Acta Scientific COMPUTER SCIENCES

Volume 7 Issue 2 May 2025

Perspective Article

Science/Education Portraits XI: The Advent of Vibe Coding

Maurice HT Ling*

HOHY PTE LTD, Singapore

***Corresponding Author:** Maurice HT Ling, HOHY PTE LTD, Singapore.

Received: April 10, 2025

Published: April 15, 2025

Abstract

The term "vibe coding" was coined by Andrej Karpathy on 03 February 2025 and defined in Merriam Webster dictionary as "the practice of writing code, making web pages, or creating apps, by just telling an AI program what you want, and letting it create the product for you". This may be the dream scenario of Frederick B. Thompson and Jean E. Sammet – both advocating English as a programming language in 1966, and I argue that vibe coding is its natural progression to today. For the purpose of this article, I tried the implementation of ProjEB (a command line interface electronic laboratory notebook) using vibe coding. Using Claude 3.5 Sonnet on GitHub Copilot within Microsoft VS Code, I produced working 994 lines of operational codes with 1685 lines of test codes within 28.5 hours from 161 lines of requirements as prompt. From this trial, I learnt that vibe coding is not exactly easy to accomplish as it substantially tapped on my previous experiences with software design and coding. I feel like a software architect or project manager having to envision the end result at the start, and to describe the vision to GitHub Copilot as the initial prompt. However, it does have enormous potential ahead and if this is the start of vibe coding, I am excited.

Keywords: Vibe Coding; AI-Assisted Coding; AI-Coding Assistant; Chatbot; Source Code Generation; Large Language Models; Coding Tools; Prompt Engineering

Introduction

Vibe coding is a form of AI-assisted coding

On February 03, 2025; Andrej Karpathy (URL 1) of Eureka Labs coined the term "vibe coding" in a tweet (URL 2) as: *There's a new kind of coding I call "vibe coding", where you fully give in to the vibes, embrace exponentials, and forget that the code even exists. It's possible because the LLMs (e.g. Cursor Composer w Sonnet) are getting too good. Also I just talk to Composer with SuperWhisper so I barely even touch the keyboard. I ask for the dumbest things like "decrease the padding on the sidebar by half" because I'm too lazy to find it. I "Accept All" always, I don't read the diffs anymore. When I get error messages I just copy paste them in with no comment, usually that fixes it. The code grows beyond my usual comprehension, I'd have to really read through it for a while. Sometimes the LLMs can't fix a bug so I just work around it or ask for random changes until it goes away. It's not too bad for throwaway weekend projects, but still quite amusing. I'm building a project or webapp, but it's not really coding - I just see stuff, say stuff, run stuff, and copy paste stuff, and it mostly works.*

Merriam Webster (https://www.merriam-webster.com/slang/vibe-coding) then defines "vibe coding" as "writing computer code in a somewhat careless fashion, with AI assistance", and describe it as "the practice of writing code, making web pages, or creating apps, by just telling an AI program what you want, and letting it create the product for you", with a further elaboration that "in vibe coding the coder does not need to understand how or why the code works, and often will have to accept that a certain number of bugs and glitches will be present".

From these definitions and descriptions, the hallmarks of vibe coding can be listed as (i) writing computer codes with artificial intelligence (AI) using prompts, (ii) no review of AI-generated computer codes, (iii) repeated iterations until AI-generated computer codes is free of errors, and (iv) perhaps the person giving the prompt has no knowledge of computer coding; implying completely trusting the AI-generated codes to execute as intended and without errors after enough iterations. This is substantially different from AI-assisted coding [1,2] which can be defined as the use of AI-tools in coding and this can include initial code generation by the AI-tools for testing and evaluation. AI-assisted coding has substantial adoption in the software engineering community [3]. This is supported by a study with 234 undergraduate students find that AI-assisted pair programming positively impacts intrinsic motivation and reduced programming anxiety [4], suggesting potential use of AI-assisted coding.

One of the main issues with AI-generated codes using large language models (LLM) [5] is hallucination. In 1995, Stephen Thaler demonstrated that hallucinations (also known as phantom experiences) can emerge from random variations to weights in artificial neural networks [6]. This phenomenon is well-observed in large language models [7,8] where models generate plausible-sounding but nonsensical information [9]. Specifically, hallucinations have been shown in ChatGPT (13 February 2023's version) [10] with hallucination rates of 39.6% for GPT-3.5 and 28.6% for GPT-4 [11].

However, vibe coding does not insist error free codes at the first iteration as Andrej Karpathy said that "when I get error messages I just copy paste them in with no comment, usually that fixes it." Hence, iterations are allowed. The iteration of code generation and code execution until error free fits the definition of a Chinese room [12], which can be procedural. Conceptually, this meant that a user with no knowledge of computer coding can issue requirements to a vibe coding assistant which then runs an iteration loop until the AI-generated codes is error free. This is predicated on unambiguous requirements by the user, and reduction in both the quantity and magnitude of errors at each iteration. In practice, unambiguous requirements often require at least basic coding knowledge or power-user level.

AI-Assisted Coding (And by Extension, Vibe Coding) is a logical progression of coding tools

Before internet, online forums such as Usenet newsgroups has become an avenue for many programmers to seek help from more experienced individuals. Usenet was conceived by Tom Truscott and Jim Ellis in 1979 and established in 1980. In 1996, Expert Exchange started as a forum for IT professionals to seek help within the community. This is followed by Stack Overflow [13], in 1998, as a question-and-answer site for programmers. At this point, a programmer needing help has resources of the entire world but is still dependent on another person willingness to help. This led to many students posting assignment questions or genuinely unexperienced persons posting simple programming questions, which resulted in many rude comments in the tone of "do not post your homework questions".

With increasing amount of sample codes and operational codes online, specialized tools such as code search engines [14,15] emerged. This reduces the need for another person needing to respond to the question and potentially incurring wrath, as online code repositories become a resource library. However, even with code search or someone replying to a coding query, it is often difficult to provide sufficient context, and the reply needs to be contextualized to the current problem. Hence, a high level of abstraction is required to isolate the issue at hand so that another person can help. The hope for a tool that can contextualize the problem or able to reason the problem in the context of the codebase, is ever present.

ChatGPT was launched on 30 November 2022 based on generative pre-trained transformer [16], and has revolutionized many fields; such as, education [17], creative arts [18], healthcare [19], research [20], and businesses [21]; for better or for worse. ChatGPT belongs to a class of applications known as chatbots and mimics spoken language [22]. Within the year, studies have shown that ChatGPT can generate source codes [23] comparable to high-performing students [24]. The most important aspect of AI tools, such as ChatGPT, is its ability to contextualize the problem. This is supported by studies showing that contextualized AI tools [25,26] are able to act as coding assistants; thus, AI-assisted coding [2] or AI-assisted programming [27] emerged. AI-assisted coding can also mean that the world of code repositories and forum posts is now within reach. More importantly, AI-coding assistants also meant a dedicate assistant who will never be frustrated. Conceptually, AI-coding assistants represents collective knowledge of the corpus it was trained on.

Source code generation from specification has a long history which undoubtedly pre-dates ChatGPT [28-30]. In 1966, both Frederick B. Thompson (professor of applied philosophy and computer science in Caltech) [31] and Jean E. Sammet (the developer of FORMAC programming language and one of the developers of COBOL programming language) [32] made a plea to consider using English as a programming language. This led to the development of REL in 1969 [33], which influenced the development of SEQUEL [34] which became SQL

Bringing the natural language interface of chatbots to source code generation seems to be obvious, and it may be what Frederick B. Thompson and Jean E. Sammet were looking for. If natural English language can be used as a specification language, then it is possible to describe requirements to a chatbot (in the form of an AI-coding assistant) and generate source codes as an outcome. This fits Merriam Webster's definition of vibe coding – "by just telling an AI program what you want, and letting it create the product for you", where the AI program is the AI-coding assistant.

Trial implementation of ProjEB using vibe coding

I had been wanting to develop an electronic laboratory notebook (ELN) [35] to replace something that I had developed in the

past, CyNote [36]. The new project, ProjEB (Project Electronic Book) is to be a command-line interface (CLI) system at the start and a graphical user interface (GUI) added on top of the CLI. However, I had not touched it since started ProjEB as a GitHub repository on 20 August 2018 (URL 3).

Finally on 07 April 2025, I decided to use vibe coding to try to implement a rudimentary version of ProjEB. I used Claude 3.5 Sonnet (URL 4; available since 21 June 2024) on GitHub Copilot (URL 5; available since 29 June 2021) [37] within Microsoft VS Code. Here is what I did. In phase 1, I wrote an extensive 161 lines (more than 4 pages) requirement for ProjEB that almost reminiscent to an actual Product Requirement Document (PRD) or Product Specification Document (PSD) – the commit on 08 April 2025 is at https://bit.ly/ProjEB-RE.

I had some previous cursory attempts to use generative AI to produce example codes that I can use. Through these attempts, I realized that I need to specify as many things as possible; hence, this extensive list of requirements (completed at 6.30 am of 08 April 2025) which I use as my initial prompt for phase 2. Phase 2 is the generation of initial codes into the files peb.py (the command line application of ProjEB) and database.py (containing database management codes), as well as example command line usage in run_peb.bat file. This is followed by Phase 3, which is getting Github Copilot to write initial test codes into test_peb.py (containing test cases for peb.py) and test_run_peb.py (containing test cases for command line usage – equivalent to test cases for rub_peb.bat). Phase 4 is then the iterative (a) execute test_peb.py (command = coverage run -m unittest test_peb.py && coverage report) and test_use_peb.py (command = python -m unittest test_use_peb.py), (b) feeding the error messages into Github Copilot to generate amendments with the occasional prompt to increase code coverage, and (c) applying the amendments to the corresponding code files. Phase 4 is deemed completed when code coverage for the 2 main files, peb.py and database.py, reaches 60% by executing test_peb.py without any failure; and executing test_use_peb.py without any failure.

This trial ended on 11 am of 09 April 2025 with 24 tests in test_peb.py and 9 tests in test_use_peb.py, amounting to 64% code coverage in database.py and 60% code coverage in peb.py. This experiment spanned 28.5 hours (from URL 6, 7); with 525 lines of codes in database.py and 469 lines of codes in peb.py (total of 994 lines of operational codes), and 1350 lines of codes in test_peb.py and 335 lines of codes in test_rub_peb.py (total of 1685 lines of test codes). This gives a crude estimate of 94 lines of codes per hour; not accounting for the 11.5 hours I spent sleeping (I was down with fever), the 8 hours of teaching I had on 08 April 2025, and so on.

My thoughts of vibe coding

Having attempted vibe coding myself, I have some thoughts about it. Here, I will elaborate on my thought from 2 different levels – (a) my thoughts and learnings before and after my attempt at vibe coding, and (b) my thoughts about vibe coding as a whole.

Since early 2023, I have been dabbling with ChatGPT. Although there are evidence of hallucination [10], especially in the area of fabricated citations [38-40]; I was decently surprised by the quality of the generated data analysis codes, which is supported by recent studies [41,42]. Hence, I felt that LLMs may be more suited for code generation which is more objective as compared to descriptions – much like grading mathematics examination scripts versus literature examination scripts. Yet, even in code generation, I realized that ambiguous or imprecise prompts tend to give wildly different results at different sessions using the same prompt. Hence, in Phase 1 (prompt engineering), I decided to explicitly define my requirements into a prompt at the start as clear requirement plays a critical role in success of software development projects [43,44]. Writing these requirements is reminiscent of my previous role as Senior Scientist where I spent months writing product specification document (PRD) for a software project. I even used similar language of "users should be able to …". It also reminds me of the days where I was a Research Associate and needs to communicate via emails with my collaborator in another state on experiments – I had to explain the order of which DNA samples must be loaded onto agarose gel so that I can use the resulting gel image in my publication. Even this requires years of experiences as tertiary lecturer writing laboratory manuals for students or explaining the steps to my project students – the importance of unambiguous step-by-step protocol. Therefore, writing the prompt itself required experiences from at least 3 different jobs. This puts me in the role of a software architect or project manager more than a junior developer. It is almost like having to explain the ins and outs to a remote junior developer, which happens to be GitHub Copilot.

More importantly, I had decided ProjEB to be a CLI at this point, which allows me to write out the examples of commands as a form of requirement. If ProjEB is meant to be GUI from the start, I will have no idea how to describe the end result to GitHub Copilot. Nevertheless, when I wrote the requirements for the software project years ago, we spent many days scribbling on the whiteboard and over GUI mock-ups before both developers and testers had copies of the GUI flow. Yet, aesthetically pleasing GUI can be considered as one of the non-functional requirement [45] – how to convey aesthetics to another person via phone?

Phase 2 (generation of initial operational codes) and Phase 3 (generation of initial test codes) are smooth. Phase 4 (iterative testing and revision) took up the most time and potentially most challenging on my patience. It feels like test-driven development [46,47]. I reckoned that I iterated more than a hundred times in this phase. There is rather low technical knowledge required at this phase except knowing which files to update – sometimes it may not be obvious and I have to look at the function names to make a judgement. There are multiple times where the errors are obvious but in the spirit of vibe coding, I let GitHub Copilot work on it for several more iterations before prompting where the error is.

On the whole, my thoughts and learnings before and after my attempt at vibe coding is that it requires the coder to be at a much higher level, such as a software architect or a project manager; while the AI-coding assistant is an entry level developer that is entrusted to code for individual functions. I also felt that iterative testing and revision (Phase 4) could be automated much like machine learning as it is tedious, time-consuming, and even boring.

What do I think of vibe coding as a whole? I think vibe coding is an inevitable progression of IT and computer science, much like automated theorem proving when mathematical logic meets computer science [48]. Yet, there are currently much debates online about the dangerous of vibe coding, especially with security [49] (URL 8-11). This is expected as all advancements come with both positive and negative impacts. In 2023, WHO reported more than 1.19 million fatalities each year as a result of road traffic crashes (URL 12) and yet nobody proposes that we should all go back to walking. I think the reasonable approach is "use with care" rather than "use blindly". Therefore, code review is important [50,51] and LLMs can take the first walkthrough [52].

From my pre-trial experience, AI-coding assistant is a handy assistant to help me locate and contextualize codes that I can use, which is substantially more convenient than sourcing for potentially similar codes to adapt. From my trial experience, vibe coding can be a productivity multiplier but as any multiplier, it can multiples both my strengths and weaknesses. I will illustrate using an analogy – before the advent of bibliographic managers like EndNote and Zotero, authors had painstakingly written out and editing reference sections for various submission. Now, I am using Zotero to help me collate my references. Nobody is thinking that I am cheating in anyway. Yet, I also have students who can mess up their reference sections despite using bibliographic managers because they simply mess up the data stored in their bibliographic managers. The same can be said for spell checkers and other software conveniences. This also reminds me of my days learning dBase IV when one of my fellow course mates commented that "browse is such a powerful command" – how many developers actually know the detailed operations from a print function to the characters displayed on screen? – third generation programming languages onwards are just layers of libraries.

Vibe coding should not and cannot replace fundamental coding skills; instead, it demands software architectural skills, which is likely a skills gap for entry developers. On March 24, 2025; Straits Times (Singapore's main newspaper) published an article on vibe coding titled "Does 'vibe coding' make everyone a programmer?" (URL 13) and 2 issues were raised – skills gap, and job displacement. As a digital immigrant, I believe that skills gap can be bridged by constant exposure and willingness to play with it. The year 2025 marks the start of Generation Beta who are expected to grow up with AI deeply integrated. The same can be said for job displacement – job displacement has been ongoing for centuries. Once upon a time, computer is an occupation [53,54]. With self-checkout services, many cashier positions disappeared as well. We have to move on. I think there are exciting days ahead, maybe choppy but definitely exciting.

Conclusion

Vibe coding can be a productivity multiplier, provided that the vibe can be accurately described. This is not an easy task for a beginner as it took substantial experiences from my past. However, I am excited as its prospects ahead.

List of URLs

URL 1 = https://karpathy.ai/
URL 2 = https://x.com/karpathy/status/1886192184808149383
URL 3 = https://github.com/mauriceling/projeb/
URL 4 = https://www.anthropic.com/news/claude-3-5-sonnet
URL 5 = https://github.com/features/copilot
URL 6 = https://github.com/mauriceling/projeb/releases/tag/SEP11-start
URL 7 = https://github.com/mauriceling/projeb/releases/tag/SEP11-VC
URL 8 = https://weerd.xyz/posts/vibe-coding/
URL 9 = https://nmn.gl/blog/vibe-coding-fantasy
URL 10 = https://checkmarx.com/blog/security-in-vibe-coding/
URL 11 = https://futurescouting.substack.com/p/why-you-shouldnt-use-vibe-coding
URL 12 = https://www.who.int/news-room/fact-sheets/detail/road-traffic-injuries
URL 13 = https://www.straitstimes.com/tech/does-vibe-coding-make-everyone-a-programmer

Conflict of Interest

The author is interested in and intends to use AI-assisted coding.

Bibliography

1. Hartman J., *et al*. "AI-Assisted Coding with Cody: Lessons from Context Retrieval and Evaluation for Code Recommendations". 18th ACM Conference on Recommender Systems (ACM, Bari Italy) (2024): 748-750.

2. Ramler R., *et al*. "Industrial Experience Report on AI-Assisted Coding in Professional Software Development". *Proceedings of the 1st International Workshop on Large Language Models for Code* (ACM, Lisbon Portugal) (2024): 1-7.

3. Martinović B and Rozić R. "Perceived Impact of AI-Based Tooling on Software Development Code Quality". *SN Computer Science* 6.1 (2025): 63.

4. Fan G., *et al*. "The impact of AI-assisted pair programming on student motivation, programming anxiety, collaborative learning, and programming performance: a comparative study with traditional pair programming and individual approaches". *International Journal of STEM Education* 12.1 (2025): 16.

5. Ramírez-Rueda R., *et al*. "Transforming Software Development: A Study on the Integration of Multi-Agent Systems and Large Language Models for Automatic Code Generation". 2024 12th International Conference in Software Engineering Research and Innovation (CONISOFT) (IEEE, Puerto Escondido, Mexico) (2024): 11-20.

6. Thaler SL. "Virtual Input" Phenomena Within the Death of a Simple Pattern Associator". *Neural Networks* 8.1 (1995): 55-65.

7. Farquhar S., *et al*. "Detecting Hallucinations in Large Language Models Using Semantic Entropy". *Nature* 630.8017 (2024): 625-630.

8. Ganesh P., *et al*. "On the Role of Prompt Multiplicity in LLM Hallucination Evaluation". *ICLR 2025 Workshop on Building Trust in Language Models and Applications* (2025).

9. Ji Z., *et al*. "Towards Mitigating LLM Hallucination via Self Reflection". Findings of the Association for Computational Linguistics: EMNLP 2023 (Association for Computational Linguistics, Singapore) (2023): 1827-1843.

10. Ling MH. "ChatGPT (Feb 13 Version) is a Chinese Room". *Novel Research in Sciences* 14.2 (2023): NRS.000832.

11. Chelli M., *et al*. "Hallucination Rates and Reference Accuracy of ChatGPT and Bard for Systematic Reviews: Comparative Analysis". *Journal of Medical Internet Research* 26 (2024): e53164.

12. Searle J. "Chinese Room Argument". *Scholarpedia* 4.8 (2009): 3100.

13. Abdalkareem R., *et al*. "What Do Developers Use the Crowd For? A Study Using Stack Overflow". *IEEE Software* 34.2 (2017): 53-60.

14. Kessel M and Atkinson C. "Code Search Engines for the Next Generation". *Journal of Systems and Software* 215 (2024): 112065.

15. Di Grazia L and Pradel M. "Code Search: A Survey of Techniques for Finding Code". *ACM Computing Surveys* 55.11 (2023): 1-31.

16. Vaswani A., *et al*. "Attention is All You Need". *Advances in Neural Information Processing Systems* 30 (2017).

17. Lo CK. "What Is the Impact of ChatGPT on Education? A Rapid Review of the Literature". *Education Sciences* 13.4 (2023): 410.

18. Lee BC and Chung J. "An Empirical Investigation of the Impact of ChatGPT on Creativity". *Nature Human Behaviour* 8.10 (2024): 1906-1914.

19. Tangsrivimol JA., *et al*. "Benefits, Limits, and Risks of ChatGPT in Medicine". *Frontiers in Artificial Intelligence* 8 (2025): 1518049.

20. Picazo-Sanchez P and Ortiz-Martin L. "Analysing the Impact of ChatGPT in Research". *Applied Intelligence* 54.5 (2024): 4172-4188.

21. A Shaji George., *et al*. "A Review of ChatGPT AI's Impact on Several Business Sectors. *Partners Universal International Innovation Journal* 1.1 (2023): 9-23.

22. Adamopoulou E and Moussiades L. "Chatbots: History, Technology, and Applications". *Machine Learning with Applications* 2 (2020): 100006.

23. Millam A and Bakke C. "Coding with AI as an Assistant: Can AI Generate Concise Computer Code?" *Journal of Information Technology Education: Innovations in Practice* 23 (2024): Article 9.

24. Nah J-H. "Analysis of ChatGPT's Coding Capabilities in Foundational Programming Courses". *Journal of Engineering Education Research* 26.6 (2023): 71-78.

25. Haindl P and Weinberger G. "Does ChatGPT Help Novice Programmers Write Better Code? Results From Static Code Analysis". *IEEE Access* 12 (2024): 114146-114156.

26. Pinto G., *et al*. "Developer Experiences with a Contextualized AI Coding Assistant: Usability, Expectations, and Outcomes". Proceedings of the IEEE/ACM 3rd International Conference on AI Engineering - Software Engineering for AI (ACM, Lisbon Portugal) (2024): 81-91.

27. Rose K., *et al*. "IS Professors' Perspectives on AI-Assisted Programming". *Issues in Information Systems* 24.2 (2023).

28. Knowlton PH. "The Use of an Algebraic Language as Both a Source and Target Language". Proceedings of the 1968 23rd ACM National Conference (ACM Press, Not Known) (1968): 787-794.

29. Antoy S., *et al*. "Specification-Based Code Generation". Twenty-Third Annual Hawaii International Conference on System Sciences (IEEE Comput. Soc. Press, Kailua-Kona, HI, USA), 2 (1990): 165-173.

30. Domı´nguez E., *et al*. "A Systematic Review of Code Generation Proposals from State Machine Specifications". *Information and Software Technology* 54.10 (2012): 1045-1066.

31. Thompson FB. "English for the Computer". Proceedings of the November 7-10, 1966, Fall Joint Computer Conference on XX - AFIPS '66 (Fall) (ACM Press, San Francisco, California) (1966): 349.

32. Sammet JE. "The Use of English as a Programming Language". *Communications of the ACM* 9.3 (1966): 228-230.

33. Thompson FB., *et al*. "REL: A Rapidly Extensible Language system". Proceedings of the 1969 24th National Conference On - (ACM Press, Not Known) (1969): 399-417.

34. Chamberlin DD and Boyce RF. "SEQUEL: A Structured English Query Language". *Proceedings of the 1976 ACM SIGFIDET (Now SIGMOD) Workshop on Data Description, Access and Control - FIDET '76* (ACM Press, Not Known) (1976): 249-264.

35. Badiola KA., *et al*. "Experiences with a Researcher-Centric ELN". *Chemical Science* 6.3 (2015): 1614-1629.

36. Ng YY and Ling MH. "Electronic laboratory notebook on Web-2Py framework". *The Python Papers* 5.3 (2010): 7.

37. Finnie-Ansley J., *et al*. "The Robots Are Coming: Exploring the Implications of OpenAI Codex on Introductory Programming". Proceedings of the 24th Australasian Computing Education Conference (ACM, Virtual Event Australia) (2022): 10-19.

38. Bhattacharyya M., *et al*. "High Rates of Fabricated and Inaccurate References in ChatGPT-Generated Medical Content". *Cureus* 15.5 (2023): e39238.

39. Walters WH and Wilder EI. "Fabrication and Errors in the Bibliographic Citations Generated by ChatGPT". *Scientific Reports* 13.1 (2023): 14045.

40. Lechien JR., *et al*. "Accuracy of ChatGPT-3.5 and -4 in Providing Scientific References in Otolaryngology-Head and Neck Surgery". *European Archives of Oto-rhino-laryngology* 281.4 (2024): 2159-2165.

41. Ruta MR., *et al*. "ChatGPT for Univariate Statistics: Validation of AI-Assisted Data Analysis in Healthcare Research". *Journal of Medical Internet Research* 27 (2025): e63550.

42. Prandner D., *et al*. "ChatGPT as a Data Analyst: An Exploratory Study on AI-Supported Quantitative Data Analysis in Empirical Research". *Frontiers in Education* 9 (2025): 1417900.

43. Rasheed A., *et al*. "Requirement Engineering Challenges in Agile Software Development". *Mathematical Problems in Engineering* 2021 (2021): 1-18.

44. Hussain A., *et al*. "The Role of Requirements in the Success or Failure of Software Projects". *International Review of Management and Marketing* 6.7 (2016): 306-311.

45. Liang JT., *et al*. "A Large-Scale Survey on the Usability of AI Programming Assistants: Successes and Challenges". 2024 IEEE/ACM 46th International Conference on Software Engineering (ICSE 2024) (ACM Press, Lisbon, Portugal) (2024).

46. Beck K. "Test Driven Development: By Example (Addison-Wesley Educational Publishers Inc)". (2002).

47. George B and Williams L. "A Structured Experiment of Test-Driven Development. *Information and Software Technology* 46.5 (2004): 337-342.

48. Loveland DW. "Automated Theorem Proving: Mapping Logic into AI". *Proceedings of the ACM SIGART International Symposium on Methodologies for Intelligent Systems*, ISMIS '86. (Association for Computing Machinery, New York, NY, USA) (1986): 214-229.

49. Negri-Ribalta C., *et al*. "A Systematic Literature Review on the Impact of AI Models on the Security of Code Generation". *Frontiers in Big Data* 7 (2024): 1386720.

50. Nazir S., *et al*. "Modern Code Review Benefits - Primary Findings of A Systematic Literature Review". Proceedings of the 3rd International Conference on Software Engineering and Information Management (ACM, Sydney NSW Australia) (2020): 210-215.

51. Braz L and Bacchelli A. "Software Security During Modern Code Review: The Developer's Perspective". Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ACM, Singapore Singapore) (2022): 810-821.

52. Wadhwa N., *et al*. "CORE: Resolving Code Quality Issues using LLMs". *Proceedings of the ACM on Software Engineering* 1.FSE (2024): 789-811.

53. Light JS. "When Computers Were Women". *Technology and Culture* 40.3 (1999): 455-483.

54. Ceruzzi PE. "When Computers Were Human". *IEEE Annals of the History of Computing* 13.3 (1991): 237-244.