

Rewiring the Brain's Reactive Patterns Through Vibe Coding: A Neuroplastic Way

Eranga Bandara^a, Ross Gore^a, Asanga Gunaratna^b, Ravi Mukkamala^a,
Nihal Siriwardanagea¹, Gihan Siriwardanageaⁱ, Sachini Rajapakse^e,
Isurunima Kularathna^e, Pramoda Karunarathna^e, Chalani Rajapakse^e,
Sachin Shetty^a, Christopher K. Rhea^a, Ng Wee Keong^c, Kasun De Zoysa^d,
Amin Hass^f, Shaifali Kaushik^l, Wathsala Herath^g, Preston Samuel^k, Anita
H. Clayton^j, Atmaram Yarlagadda¹

^a*Old Dominion University, Norfolk, VA, USA*

^b*AI Motion Labs, Melbourne, Australia*

^c*Nanyang Technological University, Singapore*

^d*University of Colombo, Sri Lanka*

^e*IcicleLabs.AI*

^f*Accenture Technology Labs, Arlington, VA, USA*

^g*Agentsway.AI*

^h*GSI Scandinavia AB*

ⁱ*Lithuanian University of Health Sciences*

^j*Department of Psychiatry and Neurobehavioral Sciences,
University of Virginia School of Medicine, Charlottesville, VA, USA*

^k*Blanchfield Army Community Hospital, Fort Campbell, KY, USA*

^l*McDonald Army Health Center, Newport News, VA, USA*

Email addresses: cmedawer@odu.edu (Eranga Bandara), rgore@odu.edu (Ross Gore), asanga.gunaratna@complianceoslab.app (Asanga Gunaratna), rmukkama@odu.edu (Ravi Mukkamala), nihal@gsi2.com (Nihal Siriwardanagea), gihan.siriwardanagea@stud.lsmu.lt (Gihan Siriwardanagea), sachini.rajapakse@iciclelabs.ai (Sachini Rajapakse), isurunima.kularathna@iciclelabs.ai (Isurunima Kularathna), pramoda.karunarathna@iciclelabs.ai (Pramoda Karunarathna), chalani.rajapakse@iciclelabs.ai (Chalani Rajapakse), sshetty@odu.edu (Sachin Shetty), crhea@odu.edu (Christopher K. Rhea), awkng@ntu.edu.sg (Ng Wee Keong), kasun@ucsc.cmb.ac.lk (Kasun De Zoysa), amin.hassanzadeh@accenture.com (Amin Hass), Skaus2000@gmail.com (Shaifali Kaushik), wathsala.herath@agentsway.ai (Wathsala Herath), preston.1.samuel.mil@health.mil (Preston Samuel), AHC8V@uvahealth.org (Anita H. Clayton), atmaram.yarlagadda.civ@health.mil (Atmaram Yarlagadda)

Abstract

Vibe coding—building software by describing it in natural language and letting an AI generate the code, “forgetting that the code even exists”—has, within a year of being named, become a dominant mode of software development. We observe that vibe coding proceeds through a tight, high-frequency loop of prompting, evaluating, and re-prompting, and that each turn is a *contact event* at which a conditioned reactive pattern may fire before deliberate appraisal. Under the principles of activity-dependent synaptic plasticity, each uninterrupted cycle strengthens the underlying pathway (long-term potentiation), so that vibe coding may quietly condition the very reactive patterns it repeatedly provokes—a profile made distinctive by the helplessness of directing code one does not understand, the unsettling of a craft identity built on comprehension, and the anxiety of shipping software one cannot vouch for. We then propose that the same loop can be engaged to *rewire* these patterns rather than entrench them. The method is *behind-the-scenes observation*: at the brief feeling-tone gap, instead of completing the reaction, the developer observes the reactive process in three stages—noticing the signal, noticing the feeling tone, and seeing the conditioned pathway as it fires. This is hypothesised to have a dual, neuroplastic effect: the old reactive pathway, activated but not completed, weakens (long-term depression), while a new observational pathway is strengthened (long-term potentiation), so that the developer’s default response to a failing result shifts, over repetition, from reaction toward observation. Vibe coding’s contact events—frequent, immediate, and low in stakes—make it unusually well suited to this practice, and because the patterns it evokes operate throughout a person’s life, what is rewired at the keyboard transfers beyond it. We develop the account through worked examples and draw out implications for developer wellbeing and for the design of AI coding tools.

Keywords: Agentic AI, AI Agents, Vibe-Coding, Neuroplasticity, Neuroscience

1. Introduction

In February 2025, the term *vibe coding* was coined to describe a new way of building software: rather than writing code line by line, a developer describes what they want in natural language and lets a large language model generate the code, “fully giv[ing] in to the vibes” and, in the original formulation,

forgetting that the code even exists [1]. Within a year the practice had spread widely enough to be named Collins Dictionary’s Word of the Year for 2025 [2, 3], and it now describes a dominant mode of interaction with AI coding assistants, from prototyping to, increasingly, production software [4, 5]. In vibe coding, the developer’s role shifts from authoring code to *guiding, judging, and re-prompting*: a request is issued, code is generated, the result is evaluated—often by whether it simply runs—and the request is revised.

We make a simple observation about this workflow, with consequences that reach beyond productivity. The vibe-coding workflow is an iterative loop, and each turn of it is a *contact event*: a moment at which a generated result meets the developer and a conditioned response may be activated before any deliberate appraisal has occurred [6, 7]. Because vibe coding proceeds through a dense, high-frequency stream of such contact events—many within a single session, many sessions within a day—it is a setting in which reactive patterns are repeatedly conditioned. When generated code fails to compile, misbehaves at runtime, or is simply not what was wanted, the response that arises is frequently not neutral: impatience, frustration, and the urge to seize control of the tool are repeatedly evoked, and each of these is the activation of a conditioned reactive pattern. By a *reactive pattern* we mean a conditioned, automatic response—a flash of frustration, an urge to control, a sinking of helplessness—that fires in reaction to a triggering cue before deliberate thought, and which, on the account we develop, is not merely a fleeting mood but a physical neural pathway that strengthens or weakens with use. Under the well-established principles of activity-dependent synaptic plasticity, every uninterrupted repetition of such a pattern strengthens the underlying pathway through long-term potentiation [8, 9, 10, 11]. Vibe coding, on this view, may quietly train the very reactive patterns it repeatedly provokes.

Vibe coding also carries a reactive profile distinct from other forms of AI interaction, arising directly from its defining feature. Because the developer often does not read or fully understand the generated code—this is precisely the “forget the code exists” premise—the loop introduces triggers that a more transparent workflow does not. There is the helplessness of debugging code one did not write and cannot readily follow; the loss of mastery for a practitioner whose craft was, until recently, comprehension itself; and the verification anxiety of shipping software one cannot fully vouch for, a concern sharpened by evidence that AI-assisted code frequently contains security-relevant defects [12]. These make vibe coding an unusually

charged conditioning setting—one in which the acceleration of output and the entrenchment of reactive patterns may advance together.

The same loop, however, admits the opposite possibility. Deeply encoded reactive patterns can be attenuated not by managing their consequences downstream but by intervening upstream—at the brief, pre-cognitive window between the arising of a feeling tone and the completion of the reactive response. We call this window the *feeling-tone gap*, and the practice that engages it *behind-the-scenes observation*: in place of the reactive re-prompt, the developer watches the underlying process operate—the signal arriving, the feeling tone it evokes, and the conditioned pathway beginning to fire—so that the cascade does not complete. When a pathway is activated but not completed, the conditions for long-term depression (LTD) are met, and the pathway weakens rather than strengthens [10, 11]. Repeated, this is hypothesized not only to weaken the reactive pathway but to strengthen an alternative, observational response in its place, so that the developer’s default reaction to a failing result shifts, over time, from frustration toward observation.

Two features of vibe coding make it particularly well suited to this practice. First, the loop is *frequent and immediate*: the interval between stimulus (a result appearing, an error surfacing) and response (the reactive re-prompt) is short and clearly bounded, so the feeling-tone window is comparatively easy to locate. Second, the stakes of any single iteration are typically *low*: a failed generation is a safe, repeatable, and inconsequential stimulus, which makes ordinary vibe coding a gentler training context than the high-stakes situations in which reactive patterns are usually encountered. The reactive patterns that surface at the keyboard are, moreover, the same conditioned patterns that operate elsewhere in a person’s life; vibe coding simply makes them observable at high frequency and low cost.

This paper makes three contributions. First, we characterise the vibe-coding workflow as a contact-event cycle and argue that it is a previously unexamined site of neuroplastic conditioning, one that can entrench or attenuate reactive patterns depending on how the loop is engaged. Second, we identify the distinctive reactive profile of vibe coding—in particular the triggers of lost comprehension, lost mastery, and verification anxiety—that sets it apart from other AI interaction [13]. Third, we describe how behind-the-scenes observation, performed in three stages at the feeling-tone gap, can turn the same loop from a driver of entrenchment into an occasion for its dissolution, and we draw out implications for developer wellbeing and for the

design of AI coding tools.

The remainder of this paper is organised as follows. Section 2 reviews related work on AI-assisted development, affect in programming, and the neuroscience of reactive conditioning and its regulation. Section 3 sets out the theoretical foundations—the reactive-pattern model, the feeling-tone gap, and the opposing mechanisms of long-term potentiation and depression—using everyday examples. Section 4 then characterises the vibe-coding loop and its distinctive reactive profile. Section 5 presents the practice of behind-the-scenes observation as applied to vibe coding. Section 6 develops worked examples from real vibe-coding sessions. Section 7 discusses implications for wellbeing and tool design, and Section 8 concludes.

2. Related Work

This work connects three literatures that have not previously been brought together: the study of AI-assisted software development, research on affect and wellbeing in programming, and the neuroscience of reactive conditioning and its regulation. We review each and then position the present contribution.

2.1. AI-Assisted Development and Vibe Coding

The generation of code by large language models, established by systems trained on large code corpora [4], has rapidly moved from autocompletion to the wholesale generation of programs from natural-language description. Empirical studies of these tools have examined their effect on developer productivity [14], the usability gap between what developers expect of code-generation tools and what they experience [15], and the ways programmers actually interact with code-generating models—alternately accelerating through familiar tasks and pausing to validate unfamiliar output [16]. The specific practice of *vibe coding*, in which the developer describes intent and defers authorship of the code itself to the model [1, 2], intensifies this shift. A recurring concern in this literature is quality and security: AI-assisted code has been found to contain security-relevant defects at elevated rates, and developers using such assistants may write less secure code while believing the opposite [12].

This body of work establishes that AI-assisted development reliably produces a fast, iterative loop of generation and evaluation, and that this loop carries real friction and risk. What it does not consider is that the affective

responses evoked within the loop might have consequences for the developer beyond the immediate task—that the repeated frustration of a failing generation could, through ordinary plasticity, leave a lasting trace.

2.2. Affect and Wellbeing in Software Development

That emotion is central to software work, and consequential for it, is well established. Developers’ affective states have been shown to track with their productivity and the quality of their work, with unhappiness carrying measurable costs [17], and frustration in particular has been studied as a common and disruptive state in the development process [18]. This literature treats developer affect as a first-class concern, worthy of study in its own right and consequential for outcomes.

The present work shares that premise but extends it in a specific direction. Where this literature characterises developer affect and its immediate correlates, we ask what the *repetition* of a reactive response, across the many contact events of a vibe-coding session, does to the developer over time—treating the evoked reaction not only as a state with task consequences but as an instance of neuroplastic conditioning.

2.3. Reactive Conditioning and Observation-Based Regulation

The mechanistic basis of our account is activity-dependent synaptic plasticity—the strengthening (long-term potentiation) and weakening (long-term depression) of neural pathways with use and disuse [8, 9, 10, 11]—together with the rapid, pre-cognitive amygdala circuitry through which conditioned reactions are triggered [6, 7]. The practice we propose belongs to a family of observation-based approaches to emotion regulation with substantial clinical support: mindfulness-based cognitive therapy, which cultivates a decentred relationship to thoughts and feelings [19, 20]; metacognitive therapy, which redirects attention to the process of thinking rather than its content [21]; and the broader account of prefrontally mediated emotion regulation [22, 23]. These approaches are typically cultivated in dedicated settings—meditation, therapy, structured exercises—rather than within an activity a person already performs many times a day.

2.4. Positioning

The contribution of this paper lies in connecting these three literatures. The AI-development literature establishes that vibe coding produces a fast,

friction-laden loop but treats the resulting affect as task-local; the software-wellbeing literature establishes that developer affect matters but does not consider its long-term neuroplastic conditioning; and the neuroscience-and-regulation literature establishes that observation can intercept reactive responses but situates the practice outside ordinary technology use. Placing these together yields the paper’s claim: that the vibe-coding loop is a high-frequency site of neuroplastic conditioning, with a reactive profile distinctive to it, at which the same observation-based mechanism studied in clinical settings can be practised—for entrenchment or for attenuation depending on how the loop is engaged. Table 1 summarises this positioning across the dimensions that define the present framework; each strand of prior work supports some of these dimensions but none combines them.

Table 1: Comparison with representative related work across the dimensions of the proposed framework. ✓ = supported; ✗ = not supported; ~ = partially supported.

Work	AI-assisted coding focus	Interaction loop as conditioning site	Neuroplastic framing (LTP/LTD)	Feeling-tone gap (upstream)	Three-stage observation	Vibe-coding reactive profile
Chen et al. [4]	✓	✗	✗	✗	✗	✗
Barke et al. [16]	✓	✗	✗	✗	✗	~
Vaithilingam et al. [15]	✓	✗	✗	✗	✗	~
Perry et al. [12]	✓	✗	✗	✗	✗	✗
Graziotin et al. [17]	✗	✗	✗	✗	✗	✗
Wrobel [18]	✗	✗	✗	✗	✗	✗
Teasdale et al. [19]	✗	✗	~	~	~	✗
Wells [21]	✗	✗	✗	~	~	✗
Ochsner & Gross [22]	✗	✗	~	✓	✗	✗
This work	✓	✓	✓	✓	✓	✓

3. Theoretical Foundations

This section sets out the mechanism on which the paper rests, using everyday examples and without reference to vibe coding; the application to vibe coding follows in Section 4. The account has four parts: reactive patterns as physical pathways, the sequence by which they fire, the opposing plasticity mechanisms that deepen or weaken them, and the observation practice that engages the weakening.

3.1. Reactive Patterns as Neurone Paths

A *reactive pattern* is a conditioned, automatic response that arises in reaction to a triggering cue—a flash of anger at a dismissive remark, impatience

at an obstacle, a stab of self-criticism at a mistake. We take such patterns to be, quite literally, physical pathways in the brain: assemblies of neurons that have become linked through experience such that activating one reliably activates the rest. A pathway of this kind is laid down in two ways: through *repetition*, as a response performed again and again in similar circumstances is progressively consolidated, and through *intensity*, as a single high-arousal event encodes a pathway in one exposure. That sustained experience physically reshapes neural structure is well documented [24]; a reactive pattern is the affective instance of this general principle.

3.2. The Reactive Process

When a reactive pattern fires, it does so through a rapid, largely automatic sequence (Figure 1). A *signal* arrives—through the senses, or internally as a memory or thought. The *amygdala* matches the signal to a conditioned pathway, rapidly and pre-consciously, before deliberate appraisal has taken place [6, 7]. This match gives rise to a *feeling tone*—a bare, pre-cognitive sense of the event as pleasant, unpleasant, or neutral, arising before and independently of conscious thought [25]. If nothing intervenes, the *pathway fires to completion*, producing the full reaction. Because this sequence runs beneath awareness, the reaction is experienced as automatic and involuntary: by the time one notices being angry, the pathway has already fired.

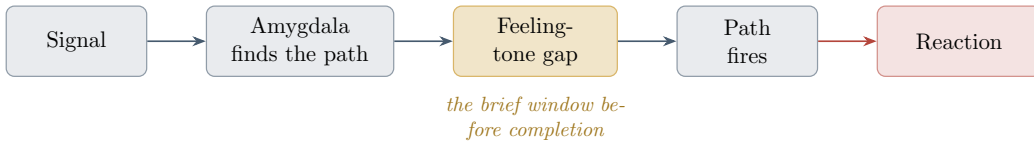


Figure 1: The reactive process. A signal is matched by the amygdala to a conditioned pathway, pre-consciously; a feeling tone arises, opening a brief window; and, if nothing intervenes, the pathway fires to completion, producing the reaction. The feeling-tone gap is the window in which the sequence can be interrupted.

3.3. Two Directions of Plasticity

Whether a reactive pathway deepens or weakens is governed by activity-dependent synaptic plasticity, and specifically by two opposing mechanisms [8, 9, 10, 11]. When a pathway is activated and allowed to fire to completion, the synapses along it are reinforced through *long-term potentiation* (LTP): the pathway is strengthened, and the reaction becomes more

likely and more automatic next time. This is why an indulged reactive pattern deepens with every episode. But the converse also holds: when a pathway is activated yet *not* allowed to complete, the conditions for *long-term depression* (LTD) are met, and the synapses weaken [26]. The same pathway that repetition strengthens, repeated activation-without-completion can dissolve. The pivotal point is therefore the feeling-tone gap: what happens in that brief window—completion or non-completion—determines the direction of plasticity.

3.4. *Behind-the-Scenes Observation*

The practice that engages non-completion we call *behind-the-scenes observation*: at the feeling-tone gap, instead of being carried by the reaction, one turns and observes the reactive process itself. It proceeds in three stages, each bringing awareness to one step of the otherwise-automatic sequence (Figure 2). *Stage 1, notice the signal*: one registers that a signal has arrived. *Stage 2, notice the feeling tone*: one notes the bare pleasant or unpleasant tone as it arises, before it elaborates into a story. *Stage 3, see the process*: one observes the conditioned pathway beginning to fire—“this is a pattern of mine, running; I am watching it, I am not it”—and, in seeing it, does not complete it. This observational stance is the same decentred, process-directed attention developed in mindfulness-based and metacognitive approaches [19, 20, 21, 22], here located precisely at the moment of activation.

The effect is hypothesised to be twofold, and this dual character is central to our account (Figure 2). Observing without completing weakens the old reactive pathway (LTD, Section 3.3); at the same time, the act of observation is itself a response, and one that is repeated each time the practice is performed, so that an alternative *observational* pathway is strengthened (LTP on the new response). Over many repetitions, the default route a trigger takes is hypothesised to shift: from *trigger* → *reaction* toward *trigger* → *observation*, a route whose endpoint is clear seeing rather than reactive suffering. The reaction is not suppressed; a non-reactive alternative is trained to take its place.

Two clarifications complete the account. First, the practice is not thought suppression or distraction: the reaction is not pushed away but *watched*, and it is precisely the watching-without-completing that engages LTD. Second, the mechanism as stated—LTD of the reactive pathway, LTP of the observational one—is a hypothesis grounded in established plasticity principles, not

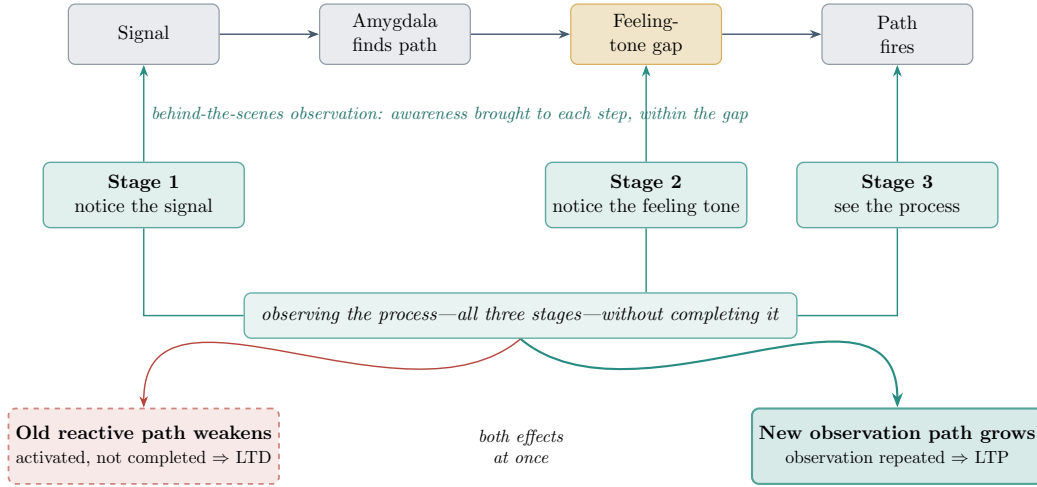


Figure 2: The three-stage observation and its hypothesised dual effect. Each stage brings awareness to one step of the reactive process (top): the signal (Stage 1), the feeling tone in the gap (Stage 2), and the pathway firing (Stage 3). Taken together, these three stages constitute the observation; performed without completing the reaction, it is hypothesised to weaken the old reactive pathway (LTD) while strengthening a new observational pathway (LTP), shifting the default response over repetition from reaction toward observation.

a directly measured finding; what is directly available is the practice itself, which is concrete and specifiable, and its behavioural consequences. With this foundation in place, we turn to the setting that is the subject of this paper.

4. Reactive Patterns in Vibe Coding

We now bring the mechanism of Section 3 to bear on vibe coding. Our claim is twofold: that the vibe-coding workflow is a dense cycle of contact events, each an occasion on which a reactive pattern may fire and be conditioned; and that vibe coding evokes a *distinctive* set of reactive patterns, arising from its defining feature—that the developer directs code they do not write and often do not read.

4.1. The Loop as a Contact-Event Cycle

Vibe coding proceeds as a loop (Figure 3). The developer issues a prompt describing what they want; the model generates code; the developer evaluates the result—running it, scanning its behaviour, noting whether it is what was

wanted; and, on the basis of that evaluation, issues a further prompt. The cycle then repeats, often many times in quick succession and many times across a working day [1, 16].

The pivotal moment in this cycle is *evaluation*: the instant at which the generated result meets the developer. This is a *contact event* in the sense of Section 3.2—a signal arrives (the result), the amygdala matches it against expectation, and a feeling tone arises before deliberate appraisal. When the result is what was wanted, the tone is pleasant; when it fails to compile, misbehaves, or diverges from intent, the tone is unpleasant and a reactive pattern is liable to fire. Two properties of the vibe-coding loop make it an unusually potent conditioning setting. It is *high-frequency*: a single session may contain dozens or hundreds of such evaluations, each a fresh contact event. And it is *low-latency*: the result returns quickly, so stimulus and reaction are tightly coupled, and the loop tightens further as the developer, carried by the reaction, fires off the next prompt without pause [27]. Under the plasticity principles of Section 3.3, each such uninterrupted cycle is a repetition that strengthens the underlying reactive pathway.

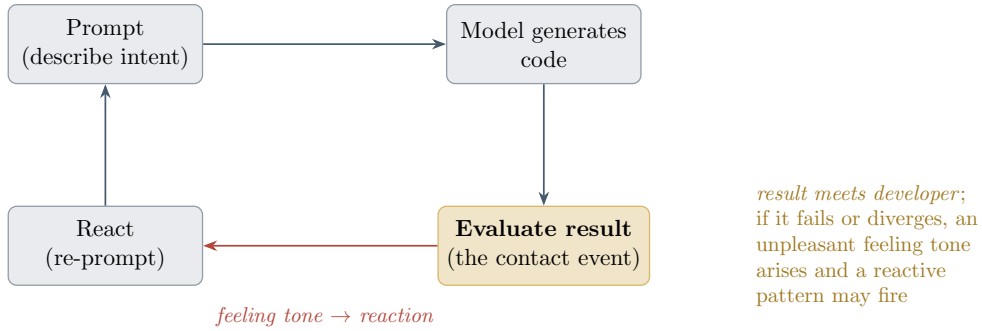


Figure 3: The vibe-coding loop as a contact-event cycle. The developer prompts, the model generates, and the developer evaluates the result—the contact event at which a feeling tone arises and, if the result fails or diverges from intent, a reactive pattern may fire before deliberate appraisal. Carried by the reaction, the developer re-prompts, and the cycle repeats at high frequency and low latency. Each uninterrupted turn strengthens the underlying reactive pathway.

4.2. The Distinctive Reactive Profile of Vibe Coding

Much of what fires at the evaluation step is familiar: the impatience and frustration that attend any tool that does not do what one wants, states long recognised as common and consequential in software work [18, 17]. But vibe

coding also evokes a set of reactive patterns that are specific to it, arising directly from the fact that the developer directs code they do not author. We identify four.

Lost comprehension.. In conventional programming, when code fails the developer can read it, reason about it, and locate the fault. In vibe coding, by design, the developer often has not read the generated code and may not readily follow it [1]. The failure therefore arrives as something to react to without a clear path to resolution: one must debug code one did not write and does not understand. This is a reactive trigger largely absent from other forms of AI interaction and from traditional coding alike—a helplessness peculiar to directing an opaque process. It is compounded by the gap, documented for code-generation tools, between what developers expect the tool to produce and what it actually produces [15, 28].

Lost mastery.. For a practitioner whose craft and professional identity were built on *understanding*—on being the one who knows how and why the code works—vibe coding can unsettle that identity. Ceding authorship to the model, and succeeding through it, can evoke a reactive pattern of threatened competence: a resistance, or a diminishment, that is not about the immediate bug but about what the workflow implies for the self. This pattern is specific to skilled developers meeting a tool that performs the part of the work they had made their own.

Verification anxiety.. Vibe coding often means shipping software one cannot fully vouch for. The developer has not inspected every line, and the evidence that AI-assisted code frequently carries security-relevant defects—and that developers may write less secure code while believing otherwise [12]—gives this unease a real basis. The result is a reactive pattern of low-grade anxiety around trust and responsibility: a persistent tension between the speed the workflow offers and the accountability it withholds.

The surrender gap.. Vibe coding invites the developer to “give in to the vibes” and forget the code exists [1]. When the code then fails, the gap between that invited posture of relaxed surrender and the reality of broken output is itself a source of reactive charge—an expectation-violation built into the practice, in which the promise of effortless creation repeatedly meets the friction of a system that does not quite deliver.

4.3. *Entrenchment Alongside Acceleration*

These patterns are conditioned by the same loop that makes vibe coding productive. Each time a failing result evokes frustration, lost-comprehension helplessness, or verification anxiety, and the developer reacts—re-prompting in irritation, pushing through the unease—the reactive pathway fires to completion and is strengthened (Section 3.3). The loop’s very efficiency works against the developer here: because the cycle is fast and repeated, the conditioning accumulates quickly. There is, moreover, a reinforcing dynamic in the intermittent success of the loop—the next prompt *might* yield the result—that can drive continued engagement past the point of diminishing returns, extending the sequence of contact events and with it the conditioning. Vibe coding thus tends, left unexamined, to entrench the very reactive patterns it provokes, even as it accelerates the production of code. It is precisely this tendency that the practice of the next section is meant to reverse.

5. Rewiring Through Vibe Coding

The argument so far has a discouraging shape: vibe coding is a dense loop of contact events that conditions a distinctive set of reactive patterns, and its efficiency accelerates their entrenchment. But the same loop, engaged differently, admits the opposite outcome. Nothing about the vibe-coding cycle requires that the developer be carried by the reaction at the evaluation step; that step is also an opportunity to *observe*. This section describes how the three-stage observation of Section 3.4 applies to vibe coding, and why vibe coding is, in several respects, unusually well suited to the practice.

5.1. *The Fork at the Evaluation Step*

Every evaluation—every moment at which a generated result meets the developer—is a fork (Figure 4). Down one route, the unpleasant feeling tone of a failing result is allowed to complete: the developer reacts, re-prompts in irritation, and the reactive pathway is strengthened through long-term potentiation. Down the other, the same feeling tone is *observed* rather than completed: at the gap, the developer watches the reactive process operate, the cascade does not run to completion, and the pathway is instead weakened through long-term depression, while the alternative observational response is strengthened (Section 3.3). The two routes are behaviourally similar—in both the developer will likely issue another prompt—but neurologically opposite: one entrenches the reactive pattern, the other dissolves it and

builds an alternative in its place. The difference lies entirely in what happens in the brief window at the contact event [29, 30].

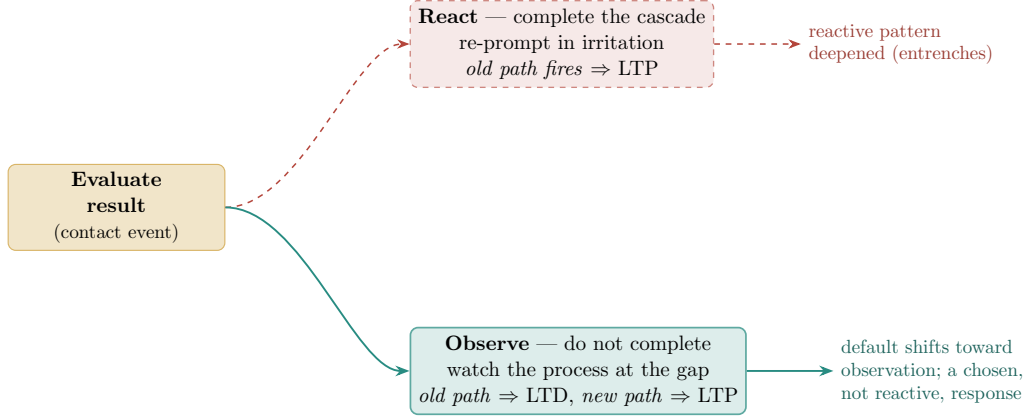


Figure 4: The fork at the evaluation step. At each contact event, the same unpleasant feeling tone can be completed or observed. Completing it—reacting and re-prompting—strengthens the reactive pathway (LTP). Observing it—watching the process at the gap without completing it—weakens the reactive pathway (LTD) while strengthening an alternative observational response, shifting the developer’s default, over many repetitions, from reaction toward observation. The two routes look similar in behaviour but are opposite in their effect on the underlying pathway.

5.2. The Three Stages at the Keyboard

Concretely, the practice unfolds in the three stages of Section 3.4, located at the vibe-coding contact event. *Stage 1, notice the signal*: the developer registers that a result has arrived—the code has run, the error has surfaced, the output is on screen. *Stage 2, notice the feeling tone*: they note the bare unpleasant feeling that accompanies a failure—the flicker of frustration, the sink of helplessness at code they cannot follow—as a sensation, before it elaborates into a story about the tool or themselves. *Stage 3, see the process*: they observe the reactive pattern beginning to fire—“the ‘it is broken again’ pattern is running; the ‘I don’t even understand this’ helplessness is running; I am watching it, I am not it”—and, in seeing it, do not complete it. What follows is still, most likely, another prompt; but it is now a *chosen* response rather than a reactive discharge. Repeated across the many contact events of a session, this is the practice hypothesised to weaken the reactive pathway and grow the observational one [31].

The stance is the decentred, process-directed attention long developed in observation-based therapies [19, 20, 21] and studied as prefrontally mediated emotion regulation [22, 23]; what vibe coding contributes is a setting in which it can be practised many times a day, at the moment reactive patterns actually fire.

5.3. *Two Modes of Practice*

The practice can be pursued in two modes. In the *self-guided* mode, the developer, having learned the three stages, simply performs them at the keyboard—pausing at a failing result to observe rather than react. This requires no change to existing tools. In the *tool-supported* mode, the coding assistant itself participates: it could, at moments likely to provoke reaction—a repeated failure, a long unproductive spiral—offer a brief prompt to observe, turning the tool from a driver of the reactive loop into a support for its interruption. We return to this design possibility in Section 7; the point here is that the practice is available immediately, without tool support, and can be strengthened by it.

5.4. *Why Vibe Coding Is Well Suited*

Several features make vibe coding an unusually favourable training context. It is *frequent*: the loop delivers many contact events per session, and plasticity is driven by repetition, so the practice accrues quickly. It is *immediate*: the interval between stimulus (a result appearing) and response (the reactive re-prompt) is short and clearly demarcated, so the feeling-tone gap is comparatively easy to locate—easier than in the diffuse provocations of ordinary life. It is *low-stakes*: a failed generation is a safe, repeatable, and largely inconsequential stimulus, which makes routine vibe coding a gentler training ground than the high-arousal situations in which reactive patterns are usually met and usually deepened. And it is *transferable*: the reactive patterns that fire at the keyboard—the impatience, the threatened competence, the anxiety around trust—are not confined to coding; they are the same conditioned patterns that operate throughout a person’s life [1]. Vibe coding simply makes them observable at high frequency and low cost. A pathway weakened through observation at the keyboard is weakened wherever it fires. In this light, the very workflow that threatens to entrench reactive patterns becomes, engaged with awareness, a practice ground for their dissolution.

6. Worked Examples

To make the practice concrete, we develop two worked examples drawn from ordinary vibe-coding sessions. In each, we contrast the reactive handling—the default, in which the feeling tone completes into a reaction—with the observed handling, in which the same situation is met with the three-stage observation of Section 5.2. The behaviour looks similar in both; the effect on the underlying pathway is opposite. Table 2 summarises the contrast.

6.1. *The Debugging Spiral*

A developer is building a feature by description. The first generation almost works but mishandles an edge case; the developer re-prompts; the next version fixes that case but breaks another; a third introduces a regression elsewhere. Each result is a fresh contact event, and each falls short in a new way.

In the *reactive* handling, each failure is allowed to complete. The first evokes mild irritation; the second, sharper frustration; by the fourth, the developer is tense, re-prompting quickly and with force, pulled forward by the sense that the next prompt *must* finally work. The reactive pathway fires and is reinforced at every turn, and the escalation is itself evidence of potentiation: the same class of stimulus is provoking a progressively stronger response within a single session. The session ends—whether in success or abandonment—with the developer depleted and the frustration pattern a little more deeply grooved than before.

In the *observed* handling, the debugging proceeds identically at the level of action—the developer still inspects each failure and issues the next prompt—but each contact event is met at the gap. At the first failure: *Stage 1*, a result has arrived; *Stage 2*, there is a flicker of irritation, noticed as a sensation; *Stage 3*, “the ‘it failed again’ pattern is starting to run; I am watching it.” The irritation is seen, not completed. The next prompt is issued from a settled rather than a driven state. Across the spiral, the reactive pathway is repeatedly activated but not run to completion—the condition for its weakening—while the act of observation is repeated and strengthened. The bug is fixed either way; but in the observed case the developer leaves the session slightly less reactive than they entered it.

6.2. *The Comprehension-Loss Moment*

A developer’s vibe-coded application, working yesterday, today throws an error deep in code the developer never read and does not understand [1]. This is the trigger distinctive to vibe coding (Section 4.2): a failure with no familiar path to resolution, because the usual recourse—read the code, reason about it—is unavailable to someone who did not write it and cannot readily follow it.

In the *reactive* handling, the helplessness completes. It may complete into a story about the self—“I am losing my ability to actually program”—which is the lost-mastery pattern firing and deepening; or into thrashing, a flurry of undirected re-prompts that discharge the discomfort without addressing the fault; or into the low-grade anxiety of not being able to vouch for one’s own software [12]. Whichever route, the distinctive vibe-coding pathway is potentiated.

In the *observed* handling, the same moment is met in three stages. *Stage 1*: the application has broken (the signal). *Stage 2*: there is a particular sinking feeling—the helplessness of the opaque failure—noticed as a bare sensation. *Stage 3*: “the ‘I don’t understand this and I’m powerless’ pattern is firing; I am watching it, I am not it.” Seen and not completed, the helplessness does not escalate into a self-story or a thrash. From that settled state, the developer chooses a response—reading into the code after all, asking the model to explain it, seeking help, or accepting the limitation—rather than being driven by the reaction. The lost-comprehension pathway is weakened rather than deepened, and a capacity to meet opaque failure with observation is built in its place.

6.3. *Behaviourally Similar, Neurologically Opposite*

The two examples share a feature worth making explicit. In each, the observed handling is almost indistinguishable, from the outside, from the reactive one: the developer debugs, re-prompts, and carries on in both. Nothing in the workflow need change, no time need be added, no tool need be installed. What differs is entirely internal—whether, at the contact event, the feeling tone is completed or observed—and yet that difference determines the direction of plasticity, and so the developer’s trajectory across a career of such sessions. This is what makes vibe coding a genuine training environment in both directions: the same millions of contact events that can entrench reactive patterns can, met with observation, dissolve them, and the choice between these outcomes is available at every turn of the loop.

Table 2: Reactive versus observed handling of the same vibe-coding contact event. The action is similar; the effect on the underlying pathway is opposite.

At the contact event	Reactive handling	Observed handling
The feeling tone	completes	is observed
The developer	reacts, re-prompts	pauses, then chooses
Behaviour	issues a prompt	issues a prompt
Old reactive path	fires $\Rightarrow$ LTP	not completed $\Rightarrow$ LTD
Observational path	–	repeated $\Rightarrow$ LTP
Within the session	escalation, depletion	settling
Over time	pattern entrenched	pattern dissolved

7. Discussion

We draw out the implications of the account for developer wellbeing and for the design of AI coding tools, situate it relative to established approaches, and state its limitations.

7.1. *Implications for Developer Wellbeing*

The dominant framing of frustration in AI-assisted development treats it as a cost to be minimised—a source of lost productivity and diminished satisfaction [17, 18]. Our account reframes it. The frustration evoked by a failing generation is not only a cost; it is also an occasion—a repeated, low-stakes, clearly-timed opportunity to practise meeting a reactive pattern with observation rather than being carried by it. On this view the vibe-coding loop is not merely a hazard to developer wellbeing but, engaged with awareness, a resource for it. Because the reactive patterns that fire at the keyboard are the same conditioned patterns that operate elsewhere in a person’s life (Section 5.4), a developer who weakens the impatience or helplessness pattern through observation at work carries that change beyond work. The wellbeing implication is therefore not confined to coding: the workplace becomes, incidentally, a place where a broadly useful capacity can be trained.

This reframing should be held carefully. It is not a counsel to seek out frustration, nor a suggestion that the friction of immature tools is therefore acceptable; the friction is real and worth reducing. The claim is only that, given that the friction exists, the reaction it provokes need not be completed,

and that not completing it is both better in the moment and beneficial over time.

7.2. Implications for AI Coding Tool Design

If the vibe-coding loop can be a site of observation, tools can be designed to support it (Figure 5). An AI coding assistant is unusually well placed to do so: it can detect the very moments most likely to provoke reaction—a repeated failure on the same task, a lengthening spiral of unproductive prompts, a rapid burst of re-prompts issued in quick succession—and, at those moments, offer a brief, optional prompt to observe before continuing. Rather than silently feeding the reactive loop, the tool would occasionally hold it open: “notice what is arising before the next prompt.” This is an instance of designing technology for wellbeing and reflection rather than for throughput alone [32].

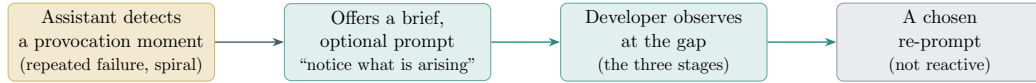


Figure 5: A design sketch for tool-supported observation. Rather than silently feeding the reactive loop, an AI coding assistant could detect moments likely to provoke reaction and offer a brief, optional prompt to observe before the next re-prompt—turning the tool from a driver of the reactive loop into a support for its interruption. The prompt must be optional and well-timed, lest it become a further source of friction.

The design carries an obvious risk: a prompt to observe, if frequent or ill-timed, becomes exactly the kind of interruption that itself provokes reaction. The intervention must therefore be sparing, well-timed, and dismissible, offered at genuine inflection points rather than on a schedule, and easy to turn off. Used well, it would represent a small but meaningful shift in what AI coding tools optimise for: not only the speed with which code is produced, but the state of the person producing it.

7.3. Relationship to Established Approaches

The observation described here is not new as a practice; it is the decentred, process-directed attention cultivated in mindfulness-based and metacognitive therapies [19, 20, 21] and studied as emotion regulation [22, 23]. What is new is its setting. Those approaches are ordinarily cultivated in dedicated contexts—meditation, therapy, set-aside exercises—and the difficulty of transfer to daily life is well known. Vibe coding offers something

unusual: a high-frequency stream of reactive triggers arising within an activity the developer already performs for hours a day, so that the practice can be done at the moment patterns actually fire, without setting aside separate time. The proposal is emphatically not a substitute for therapy or clinical care; it addresses the ordinary, sub-clinical reactivity of everyday work, and is complementary to, not a replacement for, established treatment where that is needed.

7.4. Limitations

Several limitations bear statement. First, the core mechanism—weakening of the reactive pathway through LTD, strengthening of an observational pathway through LTP—is a hypothesis grounded in established plasticity principles, not a directly measured finding; what is directly specifiable is the practice and its behavioural consequences, and the synaptic account is the theoretical frame whose predictions future work would test. Second, we offer no empirical evaluation here: the claims about entrenchment and dissolution are theoretically motivated and illustrated through worked examples, but their magnitude and reliability in practice remain to be established, and doing so—through behavioural and physiological measures of reactivity change over a course of practice—is the natural next step. Third, the effect almost certainly varies between individuals, and the practice is not equally accessible to everyone; for those in acute distress it is not a substitute for appropriate support. Finally, the account is presented for vibe coding specifically, but the mechanism is general; whether vibe coding is genuinely more suitable than other high-frequency human–computer loops is itself an empirical question.

8. Conclusion

Vibe coding has, in a very short time, become a dominant way of building software: the developer describes intent, the model generates code, and the two proceed through a fast loop of generation, evaluation, and re-prompting [1, 2]. We have argued that this loop is more than a productivity mechanism. Each evaluation is a contact event at which a conditioned reactive pattern may fire before deliberate appraisal, and because vibe coding delivers such events at high frequency and low latency, it is a dense setting for the conditioning of those patterns. Left unexamined, the loop tends to entrench them—and vibe coding carries a reactive profile of its own, in the

helplessness of directing code one does not understand, the unsettling of a craft identity built on comprehension, and the anxiety of shipping software one cannot vouch for.

The same loop, we have argued, can be turned to the opposite end. At each contact event the arising feeling tone can be observed rather than completed, in three stages—noticing the signal, noticing the feeling tone, and seeing the reactive process as it fires. This rewires the pathway: the old reactive route is weakened rather than strengthened, and an alternative observational response is built in its place. Vibe coding is, in several respects, unusually well suited to this practice: its contact events are frequent, immediate, and low in stakes, and the patterns they evoke are the same ones that operate throughout a person’s life, so that what is practised at the keyboard transfers beyond it. The difference between entrenchment and dissolution is entirely internal—whether the feeling tone is completed or observed—and available at every turn of the loop.

The account is a theoretical proposal, and its central mechanism remains a hypothesis: what we claim to specify is a concrete, teachable practice and its behavioural consequences, while the underlying story of long-term depression and potentiation is the theoretical frame whose predictions are yet to be tested. The natural next steps follow directly. The behavioural and physiological effects of the practice—changes in the latency, intensity, and generalisation of reactive responding over a course of observation—can be measured directly, as the effects of established regulation practices are measured. The tool-supported mode sketched in Section 7.2—an assistant that offers, at well-chosen moments, a brief invitation to observe—can be built and evaluated. And the question of whether vibe coding is genuinely more suitable than other high-frequency human–computer loops, or simply a vivid instance of a general phenomenon, can be examined comparatively.

Vibe coding is often described by what the developer gives up: the code, the comprehension, the craft. We have suggested that it offers something back that has gone unnoticed—a training ground for the developer’s brain, repeated thousands of times a day, for the capacity to meet a reactive pattern with awareness rather than be carried by it. Whether that capacity is trained toward entrenchment or toward dissolution is not determined by the tool. It is determined, at each small moment of contact, by the developer.

References

- [1] A. Karpathy, There’s a new kind of coding i call “vibe coding”, Post on X (formerly Twitter), February 2, coinage of the term “vibe coding” (2025).
- [2] Collins Dictionary, Vibe coding named collins dictionary’s word of the year 2025, Collins Dictionary (2025).
- [3] E. Bandara, R. Gore, X. Liang, S. Rajapakse, I. Kularathne, P. Karunarathna, P. Foytik, S. Shetty, R. Mukkamala, A. Rahman, et al., Agentsway—software development methodology for ai agents-based teams, arXiv preprint arXiv:2510.23664 (2025).
- [4] M. Chen, J. Tworek, H. Jun, Q. Yuan, H. P. d. O. Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman, et al., Evaluating large language models trained on code, arXiv preprint arXiv:2107.03374 (2021).
URL <https://arxiv.org/abs/2107.03374>
- [5] E. Brynjolfsson, D. Li, L. R. Raymond, Generative AI at work, National Bureau of Economic Research Working Paper No. 31161 (2023). doi:10.3386/w31161.
- [6] J. E. LeDoux, The Emotional Brain: The Mysterious Underpinnings of Emotional Life, Simon and Schuster, New York, 1996.
- [7] A. Öhman, The role of the amygdala in human fear: Automatic detection of threat, Psychoneuroendocrinology 30 (10) (2005) 953–958. doi:10.1016/j.psyneuen.2005.03.019.
- [8] D. O. Hebb, The Organisation of Behavior: A Neuropsychological Theory, Wiley, New York, 1949.
- [9] T. V. P. Bliss, G. L. Collingridge, A synaptic model of memory: long-term potentiation in the hippocampus, Nature 361 (1993) 31–39. doi:10.1038/361031a0.
- [10] R. C. Malenka, M. F. Bear, LTP and LTD: An embarrassment of riches, Neuron 44 (1) (2004) 5–21. doi:10.1016/j.neuron.2004.09.012.

- [11] A. Citri, R. C. Malenka, Synaptic plasticity: Multiple forms, functions, and mechanisms, *Neuropsychopharmacology* 33 (1) (2008) 18–41. doi:10.1038/sj.npp.1301559.
- [12] N. Perry, M. Srivastava, D. Kumar, D. Boneh, Do users write more insecure code with ai assistants?, in: *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2023, pp. 2785–2799. doi:10.1145/3576915.3623157.
- [13] E. Bandara, R. Gore, P. Foytik, S. Shetty, R. Mukkamala, A. Rahman, X. Liang, S. H. Bouk, A. Hass, S. Rajapakse, et al., A practical guide for designing, developing, and deploying production-grade agentic ai workflows, *arXiv preprint arXiv:2512.08769* (2025).
- [14] A. Ziegler, E. Kalliamvakou, X. A. Li, A. Rice, D. Rifkin, S. Simister, G. Sittampalam, E. Aftandilian, Productivity assessment of neural code completion, *Proceedings of the 6th ACM SIGPLAN International Symposium on Machine Programming (MAPS)* (2022) 21–29doi:10.1145/3520312.3534864.
- [15] P. Vaithilingam, T. Zhang, E. L. Glassman, Expectation vs. experience: Evaluating the usability of code generation tools powered by large language models, in: *Extended Abstracts of the 2022 CHI Conference on Human Factors in Computing Systems*, 2022, pp. 1–7. doi:10.1145/3491101.3519665.
- [16] S. Barke, M. B. James, N. Polikarpova, Grounded copilot: How programmers interact with code-generating models, in: *Proceedings of the ACM on Programming Languages (OOPSLA)*, Vol. 7, 2023, pp. 85–111. doi:10.1145/3586030.
- [17] D. Graziotin, F. Fagerholm, X. Wang, P. Abrahamsson, What happens when software developers are (un)happy, *Journal of Systems and Software* 140 (2018) 32–47. doi:10.1016/j.jss.2018.02.041.
- [18] M. R. Wrobel, Emotions in the software development process, in: *Proceedings of the 6th International Conference on Human System Interactions (HSI)*, 2013, pp. 518–523. doi:10.1109/HSI.2013.6577877.

- [19] J. D. Teasdale, R. J. Howard, S. G. Cox, Y. Ha, M. J. Brammer, S. C. R. Williams, S. A. Checkley, Functional MRI study of the cognitive generation of affect, *American Journal of Psychiatry* 156 (2) (1999) 209–215. doi:10.1176/ajp.156.2.209.
- [20] Z. V. Segal, J. M. G. Williams, J. D. Teasdale, *Mindfulness-Based Cognitive Therapy for Depression: A New Approach to Preventing Relapse*, Guilford Press, New York, 2002.
- [21] A. Wells, *Metacognitive Therapy for Anxiety and Depression*, Guilford Press, New York, 2009.
- [22] K. N. Ochsner, J. J. Gross, The cognitive control of emotion, *Trends in Cognitive Sciences* 9 (5) (2005) 242–249. doi:10.1016/j.tics.2005.03.010.
- [23] J. J. Gross, Emotion regulation: Current status and future prospects, *Psychological Inquiry* 26 (1) (2015) 1–26. doi:10.1080/1047840X.2014.940781.
- [24] B. Draganski, C. Gaser, V. Busch, G. Schuierer, U. Bogdahn, A. May, Changes in grey matter induced by training, *Nature* 427 (6972) (2004) 311–312. doi:10.1038/427311a.
- [25] R. B. Zajonc, Feeling and thinking: Preferences need no inferences, *American Psychologist* 35 (2) (1980) 151–175. doi:10.1037/0003-066X.35.2.151.
- [26] E. Bandara, R. Gore, A. Gunaratna, R. Mukkamala, N. Siriwardanagea, S. Rajapakse, I. Kularathna, P. Karunarathna, W. Herath, C. Rajapakse, et al., Mindgap: A conversational ai framework for upstream neuroplastic intervention in post-traumatic stress disorder, arXiv preprint arXiv:2605.14660 (2026).
- [27] E. Bandara, R. Gore, A. Gunaratna, R. Mukkamala, N. Siriwardanagea, G. Siriwardanagea, S. Rajapakse, I. Kularathna, P. Karunarathna, C. Rajapakse, et al., Human-ai agent interaction as a neuroplastic training environment, arXiv preprint arXiv:2607.12823 (2026).
- [28] M. Yi, Revolutionizing interaction: the role of artificial intelligent conversation agents in human-computer interaction, in: *Fourth International Conference on Signal Processing and Machine Learning (CONF-SPML 2024)*, Vol. 13077, SPIE, 2024, pp. 192–201.

- [29] E. Bandara, T. Hewa, R. Gore, S. Shetty, R. Mukkamala, P. Foytik, A. Rahman, S. H. Bouk, X. Liang, A. Hass, et al., Towards responsible and explainable ai agents with consensus-driven reasoning, arXiv preprint arXiv:2512.21699 (2025).
- [30] D. B. Acharya, K. Kuppan, B. Divya, Agentic ai: Autonomous intelligence for complex goals—a comprehensive survey, IEEE Access (2025).
- [31] E. Bandara, R. Gore, S. Shetty, S. Rajapakse, I. Kularathna, P. Karunarathna, R. Mukkamala, P. Foytik, S. H. Bouk, A. Rahman, et al., A practical guide to agentic ai transition in organizations, arXiv preprint arXiv:2602.10122 (2026).
- [32] R. A. Calvo, D. Peters, Positive Computing: Technology for Wellbeing and Human Potential, MIT Press, Cambridge, MA, 2014.