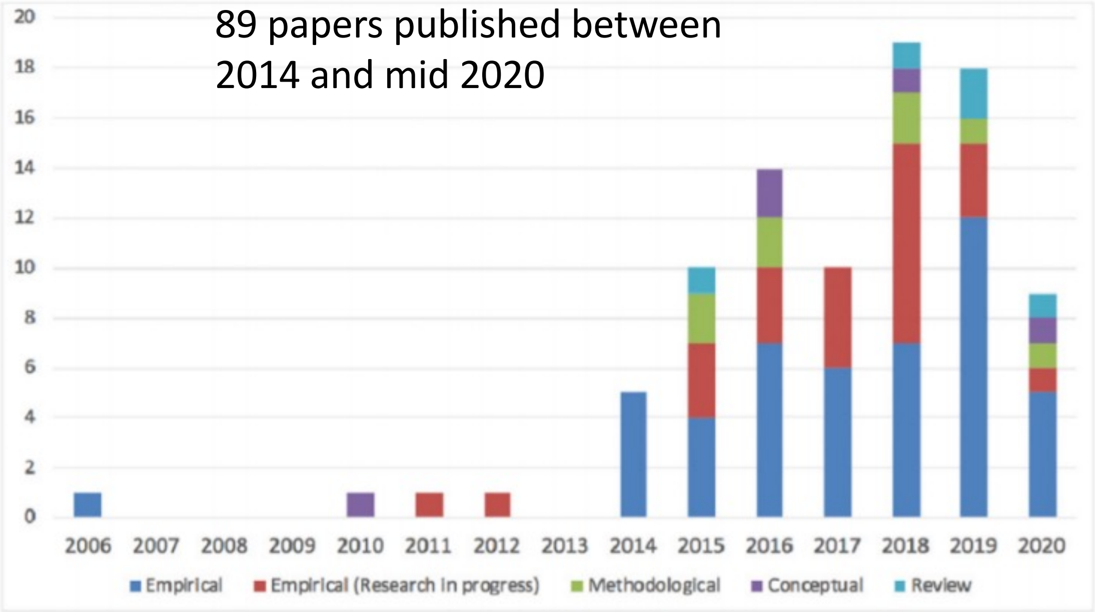
Human factors will define the future of software development.

https://fyclabs.com/blog/the-role-of-artificial-intelligence-in-software-development/

NeuroSE

NeuroSE is a “research field in software engineering (SE) that makes use of neurophysiological methods and knowledge to better understand the software development”

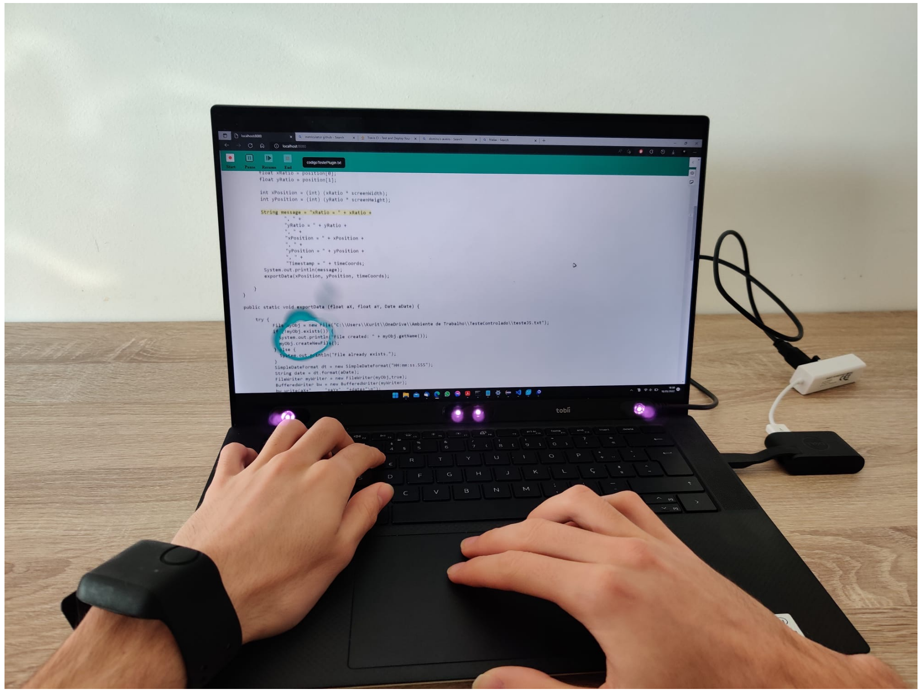
Barbara Weber, Thomas Fischer, René Riedl, **“Brain and autonomic nervous system activity measurement in software engineering: A systematic literature review”**, Journal of Systems and Software, DOI: 10.1016/j.jss.2021.110946, March 2021.



89 papers published between 2014 and mid 2020

- The number of NeuroSE papers from mid 2020 until now is **250+**
- **NeuroSE definition is already outdated:** recent papers are doing much more than using neurophysiological to “better understand the software development”.
- New **neuroscience inspired methods and tools.**

iReview: evaluates programmers' code comprehension and grades the quality of their code reviews



Assess

Assess code comprehension difficulty through measuring cognitive load changes using a low-cost smartwatch to obtain Heart Signals.

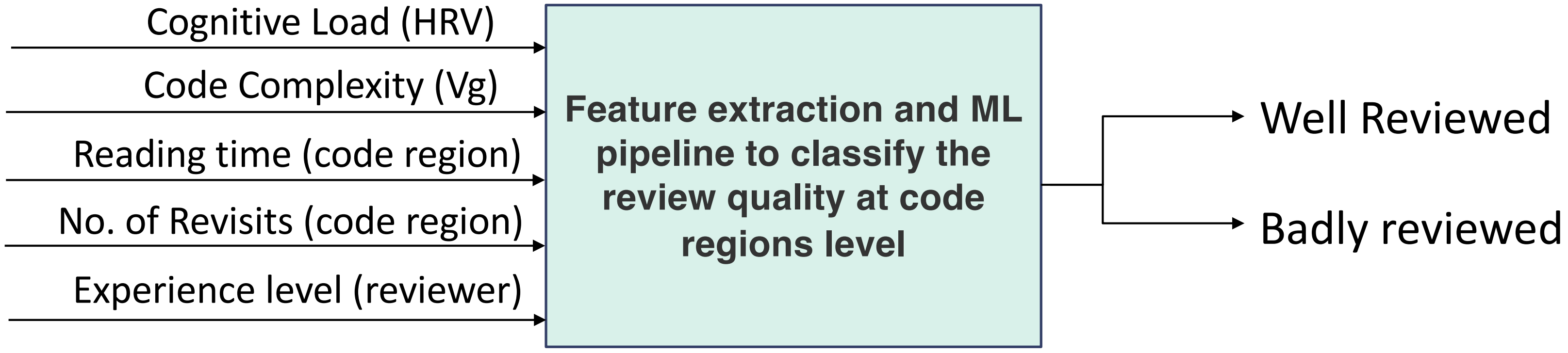
Indicate

Indicate the code regions that are associated with high cognitive load and classified as "badly reviewed" using a desktop eye-tracker.

Explain

Explain the classification result (why "badly reviewed"?).

iReview code review quality classification



Cognitive Load (HRV)

Code Complexity (Vg)

Reading time (code region)

No. of Revisits (code region)

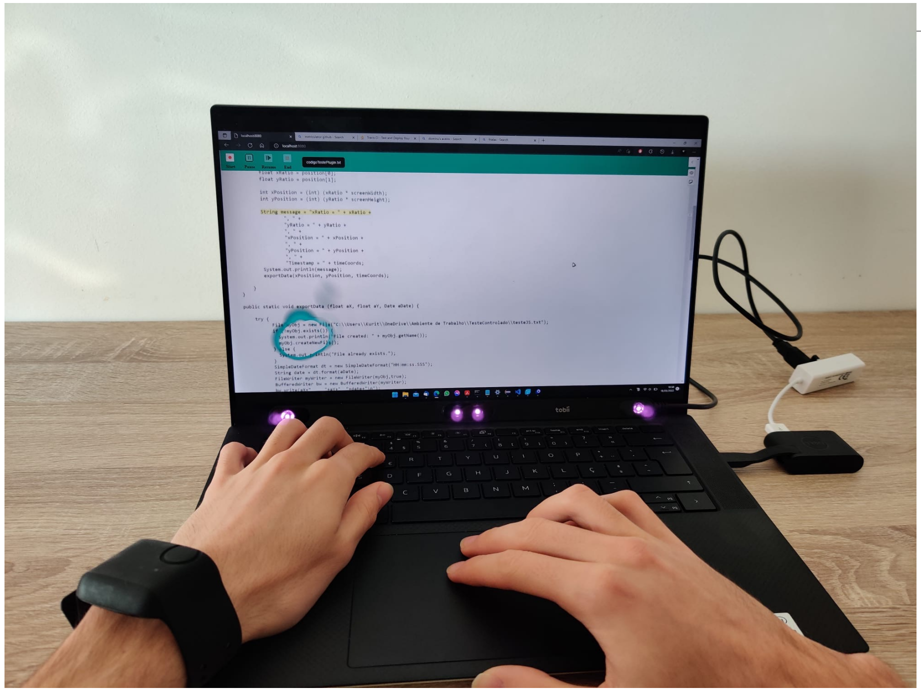
Experience level (reviewer)

Feature extraction and ML pipeline to classify the review quality at code regions level

Well Reviewed

Badly reviewed

iReview is low intrusive



Conclusion

- Human factors are essential for safety-critical systems at different levels, not only in terms of differing perceptions of safety and classic moral dilemmas regarding the consequences of safety failures.
- The support of AI in safety-critical applications, where humans assume the final responsibility for assuring safety, **does not actually guarantee safety**.
- The use of AI-generated code to develop software is a strong trend that will also affect the development of safety-critical systems. This could create a potentially dangerous scenario, with a probable increase in software faults and vulnerabilities.
- The current approach to software development, combining AI-generated code with human programmers, dramatically increases the difficulty of ensuring reliable code. The human role in this pairing is crucial for guaranteeing safety.

Acknowledgements

This work is partially financed through national funds by FCT - Fundação para a Ciência e a Tecnologia, I.P., in the framework of the Project UIDB/00326/2025 and UIDP/00326/2025.

iReview was partially supported by the BASE (Biofeedback Augmented Software Engineering) project, Fundação para a Ciência e a Tecnologia, contract No IT057-18-7327.