Friction, Flow, and the Potential Deskilling Effect of Vibe Coding

Leslye Denisse Dias Duran[1,*]

[1]*Independent Researcher*

Abstract

The emerging practice of "vibe coding," where developers rely on AI-generated code based on intuitive or aesthetic judgments rather than rigorous technical understanding raises a series of ethical concerns in regards to risks of security vulnerabilities, technical debt, and unclear responsibility allocation. Beyond these issues, the paper considers the potential for both technical and moral deskilling, as reliance on AI may erode programming expertise and diminish developers' ethical agency. To address these risks, the paper explores the introduction of purposeful friction in development workflows, understood as interventions that seek to foster human agency and critical engagement. Ultimately, this paper proposes that purposeful friction might have the potential to balance the advantages of vibe coding with the need to preserve responsibility, skill formation, and sustainable development practices.

Keywords

Deskilling, vibe coding, purposeful friction, ethical agency

1. Introduction

Over the past few years, AI-assisted programming has shifted from experimental novelty to everyday practice. Tools like GitHub Copilot that offer explanations and generate software code that can be readily integrated in software projects are being integrated into development workflows at an increasing pace. This transformation is driven by the promises of increased productivity and lowered barriers to entry [1], which could be interpreted as another attempt at democratizing the work of software developers and potentially opening the doors to individuals from less technologically advantaged contexts [2]. However, this also introduces new cultural and ethical dynamics into how software is made. One recent phenomenon within this landscape is what is informally called "vibe coding" [3] —a programming practice where AI-generated suggestions are blindly accepted or rejected based on intuition and aesthetic fit rather than on rigorous planning of how the development of a model should be and understanding of the frameworks or the tech stack used to build a robust product.

Broadly speaking, vibe coding defined the use of an AI assistant to generate code while the person guides the process with ideas and descriptions in the form of natural language prompts. It distinguishes itself from earlier forms of AI-assisted coding in that in the latter, the AI assistant generates suggestions or blocks of code for the task at hand, and it might help identifying bugs and suggesting fixes, but the programmer needs to ultimately revise and accepts the suggestions and is the one actively debugging the code. On the other hand, in vibe coding, if there is an error, the coder is meant to merely copy and paste the error as it is generated back to the AI assistant for resolution, and any corrections should be accepted without checking.

While this approach has been justified as allowing the coder to express creativity in a way that formal programming does not, it also introduces a further level of ethical and technical detachment in the field that raises ethical challenges like security vulnerabilities in commercially deployed products, technical debt, diffused allocation of responsibility, and the potential of technical and moral deskilling

HHAI-WS 2025: Workshops at the Fourth International Conference on Hybrid Human-Artificial Intelligence (HHAI), June 9–13, 2025, Pisa, Italy

*Corresponding author.

lessdias@gmail.com (L. D. Dias Duran)

0000-0001-7520-1323 (L. D. Dias Duran)

of programmers that emerged with the introduction of generative AI. This paper explores these ethical concerns, and it seeks to argue that the risk of technical and ethical deskilling from vibe coding could be mitigated with the implementation of purposeful friction.

2. Ethical Implications of Vibe Coding

Digital environments are often designed for seamlessness and flow. A general argument in favor of AI-assisted programming is the ability of the tools to anticipate the needs of the developer, provide code snippets that can be easily integrated into the code base, and enable the assembly of working prototypes with minimal technical effort. The emergence vibe coding, on the one hand, follows the known focus on providing seamless user experiences and quick results, and is viewed in a largely positive light in corporate and entrepreneurial spaces [4]. In this sense, the practice of vibe coding has introduced an approach to software development that hinges on a stronger notion of "frictionless-ness", that raises questions about security vulnerabilities, technical debt, allocation of responsibility and what types of technical and relational programming knowledge might be at risk of being eroded. On the other hand, however, vibe coding also embodies a cultural shift in perception about the possibilities of human-AI interaction that embraces the idea of an AI-assistant as a collaborator. From this view, "vibing" is a form of harmonizing with the AI coding assistant and as put by [5], it suggests a transition from deterministic to probabilistic intent mediation.

From a purely technical standpoint, vibe coding became possible because AI assistants have reached a level of technical sophistication that allows people without programming expertise to describe and iterate ideas either through written prompts or speech recognition tools. Although initially, the primary usage of vibe coding circulated around developing quick prototypes and kick-starting personal projects, in the last months the rise of AI development platforms that employ vibe coding suggests that companies, startups and programmers are leaning towards using vibe coding to launch products to the market [1]. The why or how the generated code works loses relevance in the face of faster results and a lower entry-barrier for non-programmers. However, vibe coding is also the consequence of deliberate choices by field experts of how the process of developing software should be and about the future of the technology industry. In a world that has embraced the implementation of AI in many areas of human activity, and in particular with the rush of established and new generative AI models to take the lead in the race almost every week [6], it is not surprising that the alleged advantage of speed, simplicity and lower costs provided by vibe coding compared to traditional programming is received with largely open arms.

However, there are epistemic and ethical costs associated with these goals [7]. One of the most prominent is security vulnerabilities. They are a major ethical problem because, by definition, vibe coding requires the coder to accept the AI-generated suggestions without looking into what these suggestions are, how they relate with other parts of the code, and what issues they might bring when they are deployed into the real world. An illustration of this issue occurred in early 2025 when an AI-driven development platform that embraced the vibe coding approach called *Lovable* faced criticism when it was reported that there were critical vulnerabilities in their implementation of Row Level Security (RLS) policies that exposed sensitive user data and allowed malicious code injection to approximately 170 web applications created using their services [8]. While robust oversight of which security measures are being built into a product's code is standard practice in traditional programming and even in most instances of AI-assisted programming, vibe coding does not require —or even encourage— that the coder is attentive and knowledgeable in how to spot and fix potential errors, thus leaving open potential weaknesses that can later be exploited by malicious agents.

Closely interconnected, a second ethical implication of vibe coding is technical debt. This a well known phenomenon in software engineering and it is broadly defined as the accumulation of consequences of decisions made during development, implementation or maintenance that prioritized immediate

[1]Consider the emergence of tools like Lovable, Cursor, Replit and more recently, Google's Opal

goals over quality [2]. It has been argued that technical debt can negatively affect productivity, long-term product sustainability and team morale, and it is also associated with real monetary consequences [9]. Similarly to the issue with security vulnerabilities, the very essence of vibe coding heightens the potential of accumulating technical debt, that could make the code difficult to maintain or debug, negatively impact the efficiency of the product or application, and due to the lack of structure and documentation standards, it could make collaboration between developers challenging or even unfeasible.

The third ethical issue is diffused responsibility allocation within technology organizations. It has been established that we are in the middle of a Fourth Industrial Revolution characterized by the radical transformation of virtually all of human activity spheres by the convergence of technologies like AI, robotics and gene editing [10]. This transformation has placed developers (and other technical professionals in the broad field of information technologies) in positions of significant epistemic authority whose choices have a wide reach of impact. However, the organizational structures of technology companies and the way the development work usually is carried out —in segments or modules— are not specifically designed to establish clear channels of responsibility. Vibe coding exacerbates this existing difficulty by making even more complex to determine who is in a position to hold responsibility for key decisions that impact at the point of deployment.

3. The Deskilling Potential of Vibe Coding

Another, more subtle risk is the potential of vibe coding to lead to a gradual deskilling of developers. This phenomenon, as argued elsewhere [11] is not well-understood and remains difficult to evaluate empirically as shifts in skills can respond to a great variety of factors and do not follow linear paths, i.e., they may not just decrease or increase continuously, but adapt into a new, similar skill or gradually lose its market value until a generational change leaves no one to pass the skill along to the new one. Although there is a dire lack of empirical work on this topic, the phenomenon of deskilling has been discussed at length in academia since the industrial revolution [12]. It is undeniable that the emergence of digital technologies that are being built with the intention of automating human action brings forth the possibility of losing the skills that are and will be automated and as such, it should be further discussed in the field of AI ethics.

Broadly construed, deskilling is the situation within a certain occupation in which the workers experience reduced "discretion, autonomy, decision-making quality, and knowledge as they perform their jobs. All of which negatively affects their ability to perform these jobs in a manner they see fit" [13]. I argue that deskilling can occur in two main ways: technical and moral. Technical deskilling can be defined as the reduction or decrease of existing technical skills in a determined field while moral deskilling refers to the reduction or decrease of moral skills required to take action or make decisions that have moral significance.

Vibe coding risks to diminish the value of the technical judgment employed by developers during the ideation and development phases of a pipeline and the critical evaluation of the potential outcomes of decisions regarding the code's architecture and functioning once deployed to the consumers. Instead of cultivating a deep understanding of code structures and system behaviors, users may come to rely solely on affective cues, i.e., what "looks right," "sounds right," or "feels like it works." This implies that much of the cognitive value of software development is being replaced by simple prompt-based interactions and automated model inference, thus a potential downside of vibe coding is the loss of the skill formation that comes with grappling with the "material resistances inherent in code's syntax structure and debugging challenges in a manual workflow" [14].

In regard to moral deskilling, vibe coding reinforces a long-standing concern in human-computer interaction about the automation of expertise [15], [16] about the shift of the roles of humans in these new relationships. In this context, the practice of vibe coding introduces a subtle redistribution of agency

[2]It should be noted that there are classifications of different types of technical debt and an in-depth exploration of the implications in each case is outside of the scope of this article. Suffice to say that the notion of technical debt is taken in its most general conceptualization

from the human programmer to the automated model: while the machine generates and suggests, the human is —in a way— reduced to accepting or curating the generated content. The cognitive value of creating the code, which requires an array of mental processes and accumulated expertise, is at the very least diminished. Furthermore, from an ethical perspective, this implies a shift in the locus of decision-making authority that goes from understanding the content of the code and actively engaging in making decisions about the path of development of the application, to a more passive role of accepting outputs and iterating if the result is not satisfactory, often without full transparency about how the outputs are produced.

As Griffin and colleagues have argued, the matter of the moral agency of programmers is an often-overlooked aspect of software development [17]. While in many other fields it is required of professionals to comply with a deontological code and they can be held accountable for the consequences of certain decisions —such as in healthcare, for instance—, the conversations about what it means for developers to behave ethically or to nurture their own ethical agency are not occurring [17]. They found in their qualitative study that in many instances, the developers do not think about themselves as moral agents, and similarly, they do not see their profession as intrinsically ethical. However, the implementation of AI tools is steadily permeating all areas of human action, and this is conferring unparalleled authority to the developers who make choices throughout the developmental pipeline. As such, it has become necessary to address developers as important moral agents whose choices can have far-reaching consequences.

Considering this argument, the redistribution of agency mentioned before could impact on the development of moral skills of developers, thus leading —perhaps not yet necessarily to deskilling, as it could be argued that there is no basis to determine existing moral skill— but to an inhibition of moral skilling. This means that, in particular junior developers who have less experience and time in the profession, it would be even less encouraged to nurture more reflective or critical forms of engagement with the technological possibilities and outcomes of the application they are tasked with developing.

4. Purposeful Friction in Development

In response to this phenomenon, introducing purposeful friction into development workflows could help mitigate the unintended deskilling or skilling inhibition introduced by vibe coding. These practices, defined as "programmed inefficiencies" [18] or "acts of inconvenience" [19] have been conceptualized as features or steps introduced in systems to keep the human agency in the loop. The objective of such steps is then to reinforce or invite cognitive engagement from developers, at the very least at critical stages of decision-making in which choices made can have an impact on certain fundamental interests of customers (i.e., their rights) or for the long-term sustainability of the product being developed.

There are known examples of positive friction in design, for instance, adding extra confirmation steps when one is in the process of deleting an account. Although this paper will not discuss specific strategies, there are at least three forms of purposeful friction that could help vibe coders to critically engage with their code without entirely losing the advantages it offers. These are general approaches distinguished by the moment in which they are introduced: before, after and while coding [3].

First, before a coding session, a frictional intervention is to establish which tasks should be carried out using vibe coding and which should be left to traditional programming. For instance, it could be determined that any tasks related to the storage or use of customer data for which security guardrails are necessary should never be programmed using solely vibe code. Second, after the coding session, the coder could prompt the AI-assistant to generate a detailed summary of the session and the code generated that can be employed as documentation or learning tool. This suggestion could address the concern about the technical deskilling of junior developers by harnessing the speed and functionality of AI-assistants to help these developers to acquire experience and learn about the code that was generated. This suggestions, however, assumes that the developers would find desirable to reduce the epistemic asymmetry between themselves and the AI-assistant, i.e., become developers rather than

[3]It should be noted that it is assumed that vibe coding is being used for product development and not for prototyping or for private projects without reach. In these cases, vibe coding do not raises the concerns discussed throughout this paper

just 'coders'. While vibe coding seeks to establish a more conversational form of human-machine interaction, this does not exclude that material engagement with and knowledge of the code still have significant advantages that cannot be substituted by vibe coding.

Third, during the coding process, there could be frictional interventions that introduce deliberate slowness such as having pauses to confirm or accept certain complex or critical modules of code, multiple options being presented to the coder, prompting them to read and actively choose the option that best suits the task. The aim of this kind of friction is to move beyond the question of how something works to the question of why and for whom a developer is building a product in the first place. It is through this engagement that the promise of more democratized access to programming might be better realized, while at the same time addressing critical trade-offs.

Introducing purposeful friction in development workflows where vibe coding is employed is not without its challenges. It is necessary, for instance, to distinguish between positive and negative friction. Although this paper challenges the assumption that seamlessness in development is always desirable and should be a default practice, this does not preclude the fact that there *are* types of friction that can negatively impact the developer's work and the customer's experience with a product. Similarly, it should be noted that development workflows can vary radically in terms of scope, aims and particular constraints and such factors should be taken into consideration when choosing which types of purposeful friction interventions are implemented. In this sense, the very act of introducing friction is an exercise of reflection about the direction and structure of a particular development workflow.

5. Conclusion

This paper aims to explore the ethical concerns raised by the emergence and adoption of the practice of vibe coding in software development. It contends that while the alleged advantages offered by this approach in terms of prototyping speed and lowered entry-barrier for people with little to no programming expertise can be used in certain scenarios, ultimately, vibe coding raises serious ethical questions about the future of software development. These concerns are rooted in the fact that consumers are increasingly using AI-assisted and vibe coding platforms. This creates a series of market incentives and pressures that make vibe coding an attractive —or even necessary—practice for both experienced developers who need to keep up with the fast pace of the development industry as well as junior developers without experience who want to enter it.

The ethical concerns raised are of technical and moral nature. On the one hand, security vulnerabilities and technical debt as a result of poorly implemented and brittle applications could not only jeopardize the privacy and digital integrity of consumers, but also the long-term sustainability of the products themselves, and thus, could impact in the financial health of the companies that developed them. On the other hand, the further obscuring of responsibility channels as well as the re-distribution of epistemic and moral agency, point at more subtle and long-term consequences of practices like vibe coding, that could deskill or inhibit the skilling of professionals in this field not only regarding skills that are required for the technical robustness of the applications but also the moral skills that are necessary for decisions to prioritize goals like securing the rights of people using the products, the environmental costs of developing certain types of software and the long term consequences.

The introduction of purposeful friction can be seem -and perhaps rightly so- as a contradiction to the very essence of what vibe coding stands for. However, the benefits in terms of lower entry barrier and bolstered efficiency of vibe coding do not need to be entirely sacrificed. This paper thus sought to argue that there can and should be a discussion about finding common ground between employing vibe coding and the critical importance of establishing clearer channels of attribution and distribution of responsibility in development settings. After all, given that developers have been granted significant levels of decision-making power in high stake environments, we should at the very least consider that this authority may require stricter standards of professionalization. Ultimately, this paper aims to emphasize the importance of maintaining developer agency in AI-mediated environments from ethical and technical perspectives, and to propose the introduction of purposeful friction as a feasible way

to promote forms of cognitive and moral engagement that enable more meaningful configurations of human-machine interaction in which phenomena like deskilling is not an inevitable consequence, but where reskilling and upskilling are also within the horizon of possibility.

Declaration on Generative AI

The author have not employed any Generative AI tools.

References

[1] A. M. Dakhel, V. Majdinasab, A. Nikanjam, F. Khomh, M. C. Desmarais, Z. M. J. Jiang, Github copilot ai pair programmer: Asset or liability?, Journal of Systems and Software 203 (2023) 111734. doi:10.1016/j.jss.2023.111734.

[2] L. Sundberg, J. Holmström, Democratizing artificial intelligence: How no-code ai can leverage machine learning operations, Business Horizons 66 (2023) 777–788. doi:10.1016/j.bushor.2023.04.003.

[3] A. Karpathy, 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., X (originally Twitter), post, 2025. Accessed via https://x.com/karpathy/status/1886192184808149383.

[4] K. Samuel-Adebayo, How vibe coding is creating a new ai economy, Forbes (online), 2025. Accessed via Forbes: https://www.forbes.com/sites/kolawolesamueladebayo/2025/08/04/how-vibe-coding-is-creating-a-new-ai-economy/.

[5] C. Meske, T. Hermanns, E. von der Weiden, K.-U. Loser, T. Berger, Vibe coding as a reconfiguration of intent mediation in software development: Definition, implications, and research agenda, 2025. ArXiv preprint arXiv:2507.21928, 2025.

[6] N. Jones, AI race in 2025 is tighter than ever before, Technical Report, Nature (online news), 2025.

[7] J. Kemper, S. Jankowski, Silicon valley's frictionless future: The design philosophy of frictionlessness, in: The De Gruyter Handbook of Automated Futures: Imaginaries, Interactions and Impact, volume 2, De Gruyter, 2024, pp. 287–304.

[8] R. Albergotti, The hottest new vibe coding startup may be a sitting duck for hackers, Technical Report, Semafor (online news), 2025.

[9] E. Tom, A. Aurum, R. Vidgen, An exploration of technical debt, Journal of Systems and Software 86 (2013) 1498–1516. doi:10.1016/j.jss.2012.12.052.

[10] K. Schwab, The fourth industrial revolution: what it means, how to respond, in: Handbook of research on strategic leadership in the Fourth Industrial Revolution, Edward Elgar Publishing, 2024, pp. 29–34.

[11] L. D. Dias Duran, Deskilling of medical professionals: an unintended consequence of ai implementation?, Giornale di filosofia 2 (2021). doi:DOI10.7413/1827-5834014.

[12] H. Braverman, Labor and monopoly capital: The degradation of work in the twentieth century, NYU Press, 1998.

[13] T. Hoff, Deskilling and adaptation among primary care physicians using two work innovations, Health Care Management Review 36 (2011) 338–348. doi:10.1097/HMR.0b013e31820e1051.

[14] A. Sarkar, I. Drosos, Vibe coding: Programming through conversation with artificial intelligence, 2025. doi:10.48550/arXiv.2506.23253.

[15] C. P. Janssen, S. F. Donker, D. P. Brumby, A. L. Kun, History and future of human-automation interaction, International Journal of Human-Computer Studies 131 (2019) 99–107. doi:10.1016/j.ijhcs.2019.05.006.

[16] S. Vallor, Moral deskilling and upskilling in a new machine age: Reflections on the ambiguous future of character, Philosophy & Technology 28 (2015) 107–124. doi:10.1007/s13347-014-0156-9.

[17] T. A. Griffin, B. P. Green, J. V. M. Welie, The ethical agency of ai developers, AI and Ethics 4 (2024) 179–188. doi:10.1007/s43681-022-00256-3.

[18] F. Cabitza, A. Campagner, D. Ciucci, A. Seveso, Programmed inefficiencies in dss-supported human decision making, in: P. Editors (Ed.), Proceedings of the 21st International Conference on Human-Computer Interaction, volume 11506 of *HCII '19*, Springer-Verlag, Berlin, Heidelberg, 2019, pp. 217–229. doi:10.1007/978-3-030-26773-5_18.

[19] R. R. Gosline, Why AI Customer Journeys Need More Friction, Technical Report, Harvard Business Review (online news), 2022.