

Beyond Individual Prompting Efficiency: A Socio-Technical Perspective on Computational Efficiency in AI-Assisted Software Engineering

Darja Smite 

Blekinge Institute of Technology, Sweden

Gabrielle O'Brien 

University of Michigan Ann Arbor, MI, USA

Giuseppe Destefanis 

University College London, UK

Mircea Lungu 

IT University of Copenhagen, Denmark

Ronnie de Souza Santos 

University of Calgary, Canada

Viggo Tellefsen Wivestad 

SINTEF Digital, Trondheim, Norway

Italo Santos 

University of Hawaii at Manoa, USA

Maria Teresa Baldassarre 

University of Bari, Italy

Elda Paja 

IT University of Copenhagen, Denmark

Abstract

As Artificial Intelligence (AI) assistants and agents become increasingly integrated into software engineering work, concerns about the economic and environmental costs of AI usage continue to grow. Current discussions of computational efficiency focus primarily on model architectures, hardware optimization, and inference cost reduction. In this vision paper, we argue that computational efficiency should also be understood as a socio-technical phenomenon shaped by patterns of human-AI interaction within organizational settings. Motivated by ongoing dialogues with industry partners and informed by the Theory of Planned Behavior and Social Cognitive Theory, we introduce the concept of computational behavior to describe patterns of AI-assisted work that influence how computational resources are consumed, reused, coordinated, and amplified across software engineering activities. Building on this perspective, we propose a multilevel conceptual framework linking individual computational behaviors to emergent collective computational outcomes. We argue that individually rational AI usage behaviors may accumulate into either computational waste or collective computational efficiency depending on how interactions are shared, reused, and aligned across teams and workflows. While fragmented AI use may generate duplicated prompting, redundant generation, and coordination overhead, coordinated collective practices, such as reusable prompts, shared contextualization, and workflow integration may improve collective computational efficiency. We conclude by outlining a research agenda for studying computational coordination, including empirical and organizational AI governance in AI-assisted software engineering.

2012 ACM Subject Classification Software and its engineering → Collaboration in software development; Software and its engineering → Programming teams

Keywords and phrases AI-assisted software engineering, Computational waste, Team practices, Mental models, Empirical software engineering, Vision, Research agenda

Digital Object Identifier 10.4230/LIPIcs.ESEM.2026.69

Category Emerging Results, Vision & Reflection Track Paper

Funding This research was partially supported by the Swedish Knowledge Foundation through the GAINS project (grant 2025/0103) at Blekinge Institute of Technology (BTH), the Alfred P. Sloan Foundation's Technology program at the University of Michigan and Regione Puglia through the project "AURIGA-EDICO" (Ecological Digital Conversion), under the Regional Regulation n.17/2014 – PIA (art. 26) – Project code: 4WBHY73, at the University of Bari.



© Darja Smite, Viggo Tellefsen Wivestad, Gabrielle O'Brien, Italo Santos, Giuseppe Destefanis, Maria Teresa Baldassarre, Mircea Lungu, Elda Paja, and Ronnie de Souza Santos;
licensed under Creative Commons License CC-BY 4.0

20th International Symposium on Empirical Software Engineering and Measurement (ESEM 2026).
Editors: Robert Feldt, Maria Paasivaara, Daniel Mendez, Stefan Wagner, and Marvin Muñoz Barón;
Article No. 69; pp. 69:1–69:14



Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 **Introduction**

Generative Artificial Intelligence (GenAI) is increasingly integrated into software development activities [7], and a large share of software engineers already use or plan to use AI tools in their daily work [36]. AI tools are increasingly efficient, but also complex and environmentally questionable, as they entail the often-overlooked consumption of enormous computational power and a significant carbon footprint for model training and deployment, as well as electronic waste associated with high hardware turnover and data center infrastructure [20, 43]. Therefore, assessments of AI’s environmental outcomes and the responsible adoption of AI in industrial settings are important topics in today’s reality [20].

While current optimization efforts focus primarily on model architectures, inference, hardware, and token reduction, computational efficiency is also shaped by how individuals and teams interact with AI systems. For example, software engineers may repeatedly regenerate outputs, independently rediscover similar prompting strategies, or default unnecessarily to more complex and expensive models. In contrast, organizations may reduce computational overhead through shared prompts, reusable workflows, and coordinated verification practices, to name a few examples. These examples suggest that computational efficiency, the extent to which computational behaviors produce desirable computational outcomes while minimizing unnecessary computational, organizational, and coordination costs, cannot be understood solely as a technical property of AI systems, but must also be studied as a socio-technical phenomenon.

In this vision paper, we argue that an important share of computational waste originates from the use of AI at the organizational level. We start with a summary of related work on computational efficiency from a technical perspective, followed by the motivation for why computational outcomes in organizational AI usage is also a socio-technical problem. We present our multi-level conceptualization of AI interaction behaviors across individual, group, and organizational levels. Particularly, our framework distinguishes between individual rationality and collective efficiency, between productive and wasteful redundancy, and between aggregated and socially amplified interactions. The vision paper concludes with a research agenda for empirical software engineering.

2 **Background**

2.1 Technical Perspective on Computational Efficiency

As GenAI grows in scale and adoption, sustainability concerns have become increasingly prominent. Green AI emphasizes advancing scientific progress while reducing resource consumption whenever possible [32].

Prior work has examined sustainability impacts across training and inference phases [19, 21, 30]. Proposed optimizations span the hardware, algorithmic, and systems levels, including phase-splitting architectures [27], distributed execution frameworks [33], and custom inference libraries [16, 41]. Current evidence suggests that LLM deployments can consume considerable amounts of electricity when inference demands are aggregated across millions of users [40]. Studies on heterogeneous data center architectures show that even modest improvements in task scheduling can meaningfully reduce inference energy usage, highlighting the resource-intensive nature of LLM workloads and the need for infrastructure-level optimizations [42]. Measurement remains challenging, as sustainability metrics are inconsistently defined across studies, which limits comparability and cumulative progress [2, 8, 19]. This inconsistency also makes it difficult to attribute computational outcomes to specific causes, including human interaction patterns.

Yet, as software engineering history repeatedly shows, seemingly technical challenges are rarely solved through technical improvements alone. Advances in programming languages, for example, have not eliminated persistent problems in software projects, partially because many outcomes depend on coordination, workflows, and organizational practices.

A similar shift is now needed in discussions of AI efficiency. Recent work has begun to extend efficiency concerns beyond infrastructure toward human interaction patterns [31]. While computational outcomes are treated as technical property of AI systems, they are also shaped by how people use those systems in practice [20]. For example, prior studies suggest that some LLM queries could be served by lower-cost alternatives, pointing to demand-side behavior as an important and underexplored lever for improving computational efficiency [13].

2.2 The Impact of Human-AI Interaction on Computational Outcomes

Attitudes Toward AI. Developers' attitudes toward AI tools range from enthusiastic early adoption to active skepticism and resistance, and this variation has direct computational consequences [29, 44]. However, attitudes do not operate in isolation: recent work shows that GenAI use is also shaped by compatibility with existing workflows, organizational support, and collective usage patterns within teams [37]. The Theory of Planned Behavior (TPB) [1] provides a principled frame: attitude toward a behavior, combined with perceived social pressure and perceived behavioral control, predicts behavioral intention and, in turn, behavior itself. AI enthusiasm, supported by documented productivity but amplified socially beyond what task demands justify, leads developers to shift to the most capable and expensive models regardless of task fit [28, 29]. This defaulting behavior resembles the Price-Quality Heuristic [11]: developers may implicitly treat more capable and expensive models as better choices even when the task does not require them. This tendency is especially concerning given longitudinal mixed-methods evidence showing no statistically significant correlation between developers' self-reported productivity change and changes in commitment-based activity [38], suggesting that developers' subjective assessments of AI support may be poorly calibrated against its measurable effects. Within Social Cognitive Theory (SCT), according to Bandura [4, 5], this reflects uncalibrated outcome expectations, that is, beliefs that AI support with more powerful models always yield better results, regardless of the task at hand. On the other end, algorithm aversion [9] drives over-verification and redundant prompting, a pattern enforced by self-efficacy in developers who lack confidence in their own ability to evaluate AI outputs. Between these extremes, automation bias leads to uncritical acceptance of flawed outputs, downstream error loops and costly remediation [26, 35]. Indeed, in a recent study, Zhou et al. [46] assess that 48,8% of programmer actions in LLM-assisted development have been found to be biased, with automation and confirmation bias being dominant patterns. Together, these three attitude-driven pathways - overconsumption, over-verification, and under-verification followed by rework - illustrate why the TPB's multi-component structure matters, confirming that attitude-level interventions alone are insufficient if team practices and perceived behavioral controls are not properly addressed.

Mental Models, Self-Efficacy and Outcome Expectations. Users with more accurate mental models of AI achieve better outcomes in AI-assisted tasks [6, 10, 15], yet forming accurate mental models of LLM-based tools is particularly challenging [10, 24, 39, 45]. Commercial AI tools frequently combine LLM capabilities with tool-calling and knowledge base integration, blurring the boundary between what the model knows and what it can retrieve at inference time. O'Brien et al. [25] found that users frequently hold categorical misconceptions about which affordances are enabled for a given tool, leading to confusion

about what actions were taken before a response is returned. Such misconceptions have also been observed as cultural phenomena: folk prompting strategies such as tipping or threatening models persist despite producing no reliable performance gains [22]. These misconceptions translate into computationally costly behaviors. The monotonicity belief [12] that more context always improves performance, drives context window stuffing and wasteful token consumption. Misunderstandings about model memory lead to repeated re-injection of context across sessions, a pattern documented both among novice programmers and non-AI-experts [18, 45] and, in telemetry from professional developers, as frequent re-prompting to supply context the model lacked [23]. Further, categorical confusion about tool capabilities results in reliance on expensive general purpose models for tasks that are better suited for cheaper alternatives.

3 Conceptual Framework

In this section, we outline our multilevel conceptual framework of computational efficiency. The framework is organized according to guidelines for theory building [34] and describes our vision, constructs, propositions, theoretical grounding, and scope. The framework is inspired by ongoing dialogues with industry partners and theoretical concepts from social science.

3.1 The vision

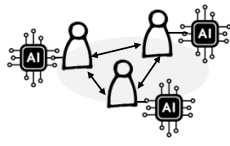
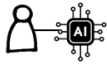
We argue that advances in sustainable and effective AI-assisted software engineering will require complementing technical optimization with a deeper understanding of computational efficiency from a socio-technical perspective. While existing focus on AI models, hardware infrastructure, and inference optimization remains important, it only partially explains how computational resources are consumed in the context of AI-assisted software engineering.

Computational cost is local, and the people who incur it are not aware of the overall impact of their actions on the organization. The individuals are aware of the tokens they personally consume, the organization sees the overall expenses, but neither can see which collaborative practices are beneficial, which redundancies are wasteful, or how differently individuals use resources to re-create the same contexts.

The vision that this paper presents is one in which computational cost is monitored, managed, and treated the same way maintainability, code quality, and technical debt are treated by mature organizations: good practice patterns are disseminated, awareness is encouraged, and tactics ensure that computational effort is used efficiently towards creating value while still allowing for the exploration and freedom that individual learning requires.

At the individual level, human-AI interactions may differ both in rationality and efficiency. Interactions that are locally rational in the sense that they align with an individual developer's immediate goals, constraints, and incentives may contribute to collectively inefficient outcomes when repeated independently across teams or workflows. At the group and organizational level, interaction behaviors may accumulate, amplify, or become coordinated through shared practices, producing emergent patterns of computational efficiency or waste (see Figure 1). From this perspective, *computational efficiency becomes an emergent socio-technical property rather than merely a characteristic of underlying technical systems*.

Finally, as software is increasingly operated by autonomous or semi-autonomous agents, computational behaviors of interest in our context extend beyond direct prompting to include how developers configure, delegate to, constrain, and supervise agentic workflows.

Level of analysis	Human-AI interaction	Computational outcomes	Examples
 Group and org level	Socially amplified interaction	Wasteful use	Collaborative prompt spirals, Tokenmaxxing
		Efficient use	Collaborative sensemaking, Shared contextualization
	Aggregated collective interaction	Wasteful redundancy	Redundant querying, Duplicated generation
		Efficient redundancy	Parallel exploration, Redundant learning
 Individual level	Individual interaction	Irrational use	Prompt thrashing, Context window stuffing
		Rational use	Exploratory prompting, Risk-aware verification

■ **Figure 1** Multilevel conceptual framework of computational efficiency that distinguishes between individual rationality and collective efficiency and showing their possible computational outcomes.

3.2 Constructs

We propose to study *computational behaviors*, i.e., patterns of human-AI interactions that influence computational resource consumption during software engineering work.

Our overarching goal is to understand and improve *computational efficiency*, i.e., the extent to which computational behaviors produce desirable computational outcomes while minimizing unnecessary computational, organizational, and coordination costs.

In this context, *computational outcomes* refer to the consequences of computational behaviors, including resource consumption at the individual, group, and organizational levels (e.g., total time, token consumption) and the organizational and economic consequences produced (e.g., velocity, duplication).

We further introduce the concept of *computational waste*, referring to avoidable computational expenditure that produces limited additional value relative to organizational goals or outcomes achieved. Finally, *rationality* is the property of computational behavior of being logical/justified from the perspective of the actor who performs it.

3.3 Propositions (or relationships)

3.3.1 Individual level

At the individual level, we focus on observable computational behaviors through which individuals engage with AI systems to accomplish their work tasks, and which shape computational outcomes. These behaviors include prompting, verification, context provision, exploration, refinement, regeneration, and interpretation of AI-generated outputs. Such interactions influence computational outcomes by shaping how computational resources are consumed during software engineering work.

Proposition 1: Individual Human-AI interactions vary in rationality. The computational outcomes of AI-assisted software engineering are influenced by the ways in which individuals interact with AI systems, including how often they prompt, regenerate, verify, refine, or contextualize AI-generated outputs. To better understand these behaviors, we focus on the *rationality* of computational behaviors. We use rationality in a bounded sense, referring to whether an interaction is justifiable from the developer’s local perspective, rather than to optimal decision-making in a formal sense. For example, repeated prompting and verification may be rational when developers face uncertainty, high-risk tasks, or opportunities for learning and exploration. In contrast, irrational interactions include repeated regeneration of

a prompt without converging on a usable result (*prompt thrashing*), redundant or unnecessary context provision (*context window stuffing*), or reliance on unnecessarily advanced and costly AI systems for tasks that can be more efficiently performed by simpler models or manually.

Proposition 2: Rationality of individual computational behaviors is shaped, among others, by attitudes toward and mental models of AI capabilities. Individual computational behaviors are influenced by developers' attitudes, expectations, and mental models regarding AI capabilities and limitations. For example, developers who perceive AI systems as highly capable may delegate larger portions of software engineering work to AI assistance, rely less on manual verification, or engage in extensive exploratory prompting. On the other hand, developers who distrust AI systems or underestimate their capabilities may avoid potentially beneficial AI-supported workflows or rely excessively on manual work. As a result, differences in attitudes and perceptions of capability may produce substantially different computational behaviors, even for similar tasks.

3.3.2 Group level

As software engineering work is highly collaborative, computational outcomes cannot be fully understood through individual behavior alone. At the group level, computational outcomes emerge through the accumulation and coordination of interaction behaviors across groups and workflows. We distinguish between aggregated and socially amplified computational behaviors.

Proposition 3: Individually rational interactions may aggregate as collective waste. Aggregated computational behaviors emerge through the accumulation of independently executed AI usage behaviors across collaborators, often without direct coordination. Such aggregation may introduce computational redundancy through repeated querying, duplicated generation, parallel experimentation, or fragmented AI-assisted workflows. For example, multiple developers independently exploring similar prompting strategies may each behave rationally from an individual perspective while collectively reproducing avoidable computational effort, i.e., collective waste. This mechanism extends to autonomous agents: when multiple agents or agentic pipelines independently spawn overlapping sub-tasks, retries, or parallel exploration branches, redundant computation can accumulate with no human coordination point at all. Redundancy becomes wasteful when it duplicates computational effort without producing a proportional increase in value. However, redundancy itself is not inherently inefficient. In some cases, it can serve valuable purposes such as enabling independent verification, improving resilience, supporting exploration, or facilitating collective learning. Organizations can e.g. have slow approval processes for new tools and processes. Allowing some experimentation, while often initially wasteful on an individual level, could aggregate to early evidence that motivates organizations to fast-track serious evaluation and approval of promising tools or processes in order to remain competitive

Proposition 4: Socially amplified human-AI interactions can improve collective efficiency when managed well. In addition to aggregated behaviors, software teams, groups, and communities may engage in socially amplified computational behaviors, in which Human-AI interaction with AI systems, prompts, contexts, workflows, or agents is shared or coordinated across multiple users. Unlike independent aggregation, socially amplified interactions involve interdependent AI-assisted collaboration or interaction behaviors. For example, collaborative

prompt repositories, shared contextualization practices, and coordinated AI workflows reduce duplicated inference and improve reuse across collaborators. In contrast, collaborative prompt spirals, persistent context expansion, excessive collective verification, or recursive agent interactions amplify computational waste. Multi-agent settings intensify this dynamic: when agents recursively delegate subtasks to other agents or re-invoke themselves without adequate stopping conditions, small inefficiencies compound across the execution trajectory rather than resetting at each turn [3]. Conversely, well-designed agent orchestration, e.g., shared memory across agents, explicit hand-off protocols, or bounded delegation depth, can reduce duplicated inference and convert multi-agent coordination into a source of collective efficiency rather than waste. Thus, at a group level, amplification may either improve or degrade collective computational efficiency. For example, collaborative prompt repositories, shared contextualization practices, and coordinated AI workflows reduce duplicated inference and improve reuse across collaborators. In contrast, collaborative prompt spirals, persistent context expansion, excessive collective verification, or recursive agent interactions amplify computational waste. Multi-agent settings intensify this dynamic: when agents recursively delegate subtasks to other agents or re-invoke themselves without adequate stopping conditions, small inefficiencies compound across the execution trajectory rather than resetting at each turn [3]. Conversely, well-designed agent orchestration, e.g., shared memory across agents, explicit hand-off protocols, or bounded delegation depth, can reduce duplicated inference and convert multi-agent coordination into a source of collective efficiency rather than waste. Thus, at a group level, amplification may either improve or degrade collective computational efficiency.

3.3.3 Organizational level

At the organizational level, computational behaviors shape broader patterns of computational alignment and misalignment across teams, workflows, and organizational structures.

Proposition 5: Individualized AI adoption weakens organization-wide computational alignment. When AI usage practices, computational expenditure, and collaborative workflows are insufficiently coordinated, organizational-level computational use is misaligned. Such fragmentation is more likely to surface in organizations that emphasize individual AI adoption over coordinated collective practices. As a result, hidden computational costs emerge through repeated interactions, duplicated inference, fragmented workflows, and weak institutionalization of efficient AI-assisted practices.

Proposition 6: Management beliefs and top-down AI adoption initiatives amplify computational behaviors across workflows and teams. Computational inefficiency does not only aggregate, it may also be amplified through top-down organizational dynamics. Management shapes AI-related norms, expectations, and governance structures that influence how employees interact with AI systems. For example, unrealistic expectations regarding the transformative capabilities of AI may encourage excessive AI usage or “Tokenmaxxing” behaviors [14], where employees increase visible AI interaction to signal productivity, innovation, or organizational alignment. Similarly, inaccurate organizational mental models regarding how AI systems function may lead organizations to deploy expensive general-purpose models in workflows better suited for deterministic computation or specialized tools, primarily to demonstrate AI adoption to stakeholders or customers.

3.4 Theoretical foundations

Our conceptual framework is grounded in the Theory of Planned Behavior (TPB) [1] and Social Cognitive Theory (SCT) [4, 5]. Together, these theories provide complementary explanations for how computational behaviors emerge, spread, and shape computational outcomes in AI-assisted software engineering, and can inform the design of empirical studies in this field.

As described in Section 2.2, TPB explains how individual attitudes, perceived social norms, and perceived behavioral control influence behavioral intentions and observable behavior [1]. In the context of AI-assisted software engineering, TPB helps explain why developers adopt different AI interaction behaviors, including prompting frequency, verification practices, context provision, regeneration strategies, and model selection. From this perspective, computational behaviors are not solely determined by technical necessity, but also by developers' beliefs regarding the usefulness, reliability, and expected value of AI assistance. TPB also highlights the importance of subjective norms within software engineering teams. Developers may adopt computational behaviors not only because they perceive them as effective, but also because such behaviors become socially expected, rewarded by the management, or normalized within work groups. As a result, computational behaviors may spread independently of their actual contribution to collective computational efficiency.

However, TPB's explanatory power has proven to be limited in situations when behaviors are automatic or habitual rather than reasoned [17]. This motivates pairing TPB with SCT.

SCT broadens the socio-technical lens for understanding how computational behaviors evolve collectively through learning, observation, reinforcement, and shared work practices [4, 5]. SCT emphasizes the role of self-efficacy, outcome expectations, and observational learning in shaping human behavior. In AI-assisted software engineering, developers' beliefs regarding their ability to effectively use AI systems influence whether they engage in exploratory prompting, careful verification, iterative refinement, or repetitive low-skill interaction patterns. Similarly, outcome expectations shape assumptions regarding the value and capabilities of AI systems, influencing model selection, escalation to more capable models, and the extent of computational effort invested in AI-assisted tasks. Importantly, SCT also explains how computational behaviors become socially reinforced and amplified within teams and organizations. Developers observe and imitate AI interaction practices used by peers, especially when such behaviors appear productive or are organizationally rewarded. Through repeated observation and reinforcement, individually rational computational behaviors may stabilize into shared collective practices regardless of whether they improve collective computational efficiency. This perspective is particularly important for understanding how computational outcomes emerge at the group and organizational levels.

3.5 Scope

The proposed conceptual framework focuses on computational behaviors emerging during AI-assisted software engineering work within organizational settings. The model is intended as a socio-technical complement to existing systems-level approaches to computational optimization. Therefore, it does not seek to explain computational efficiency through model architectures, hardware infrastructure, inference optimization, or energy-aware computing techniques. Instead, the framework focuses on how human interaction patterns influence organizational consumption, coordination, and reuse of computational resources during software engineering work.

The framework focuses on collective computational outcomes rather than isolated individual productivity measures. Although individual attitudes, mental models, and computational behaviors are included as antecedents, the framework primarily seeks to explain how computational efficiency and computational waste emerge within groups. It is also noteworthy that computational efficiency in our framework is analytically distinct from productivity or task-completion speed. Interventions that reduce computational waste may, in some cases, slow individual task completion or reduce learning opportunities. The framework does not assume these outcomes are always aligned. Conversely, computationally inefficient behaviors may improve learning outcomes, code quality, or long-term team velocity even where they increase immediate computational cost.

Notably, the framework is developed around human-directed interactions. The use of autonomous or semi-autonomous agents that execute multi-step tasks with limited human oversight at each step is an important boundary requiring further extensions. The constructs of computational behavior, waste, and rationality extend naturally to how developers configure, delegate to, and supervise such agents, but the framework does not yet fully address computational dynamics that emerge from agent-to-agent interaction absent a direct human decision point.

Finally, while the framework is developed in the context of AI-assisted software engineering, several of the propositions may generalize to other forms of AI-assisted knowledge work characterized by collaborative workflows and coordinated actions.

4 Proposed Research Agenda

In this section, we outline future research directions on computational efficiency.

4.1 Understanding Individual Human-AI Interaction Behaviors and Computational Outcomes

Industrial relevance. Organizations increasingly recognize that the cost of AI-assisted software engineering depends not only on model pricing but also on developers' everyday interaction behaviors. In one partner organization, developers expressed uncertainty regarding which models were appropriate for different tasks, with some defaulting to expensive models while others experimented with cheaper alternatives. In their case, the difference in cost between high-end and low-end models is noticeable (the former having a 15x request multiplier with usage around 90% of the total AI-related cost compared to 3x for the latter). Practices such as iterative sparring, repeated generation, and autonomous debugging loops can have substantial economic consequences. This becomes especially critical in light of transition from request-based to token-based pricing, where existing interaction patterns are estimated to triple monthly AI-related costs (from around USD 20,000 to USD 65,000). For industry, this makes individual AI literacy, task-model fit, and interaction discipline central concerns for balancing productivity, learning and computational cost.

Research questions. Based on these considerations, we suggest the following directions.

What constitutes wasteful computation? As a starting point, wasteful computation can be operationalized as AI interactions whose outputs are discarded, duplicated, unnecessarily regenerated, or replaceable by cheaper deterministic or specialized alternatives. Empirical work is needed to validate and refine such an operationalization across different software engineering tasks and contexts.

How do self-efficacy beliefs, attitudes, and mental models of AI in general, and agent autonomy in particular, influence computational behavior and delegation decisions? TPB and SCT suggest that developers' attitudes, perceived behavioral control, outcome expectations, and self-efficacy beliefs shape how they interact with AI systems. Future work should investigate which beliefs and mental models contribute to exploratory, coordinated, wasteful, or overly cautious computational behaviors.

4.2 Understanding Collective Human-AI Interaction Behaviors and Collective Computational Outcomes

Industrial relevance. Computational outcomes increasingly emerge from collective patterns of AI use rather than isolated individual interactions. In a partner company, teams organically negotiate model choices, develop informal heuristics of task-model fit taking into account reliability, speed, and cost, and collectively evaluate when expensive models are worthwhile. For example, some developers argued that more capable models may be cheaper in practice because they avoid costly error loops, while others report cases where expensive models still become trapped in unnecessary debugging and backtracking. These experiences suggest that computational efficiency is partly a collective learning and coordination problem: organizations need mechanisms for sharing knowledge about AI capabilities, failure modes, and task suitability, while preserving local autonomy and innovation.

Research questions. Based on these considerations, we suggest the following directions.

Under what conditions does aggregation of individually rational AI use produce collective waste? Individually rational AI interactions may accumulate into collective computational patterns such as duplicated inference, repeated experimentation, fragmented workflows, or computational reuse. Future work should investigate how aggregation mechanisms influence collective computational efficiency.

How can productive redundancy be distinguished from wasteful redundancy in practice? Not all repeated computation is undesirable. Redundancy may support learning, independent verification, resilience, creativity, or exploration. Future work should investigate how organizations can practically distinguish computational redundancy that contributes to collective value from redundancy that merely reproduces already available computational effort.

Which coordination mechanisms reshape these dynamics? Shared AI artefacts (prompts, contexts), collaborative AI workflows and multi-agent orchestration, organizational memory systems, and coordinated AI-assisted development practices may improve computational reuse and reduce duplicated effort. However, tightly coupled AI-assisted collaboration may also amplify coordination overhead, recursive generation, and collective computational waste. Future work shall focus on understanding the costs and benefits of different coordination mechanisms.

4.3 Understanding Organizational Computational Visibility, Alignment and Governance

Industrial relevance. As AI-assisted software engineering scales, organizations increasingly require computational visibility, alignment, and governance. Partner organizations report a transition from exploratory AI adoption toward governed AI operations, including usage dashboards, model cost overviews, and internal adoption metrics. However, visibility into computational behaviors often remains incomplete, particularly regarding duplicated effort,

repeated inference, and interaction patterns across teams. Organizations therefore need governance mechanisms that reduce avoidable computational waste and costs, without suppressing experimentation, learning, and innovation.

Research questions. Based on these considerations, we suggest the following directions.

How can organizations improve visibility into and control over collective computational behaviors on an organizational scale? As AI-assisted work becomes increasingly distributed across individuals, teams, and autonomous agents, organizations require mechanisms for observing, coordinating, reusing, and controlling computational resources across workflows. Future work should investigate how computational visibility systems can support learning, reuse, and computational coordination, and what corporate AI governance practices help control the AI expenditure adherence to budgets.

How do organizational narratives and leadership attitudes toward AI shape computational behaviors? Leadership expectations, AI strategies, and symbolic pressures to demonstrate AI adoption may influence how employees interact with AI systems. Future work should investigate how top-down narratives regarding AI productivity, innovation, or competitiveness shape computational behaviors such as excessive model usage, or performative AI interaction.

How can organizations reduce avoidable computational waste without suppressing learning, exploration, resilience, and innovation? Governance mechanisms aimed at reducing computational expenditure may unintentionally discourage experimentation, independent verification, or exploratory learning. Future work should investigate how organizations can balance computational efficiency with developer autonomy, creativity, resilience, and collective learning.

5 Conclusion

Interest in teams composed of multiple human and AI engineers seamlessly coordinating and combining their work is already high on the research agenda [7]. This is easier said than done. Yet, stakes are concrete: companies like Uber reportedly exhaust its entire yearly AI-coding budget within months of deployment, partially driven by internal tokenmaxxing invitations [14], and single engineer's fleets of 100 autonomous coding agents are capable of generating over a million of dollars in API spent within a single month. The instinctive response, simply curbing AI usage, is itself a trade-off: reducing computational expenditure can just as easily suppress the exploration, verification, and learning that make AI-assisted work valuable in the first place. So, what organizations need is governance that manages computational efficiency rather than blindly minimizes it.

In this vision paper, we argue that computational efficiency in AI-assisted software engineering should be understood not only as a technical property of AI models and infrastructure, but also as an emergent socio-technical dynamics shaped by patterns of human-AI interaction, collective coordination, and organizational governance. As such, our vision extends current discussions of AI efficiency beyond systems-level optimization toward the organizational coordination of AI-assisted work. In particular, we highlight the importance of understanding how computational behaviors aggregate, amplify, and become socially reinforced across teams and workflows, and how organizational structures shape computational visibility, reuse, and governance.

This tension is sharpest where AI participants are no longer simply tools invoked by software engineers but autonomous or semi-autonomous agents that plan, delegate, and act with limited human oversight at each step: autonomy that is locally efficient for a single agent or team can generate hidden computational costs that only surface once agentic behaviors are aggregated and observed at the organizational level.

Our targeted outcomes include terminology, measurement programs, and recommendations for *organizational computational governance*. We plan to collect empirical experiences and derive requirements for computational policies, collective AI resources, collaborative infrastructures, workflow-level AI governance, reasoning budgets, and organizational AI observability. More broadly, we hope our vision will encourage software engineering research to move beyond studying isolated individual AI interactions toward understanding the collective socio-technical dynamics through which computational efficiency and waste can be managed in industrial settings. Ultimately, we do not suggest eliminating computational cost, but distinguishing the redundancy, exploration, and autonomy that generate long-term value from the waste that does not, at both the human and agentic level.

Data Availability Statement

This is a vision paper and reports no empirical data.

References

- 1 I Ajzen. The theory of planned behavior. *Organizational Behavior and Human Decision Processes*, 50(2):179–211, 1991. Theories of Cognitive Self-Regulation. doi:10.1016/0749-5978(91)90020-T.
- 2 LF Wolff Anthony, B Kanding, and R Selvan. Carbontracker: Tracking and predicting the carbon footprint of training deep learning models. *arXiv preprint*, 2020. arXiv:2007.03051.
- 3 Longju Bai, Zhemin Huang, Xingyao Wang, Jiao Sun, Rada Mihalcea, Erik Brynjolfsson, Alex Pentland, and Jiaxin Pei. How do ai agents spend your money? analyzing and predicting token consumption in agentic coding tasks. *arXiv preprint*, 2026. doi:10.48550/arXiv.2604.22750.
- 4 A Bandura. Self-efficacy: Toward a unifying theory of behavioral change. *Advances in Behaviour Research and Therapy*, 1(4):139–161, 1978. Perceived Self-Efficacy: Analyses of Bandura’s Theory of Behavioural Change. doi:10.1016/0146-6402(78)90002-4.
- 5 Albert Bandura. *Social Foundations of Thought and Action: A Social Cognitive Theory*. Prentice-Hall, Englewood Cliffs, NJ, 1986.
- 6 G Bansal, B Nushi, E Kamar, WS Lasecki, DS Weld, and E Horvitz. Beyond Accuracy: The Role of Mental Models in Human-AI Team Performance. *AAAI Conference on Human Computation and Crowdsourcing*, 7:2–11, 2019. doi:10.1609/HCOMP.V7I1.5285.
- 7 A Carleton, D Falessi, H Zhang, and X Xia. Generative ai: Redefining the future of software engineering. *IEEE Software*, 41(6):34–37, 2024. doi:10.1109/MS.2024.3441889.
- 8 L Cruz, J Paulo Fernandes, M H Kirkeby, S Martínez-Fernández, J Sallou, H Anwar, E Barba Roque, J Bogner, J Castaño, F Castor, et al. Greening ai-enabled systems with software engineering: A research agenda for environmentally sustainable ai practices. *ACM SIGSOFT Software Engineering Notes*, 50(3):14–23, 2025.
- 9 B Dietvorst, J Simmons, and C Massey. Algorithm aversion: People erroneously avoid algorithms after seeing them err. *Journal of Experimental Psychology: General*, 144:114–126, 2014. doi:10.1037/xge0000033.
- 10 KI Gero, Z Ashktorab, C Dugan, Q Pan, J Johnson, W Geyer, M Ruiz, S Miller, DR Millen, M Campbell, S Kumaravel, and W Zhang. Mental Models of AI Agents in a Cooperative Game Setting. *Conference on Human Factors in Computing Systems - Proceedings*, April 2020. doi:10.1145/3313831.3376316.
- 11 A Gneezy, U Gneezy, and D O. Lauga. A reference-dependent model of the price–quality heuristic. *Journal of Marketing Research*, 2014. doi:10.1509/jmr.12.04.
- 12 D Jayagopal, J Lubin, and SE Chasins. Exploring the Learnability of Program Synthesizers by Novice Programmers. *UIST 2022 - Proceedings of the 35th Annual ACM Symposium on User Interface Software and Technology*, October 2022. doi:10.1145/3526113.3545659.
- 13 M Jennings, N Deb, and R de Souza Santos. Sustainable ai assistance through digital sobriety. *International Workshop on Responsible Software Engineering*, 2026.

- 14 Tim Keary. Is the cult of tokenmaxxing just another fad or the new normal? *Forbes*, April 2026. URL:<https://www.forbes.com/sites/timkeary/2026/04/13/is-the-cult-of-tokenmaxxingjust-another-fad-or-the-new-normal/>.
- 15 T Kulesza, S Stumpf, M Burnett, and I Kwan. Tell me more? the effects of mental model soundness on personalizing an intelligent agent. *Conference on Human Factors in Computing Systems*, pages 1–10, 2012. doi:10.1145/2207676.2207678.
- 16 G Li, Y Xi, J Ding, D Wang, Z Luo, R Zhang, B Liu, C Fan, X Mao, and Z Zhao. Easy and efficient transformer: Scalable inference solution for large NLP model. In A Loukina, R Gangadharaiyah, and B Min, editors, *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Industry Track*, pages 62–68, Hybrid: Seattle, Washington + Online, July 2022. Association for Computational Linguistics. doi:10.18653/v1/2022.naacl-industry.8.
- 17 Moez Limayem, Sabine Gabriele Hirt, and Christy MK Cheung. How habit limits the predictive power of intention: The case of information systems continuance1. *MIS quarterly*, 31(4):705–737, 2007. URL:<http://misq.org/how-habit-limits-the-predictive-power-of-intention-the-case-of-information-systems-continuance.html>.
- 18 F Lucchetti, Z Wu, A Guha, MQ Feldman, and CJ Anderson. Substance beats style: Why beginning students fail to code with llms. In *Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Vol 1: Long Papers)*, pages 8541–8610, 2025.
- 19 A S Luccioni, S Viguier, and A-L Ligozat. Estimating the carbon footprint of bloom, a 176b parameter language model. *Journal of machine learning research*, 24(253):1–15, 2023. URL: <https://jmlr.org/papers/v24/23-0069.html>.
- 20 AS Luccioni, E Strubell, and K Crawford. From efficiency gains to rebound effects: The problem of jevons’ paradox in ai’s polarized environmental debate. In *ACM Conference on Fairness, Accountability, and Transparency*, pages 76–88, 2025.
- 21 S Luccioni, Y Jernite, and E Strubell. Power hungry processing: Watts driving the cost of ai deployment? In *Proceedings of the 2024 ACM conference on fairness, accountability, and transparency*, pages 85–99, 2024.
- 22 L Meincke, ER Mollick, L Mollick, and D Shapiro. Prompting science report 3: I’ll pay you or i’ll kill you – but will you care? Research paper, The Wharton School, August 2025. doi:10.2139/ssrn.5375404.
- 23 Daye Nam, Ahmed Omran, Ambar Murillo, Saksham Thakur, Abner Araujo, Marcel Blistein, Alexander Frömmgen, Vincent Hellendoorn, and Satish Chandra. Understanding and supporting how developers prompt for llm-powered code editing in practice. *arXiv preprint*, 2025. arXiv:2504.20196.
- 24 S Nguyen, H McLean Babe, Y Zi, A Guha, CJ Anderson, and MQ Feldman. How Beginning Programmers and Code LLMs (Mis) read Each Other. In *CHI Conference on Human Factors in Computing Systems*, pages 1–26, 2024.
- 25 G O’Brien, AP Santos Alves, S Baltes, G Liebel, M Lungu, and M Kalinowski. User misconceptions of llm-based conversational programming assistants. *arXiv preprint*, 2025. arXiv:2510.25662.
- 26 R Parasuraman and D Manzey. Complacency and bias in human use of automation: An attentional integration. *Human factors*, 52:381–410, June 2010. doi:10.1177/0018720810376055.
- 27 P Patel, E Choukse, C Zhang, A Shah, Í Goiri, S Maleki, and R Bianchini. Splitwise: Efficient generative llm inference using phase splitting. In *ACM/IEEE 51st Annual International Symposium on Computer Architecture (ISCA)*, pages 118–132, 2024. doi:10.1109/ISCA59077.2024.00019.
- 28 S Peng, E Kalliamvakou, P Cihon, and M Demirer. The impact of ai on developer productivity: Evidence from github copilot, 2023. arXiv:2302.06590.
- 29 D Russo. Navigating the complexity of generative ai adoption in software engineering. *ACM Trans. Softw. Eng. Methodol.*, 135(5):50, 2024.

- 30 S Samsi, D Zhao, J McDonald, B Li, A Michaleas, M Jones, W Bergeron, J Kepner, D Tiwari, and V Gadepally. From words to watts: Benchmarking the energy costs of large language model inference. In *High Performance Extreme Computing Conference (HPEC)*, pages 1–9. IEEE, 2023.
- 31 R de Souza Santos, C Magalhães, and I Santos. Reconsidering conversational norms in llm chatbots for sustainable ai. *International Workshop on Bots and Agents in Software Engineering*, 2025.
- 32 R Schwartz, J Dodge, N A Smith, and O Etzioni. Green AI. *Communications of the ACM*, 63(12):54–63, 2020. doi:10.1145/3381831.
- 33 Y Sheng, L Zheng, B Yuan, Z Li, M Ryabinin, B Chen, P Liang, C Ré, I Stoica, and C Zhang. Flexgen: high-throughput generative inference of large language models with a single gpu. In *International Conference on Machine Learning*, pages 31094–31116. PMLR, 2023.
- 34 DIK Sjøberg, T Dybå, BCD Anda, and JE Hannay. Building theories in software engineering. In *Guide to Advanced Empirical Software Engineering*, pages 312–336. Springer, 2008. doi:10.1007/978-1-84800-044-5_12.
- 35 LJ Skitka, KL Mosier, and M Burdick. Does automation bias decision-making? *Int. J. Hum.-Comput. Stud.*, 51(5):991–1006, 1999. doi:10.1006/ijhc.1999.0252.
- 36 StackOverflow. Stack overflow developer survey 2025, 2025. URL: <https://survey.stackoverflow.co/2025/AI>.
- 37 V Stray, A Barbala, and V Tellefsen Wivestad. Human-ai collaboration in software development: A mixed-methods study of developers’ use of github copilot and chatgpt. In *Proceedings of the 33rd ACM International Conference on the Foundations of Software Engineering*, pages 1325–1332, 2025.
- 38 V Stray, E Goldmann Brandtzæg, V T Wivestad, A Barbala, and N B Moe. Developer productivity with and without github copilot: A longitudinal mixed-methods case study. *arXiv preprint*, 2025. arXiv:2509.20353.
- 39 L Tankelevitch, V Kewenig, A Simkute, AE Scott, A Sarkar, ASellen, and S Rintel. The Metacognitive Demands and Opportunities of Generative AI. In *CHI Conference on Human Factors in Computing Systems*. Association for Computing Machinery, May 2024. doi:10.1145/3613904.3642902.
- 40 T Vartziotis, M Schmidt, G Dasoulas, I Dellatolas, S Attademo, V D Le, A Wiechmann, T Hoffmann, M Keckeisen, and S Kotsopoulos. Carbon footprint evaluation of code generation through llm as a service. In *International Stuttgart Symposium*, pages 230–241. Springer, 2024.
- 41 X Wang, Y Xiong, Y Wei, M Wang, and L Li. Lightseq: A high performance inference library for transformers. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Industry Papers*, pages 113–120, 2021.
- 42 G Wilkins, S Keshav, and R Mortier. Hybrid heterogeneous clusters can lower the energy consumption of llm inference workloads. In *ACM International Conference on Future and Sustainable Energy Systems*, pages 506–513, 2024.
- 43 J Winsta. The hidden costs of ai: A review of energy, e-waste, and inequality in model development. *arXiv preprint*, 2025. arXiv:2507.09611.
- 44 I Zakharov, E Koshchenko, and A Sergeyuk. Ai in software engineering: Perceived roles and their impact on adoption. In *ACM International Conference on the Foundations of Software Engineering*, FSE Companion ’25, pages 1305–1309. Association for Computing Machinery, 2025. doi:10.1145/3696630.3730563.
- 45 JD Zamfirescu-Pereira, RY Wong, B Hartmann, and Q Yang. Why Johnny Can’t Prompt: How Non-AI Experts Try (and Fail) to Design LLM Prompts. *Conference on Human Factors in Computing Systems - Proceedings*, April 2023. doi:10.1145/3544548.3581388.
- 46 Xinyi Zhou, Zeinadsadat Saghi, Sadra Sabouri, Rahul Pandita, Mollie McGuire, and Souti Chattopadhyay. Cognitive biases in llm-assisted software development, 2026. doi:10.48550/arXiv.2601.08045.