

# Understanding the Human-LLM Dynamic: A Literature Survey of LLM Use in Programming Tasks

Deborah Etsenake  
 detsenake@uwaterloo.ca  
 University of Waterloo  
 Waterloo, Ontario, Canada

Meiyappan Nagappan  
 mei.nagappan@uwaterloo.ca  
 University of Waterloo  
 Waterloo, Ontario, Canada

## ABSTRACT

Large Language Models (LLMs) are transforming programming practices, offering significant capabilities for code generation activities. While researchers have explored the potential of LLMs in various domains, this paper focuses on their use in programming tasks, drawing insights from user studies that assess the impact of LLMs on programming tasks. We first examined the user interaction behaviors with LLMs observed in these studies, from the types of requests made to task completion strategies. Additionally, our analysis reveals both benefits and weaknesses of LLMs showing mixed effects on the human and task. Lastly, we looked into what factors from the human, LLM or the interaction of both, affect the human's enhancement as well as the task performance. Our findings highlight the variability in human-LLM interactions due to the non-deterministic nature of both parties (humans and LLMs), underscoring the need for a deeper understanding of these interaction patterns. We conclude by providing some practical suggestions for researchers as well as programmers.

## CCS CONCEPTS

- General and reference → Surveys and overviews.

## KEYWORDS

Large Language Models, LLMs, User studies, User evaluations, natural language processing, programming tasks, usability studies

### ACM Reference Format:

Deborah Etsenake and Meiyappan Nagappan. 2024. Understanding the Human-LLM Dynamic: A Literature Survey of LLM Use in Programming Tasks. In *Proceedings of ACM Conference (Conference'17)*. ACM, New York, NY, USA, 16 pages. <https://doi.org/10.1145/nnnnnnnnnnnnnnnn>

## 1 INTRODUCTION

In recent years, Large Language Models (LLMs) have surged in prominence. Trained across diverse fields like computing, literature, and psychology, LLMs offer versatility for various tasks, including natural language programming (code generation) and code analysis. This rise in LLM usage has spurred extensive research exploring their efficiency across different tasks [13, 55, 63, 75, 85],

---

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from [permissions@acm.org](mailto:permissions@acm.org).

*Conference'17, July 2017, Washington, DC, USA*  
 © 2024 Association for Computing Machinery.  
 ACM ISBN 978-x-xxxx-xxxx-x/YY/MM...\$15.00  
<https://doi.org/10.1145/nnnnnnnnnnnnnnnn>

human usability[42, 88, 108], and, yet others have leveraged the LLM architecture for specialized applications [29, 48, 69].

Many researchers have evaluated the performance of LLMs for specific programming tasks. ChatGPT, for instance, demonstrated proficiency in passing most introductory programming assessments, although struggling with more complex ones [62, 75]. Interestingly, when compared to students, LLMs have shown superior performance in these introductory programming assessments [32]. LLMs have also proven effective in program repair, surpassing the performance of automated program repair techniques [99]. These are just a few examples of the capabilities of LLMs.

In addition to the evaluations done for LLMs against datasets, the usability of LLMs has also gained attention, prompting evaluations involving humans to gauge real-time performance and practical applications [46, 95, 105]. While some studies suggest LLMs enhance performance in programming tasks and other contexts, other studies indicated mixed or limited benefits [6, 105]. Several user studies have also delved into the interaction patterns of programmers with LLMs [10, 42, 88], unveiling diverse patterns of interaction and prompting strategies among programmers.

In this paper, we conducted a literature survey examining user studies assessing the interaction between humans and LLMs, as well as user studies that investigate the human and task performance effects when LLMs are used by humans across various programming tasks. We aimed to analyze LLM performance across diverse tasks and offer practical guidance for non-experts in machine learning (non-AI experts), when utilizing LLMs for programming tasks as well as shed light on the gaps in user studies for researchers to explore. Specifically, our study addressed the following research questions:

- RQ1:** How are programmers interacting with LLMs (Interaction Observations)?
- RQ2:** Does the use of LLMs lead to enhanced human capabilities (Human Enhancement Evaluation)?
- RQ3:** Does the use of LLMs improve task performance (Task Performance Evaluation)?
- RQ4:** Are there interaction behaviours that lead to enhanced human capabilities or improved task performance (User-LLM Interaction Effects on Human Enhancement and Task Performance)?

By restricting our analysis to observed interactions of the human with the LLM, we gain insight into real-world usage patterns and the effectiveness of LLMs in practical scenarios. This approach provides a more comprehensive understanding of LLMs' capabilities and limitations when utilized by humans.

## 2 RESEARCH METHODOLOGY

To gather data for our study, we focused on LLMs utilizing the transformers architecture [90], widely regarded as the leading framework for LLMs. We conducted a search across various publishers (ACM, IEEE, Scopus and ArXiv), using the following search terms;

---

“ChatGPT” OR “Co-pilot” OR “coPilot” OR “copilot” OR “LLM” OR “LLMs” OR “GPT” OR “large language model” OR “large-language model” OR “large language models” OR “large-language models”  
**AND**  
 “programmers” OR “programmer”  
**AND**  
 “programming” OR “code generation” OR “program synthesis”

---

We intentionally left the search terms to be as broad as possible in order to get a diverse set of papers containing user studies. Despite this, there was still a scarcity of papers. Due to the scarcity of papers, we broadened our scope to encompass experience reports (studies in which the researcher reports on their experience of using an LLM for various tasks), treating them as single-user studies. Our review focused on papers published post-2017, coinciding with the introduction of the transformer architecture. Additionally, we utilized search engine features (where applicable) to refine our results, aiming to restrict them to studies involving human experiments.

We examined the title and abstract of each paper to determine if it met all of our inclusion criteria, which are as follows:

- (1) The paper pertains to programming with LLMs.
- (2) The paper includes a user study or is an experience report.
- (3) The LLM discussed in the paper is based on the transformer architecture.

Additionally, we utilized backward (considering only references published post-2017) and forward snowballing to augment our initial set of papers.

After delving deeper into the papers, beyond just the abstract, we opted to exclude some papers that were either unrelated to the programming process (i.e. not related to code generation, or code explanations), or did not feature observed interactions of the user with the LLM. We also had outlined the following exclusion criteria:

- (1) Studies exploring LLM capabilities using datasets
- (2) Studies that are off focus from evaluating LLMs or enhancing LLM usage.

## 3 GENERAL INSIGHTS

After completing our search, we settled on a set of 88 papers. These user studies spanned various fields of computing and encompassed tasks related to both code generation and code analysis.

We loosely categorized the studies based on objectives and we realized the studies that we found had a diverse set of objectives; some studies were focused on assessing human performance (e.g. productivity, pair programming performance, etc), others evaluated the LLM for specific kind of tasks (e.g code quality, data analysis) or domain of tasks (e.g software engineering, data science, etc). We also noticed that our set of papers had applications in computing education and also in computing industry.

Among the papers discovered, we found that the human studies conducted could be categorized loosely into two major types of

Goal of Study	Number of Papers
Human Enhancement Evaluation	8
Testing or Improving Learning	17
Pair Programming	2
Creative Coding	2
Human-LLM Interaction	18
Evaluating the LLM Response	10
Software Engineering Concepts	16
Code Security	6
Code Translation	2
Data Science	4
Embedded Systems	3

**Table 1: A very loose categorization of the aim of the user studies evaluated in this survey**

experiments: Controlled/Laboratory experiments and Field Experiments.

Within these types of experimentation, there were 3 data analysis methods that we discovered: human enhancement evaluation, Task performance evaluation and interaction analysis. We illustrate this in Table 2 to give a summary of the number of papers that focused on each data analysis method. 43 studies used a combination of methods.

### 3.1 Human Enhancement Evaluation Metrics

This type of data analysis is an evaluation of the effects that using the LLM has on the user. We compiled the enhancement metrics from papers assessing user-LLM performance. Despite slight variations in terminology, we grouped metrics with similar names together. There were 2 main human performance metrics found: time productivity and learning check. We discuss these in detail in section 5. It is important to note that there were more performance metrics assessed within these studies (e.g self-efficacy[18, 80], trust, etc). While we recognize that these metrics are also important, we focused here on human performance metrics identified that could be evaluated objectively.

### 3.2 Task Performance Evaluation Metrics

This type of data analysis is an evaluation of the code produced by the LLM. This includes metrics like code quality, readability, correctness as it relates to the task, etc. We describe these metrics in further detail in section 6.

### 3.3 Human-AI Interaction Data Analysis

We found that 53 papers evaluated the interactive components of the human with the LLMs. In fact, some studies only focused on this. Within this set of studies, there were 3 major themes encountered when reviewing these papers:

- (1) The kind of requests made to the LLM
- (2) How the LLM was prompted for a request type
- (3) How users interacted with the LLM to accomplish their task

We expand on these themes in section 4.

Analysis Technique	References
Interaction Analysis ( <b>RQ1</b> ) 53 papers	Vaithilingam et al. [88], Bird et al. [12], Kazemitaar et al. [41], Qian and Wexler [70], Tan et al. [83], Sun et al. [81], Shoufan [78], Liffiton et al. [49], Kim et al. [45], Kazemitaar et al. [43], Sheese et al. [77], Kuramitsu et al. [48], Jing et al. [37], Yan et al. [102], Vasiliniuc and Groza [89], Denny et al. [22], Ouaazki et al. [61], Feng et al. [29], Tseng et al. [86], Wang et al. [91], Liu et al. [52], Nam et al. [57], Rao et al. [71], Chopra et al. [16], Arawjo et al. [7], Yan et al. [103], Prather et al. [68], Barke et al. [10], Kazemitaar et al. [42], Jiang et al. [36], Mozannar et al. [56], Tang et al. [84], Prasad et al. [67], Arora et al. [8], Nguyen et al. [59], Al Madi [6], Jayagopal et al. [35], Bernstein et al. [11], MacNeil et al. [54], Jury et al. [39], Vaithilingam et al. [87], Ross et al. [72], [107], Wang et al. [94], Khojah et al. [44], Wang et al. [92], Patton et al. [64], Perry et al. [66], Oh et al. [60], Feng and Chen [30], Weisz et al. [95], Gu et al. [33], Chopra et al. [17], Wang et al. [93], Karli et al. [40]
Human Enhancement Evaluation ( <b>RQ2</b> ) 43 papers	Vaithilingam et al. [88], Kazemitaar et al. [41], Peng et al. [65], Liu et al. [50], Tan et al. [83], Yilmaz and Karaoglan Yilmaz [106], Ma et al. [53], Kosar et al. [47], Kim et al. [45], Kuramitsu et al. [48], Xue et al. [101], Jing et al. [37], Yan et al. [102], Vasiliniuc and Groza [89], Denny et al. [22], Ouaazki et al. [61], Imai [34], Feng et al. [29], Wang et al. [91], Yen et al. [105], Ferdowsi et al. [31], Nam et al. [57], Rao et al. [71], Chopra et al. [16], Yan et al. [103], Tang et al. [84], Nguyen et al. [59], Fakhouri et al. [28], Vaithilingam et al. [87], Kim et al. [46], Wang et al. [94], Dirin and Laine [24], Chatterjee et al. [14], Aillon et al. [5], Sandoval et al. [73], Feng and Chen [30], Su et al. [79], Weisz et al. [95], Liu et al. [51], Chopra et al. [17], Johnson et al. [38], Karli et al. [40],
Task Performance Evaluation ( <b>RQ3</b> ) 40 papers	Kazemitaar et al. [41], Peng et al. [65], Liu et al. [50], Qian and Wexler [70], Tan et al. [83], Sun et al. [81], Shoufan [78], Xue et al. [101], Denny et al. [23], Vasiliniuc and Groza [89], Imai [34], Ferdowsi et al. [31], Yan et al. [103], Kazemitaar et al. [42], Jiang et al. [36], Nguyen et al. [59], Fakhouri et al. [28], Al Madi [6], Nejjar et al. [58], Wermelinger [96], Cipriano and Alves [19], Sänger et al. [82], Erhabor et al. [27], Acher and Martinez [4], Su et al. [80], Ross et al. [72], [3], Câmara et al. [21], Wang et al. [94], Wang et al. [92], Choudhuri et al. [18], Chatterjee et al. [14], Aillon et al. [5], Sandoval et al. [73], Perry et al. [66], Asare et al. [9], Weisz et al. [95], Zhou and Li [110], Englhardt et al. [26], Johnson et al. [38],

**Table 2: Summary of research papers categorized by the analysis methods they employed and the corresponding research questions they addressed.**

### 3.4 Categorization of users

During our review, we identified two main categorizations: Industry practitioners versus academia users, and novices versus experts. These distinctions provide a reference for assessing the balance of the papers in this survey and the extent to which the user study results can be generalized.

**3.4.1 Industry Practitioners Vs Academia Users.** This categorization reflects the industry affiliation of participants rather than their expertise in a specific task. Academia users included undergraduates, graduates, and potentially professors (based on experience reports, it's possible that some papers were authored by professors rather than graduate students). Industry practitioners were those actively working in various fields, not limited to computing. While some academia users may have had industry experience, we primarily relied on the information provided in the studies, avoiding speculation and focusing on participant recruitment sources to form this list. Although the title suggests a comparison, this categorization was not used to differentiate how participants were grouped in the experiments. Instead, it served either as a diversity statement to indicate participant group balance or simply to provide context about participant backgrounds. We give a detail of the papers associated with each industry in table 3.

**3.4.2 Novices Vs Experts.** This categorization is based on the users' familiarity with the task. In the evaluated user studies, participants were classified as novices or experts according to their expertise in the task domain. Each study had its own criteria for defining expertise, which could be based on programming language knowledge, task-specific experience, or a combination of both. We adhered to the expertise assessments specified in each paper, focusing on how they related to the task being evaluated.

Notably, we observed that even among groups categorized as "experts" or "novices," there existed variations in expertise levels. To account for this and to provide clearer terminology regarding expertise levels, we adopted the Dreyfus Model of Skill Acquisition [25], although with slight modifications [76]. We categorized "novice" programmers as "novices/beginners" and "expert" programmers as "Competent/Proficient/Experts," aiming to account for skill level variations within the same group. As an abbreviation, we still stick to the terms "novices" and "experts" when referencing the participant pools in this paper.

For single-user studies (experience reports), we grouped participants as experts based on the assumption that some level of expertise is required to write a research paper. For users studies using students taking a course, those users were categorized as novices (unless stated otherwise). We believe this is a reasonable

User Group	Associated Papers
Academia Users	Prather et al. [68], Shoufan [78], Kazemitaar et al. [41], Tan et al. [83], Sun et al. [81], Yilmaz and Karaoglan Yilmaz [106], Ma et al. [53], Kosar et al. [47], Kim et al. [45], Kazemitaar et al. [43], Liffiton et al. [49], Kuramitsu et al. [48], Xue et al. [101], Jing et al. [37], Sheese et al. [77], Denny et al. [23], Yan et al. [102], Denny et al. [22], Ouazki et al. [61], Imai [34], Yen et al. [105], Arawjo et al. [7], Yan et al. [103], Kazemitaar et al. [42], Tang et al. [84], Prasad et al. [67], Arora et al. [8], Nguyen et al. [59], Al Madi [6], Jayagopal et al. [35], Nejjar et al. [58], Wermelinger [96], Cipriano and Alves [19], Bernstein et al. [11], Sänger et al. [82], MacNeil et al. [54], Jury et al. [39], Acher and Martinez [4], Vaithilingam et al. [87], Acher et al. [3], YM et al. [107], Câmara et al. [21], Wang et al. [94], Wang et al. [92], Choudhuri et al. [18], Dirin and Laine [24], Patton et al. [64], Aillon et al. [5], Sandoval et al. [73], Feng and Chen [30], Zhou and Li [110], Gu et al. [33], Johnson et al. [38], Karli et al. [40],
Industry Practitioners	Vaithilingam et al. [88](1user), Peng et al. [65], Qian and Wexler [70], Vasiliniuc and Groza [89], Tseng et al. [86], Wang et al. [91], Liu et al. [52], Rao et al. [71], Chopra et al. [16], Jiang et al. [36], Mozannar et al. [56], Su et al. [80], Ross et al. [72], Khojah et al. [44], Chatterjee et al. [14], Weisz et al. [95], Liu et al. [51], Chopra et al. [17], Wang et al. [93]
Mixed Industries	Feng et al. [29], Ferdowsi et al. [31], Nam et al. [57], Barke et al. [10], Fakhoury et al. [28], Erhabor et al. [27], Perry et al. [66], Oh et al. [60], Asare et al. [9], Su et al. [79], Englhardt et al. [26],

**Table 3: Categorization of user groups based on their industry affiliations.**

Participant Pool Groups	Dreyfus Categories
"Novice"	Novice/Beginner
"Expert"	Competent/Proficient/Expert
"Mixed Group"	All Groups stated above

**Table 4: Participant Groups in Studies and Dreyfus Model Categories**

assumption as students taking an undergraduate course are doing it because of a lack of expertise in the subject of study. 27 studies also decided to used mixed groups, without assessing how expertise affected performance.

We give a list of the papers categorized by expertise level in Table 5.

## 4 RQ1: INTERACTION OBSERVATIONS

**4.0.1 How we addressed it.** We qualitatively answer this research question by examining studies that investigated the interactive component of users and LLMs and collated their findings.

**4.0.2 The Results.** As mentioned in the previous section; there were 3 main themes identified when reviewing the user-LLM interaction data reported in the studies. In this section, we explain them in further detail.

### 4.1 The kind of requests made to the LLM

This category has to do with the type of requests users made of the LLM. A number of studies reported upon this and upon synthesizing all of their reports, we discovered that users made 4 types of inquiries when requesting information from the LLM. In the following explanations, there may be some confusion regarding whether this categorization pertains to individual prompts or different prompting phases. We observed that studies focusing on

interaction analysis (beyond just examining the prompts) describe these categories as phases of interaction, while those concentrating on prompt analysis refer to them as types of requests. Through our analysis and synthesis, we found that these concepts are closely related, so we have combined them in our discussion below.

**4.1.1 Learning/Exploration Requests:** This category/phase includes requests where the user requests information from the LLM in order to gain an understanding of the task/topic/code [8, 10, 43, 44, 48, 61, 66, 67, 72, 77, 89, 91, 101] or exploring ways to implement the task [8, 44, 71]. With these kind of prompts, users were using the LLM as an expert consultant [44] as well as for personalized learning. This form of interaction was shown to involve many steps [17]. In two cases, a user interface (UI) that involved interacting with the LLM beyond the typical chat interface, this type of request was still observed [44, 71]. While not a large percentage of prompts belong to this category [8, 67] (out of the total number of prompts), there were notable cases where users continued prompting the LLM with exploration prompts [7, 91] for the entirety of the task time (for user studies where a specific task was given). This was noted to occur among novice users. It is unclear if the exploration prompting led participants to not complete the task or if they just inquired from the LLM and proceeded to solve the task themselves. One study made note of the fact that the purpose for which users used this sort of request was achieved as they were able to learn something [101]. We can find more insight on whether users are learning with the use of LLMs in our evaluation of human enhancement metrics in section 5.

**4.1.2 Solution-oriented Prompts:** This category/phase includes requests that have to do with generating information specific to the task. A large percentage of prompts were found to belong to this category [11, 44]. Users requests in this case include natural language code implementation details [11, 48, 66, 67, 77, 86, 101], test

User Group	Associated Papers
Novices	Prather et al. [68], Shoufan [78], Kazemitaar et al. [41], Sun et al. [81], Yilmaz and Karaoglan Yilmaz [106], Ma et al. [53], Kosar et al. [47], Kazemitaar et al. [43], Liffiton et al. [49], Kuramitsu et al. [48], Kuramitsu et al. [48], Denny et al. [23], Yan et al. [102], Vasiliniuc and Groza [89], Denny et al. [22], Ouazki et al. [61], Imai [34], Kazemitaar et al. [42], Prasad et al. [67], Nguyen et al. [59], Jayagopal et al. [35], Bernstein et al. [11], MacNeil et al. [54], Jury et al. [39], Kim et al. [46], Choudhuri et al. [18], Dirin and Laine [24], Johnson et al. [38],
Experts	Liu et al. [50], Qian and Wexler [70], Tan et al. [83], Xiao et al. [100], Feng et al. [29], Yen et al. [105], Barke et al. [10], Mozannar et al. [56], Fakhoury et al. [28], Nejjar et al. [58], Wermelinger [96], Cipriano and Alves [19], Sanger et al. [82], Erhabor et al. [27], Acher and Martinez [4], Vaithilingam et al. [87], Su et al. [80], Acher et al. [3], Camara et al. [21], Wang et al. [94], Khojah et al. [44], Wang et al. [92], Chatterjee et al. [14], Aillon et al. [5], Asare et al. [9], Feng and Chen [30], Liu et al. [51], Zhou and Li [110], Gu et al. [33], Chopra et al. [17],
Mixed Group	Vaithilingam et al. [88], Kim et al. [45], Jing et al. [37], Tseng et al. [86], Wang et al. [91], Liu et al. [52], Ferdowsi et al. [31], Nam et al. [57], Rao et al. [71], Chopra et al. [16], Arawjo et al. [7], Yan et al. [103], Jiang et al. [36], Tang et al. [84], Arora et al. [8], Al Madi [6], Ross et al. [72], YM et al. [107], Patton et al. [64], Sandoval et al. [73], Perry et al. [66], Oh et al. [60], Su et al. [79], Weisz et al. [95], Wang et al. [93], Englhardt et al. [26], Karli et al. [40],

**Table 5: User Groups by familiarity with tasks along with the papers that report on this information on participants**

case generation [8], requests to improve on current code [44, 61, 66], code documentation [8], and, pseudocode prompts [42, 66]. These implementation queries varied in terms of details with some being quite broad and others being very specific[86]. With these type of requests, there was some indication of a copying and pasting behaviour where the similarity to the task description was fairly high [8, 22, 42, 66]. This behaviour was observed among studies that made use of students (novices) as participants.

**4.1.3 Error-Correcting Prompts:** This category includes requests where the user requested that the LLM debug their code [8, 43, 77, 94] or they reported an error in the response. In the case of reporting an incorrect response, there were situations where users did not give details about how the LLM should fix the response [77] and there were instances where users would tell the LLM clearly what the issue is and how to correct the error by refining their prompts [48, 61, 103].

**4.1.4 Unrelated to solution Prompts:** There were requests where users created prompts unrelated to the task. These were relational prompts [45, 48] such as thanking the LLM as well as cases where the context did not align with task at hand and thus could not be categorized properly by the researchers [8, 67, 77].

## 4.2 How the LLM was prompted for a request type

This category examines the prompting strategies used by users to elicit desired responses from the LLM. We excluded any prompting patterns identified in experience reports, focusing instead on how programmers without much experience in prompting patterns approach the task of prompting.

With regards to how participants started their prompting interaction with the LLM, there were two main prompting styles.

- **Single Prompt Method:** This can also be referred to as the zero-shot method (to borrow from machine learning

terminology). This was a prompting method where users requested a solution to their issue from the LLM all in one prompt [36, 42, 72, 86].

- **Multi-Prompt Method:** This was a method where users would initially break down tasks (intentionally or unintentionally) and request for a solution to each sub-task in a separate prompt [41, 42, 72].

Some users initially employed the single prompt method [41, 72, 110], while others used the multi-prompt method [26, 42, 72, 110].

This initial categorization of LLM prompting helps us understand how users start out prompting the LLM. In the course of the interaction, users incorporated re-prompting strategies in order to get back on track when the LLM failed to produce the correct response. Note that these prompting patterns we identified from synthesizing the studies are not context-specific and they also don't encompass the entire user task. They generally address the methods that users used to get at a particular solution needed from the LLM, whether it be an implementation solution, debugging issue or learning prompt. These patterns we outline below:

**4.2.1 Re-asking the Question:** This was a prompting pattern in which users tried to get the LLM to regenerate the response by just asking the exact question again[78]. It was identified among novices. Due to the non-deterministic nature of LLMs, this seems to be a valid approach.

**4.2.2 Asking the LLM for verification:** This pattern was also identified among novices [78]. Users, in an attempt to verify the LLM response would ask the LLM to double-check its own response. According to the study done by Shoufan [78], it seemed to be used to getting reassurance from the LLM as well as getting the LLM to double-check its response.

**4.2.3 Task Transformation:** This was a case where participants out of frustration or some other reason, decide to completely change the nature of the task that they wanted the LLM to solve[52, 91].

This method is usually used after several attempts at trying to get the LLM to do something and it failing repetitively.

**4.2.4 Word Restructuring:** With this method, users resorted to rephrasing their prompts in order to get the LLM to elicit the right response [52, 102].

**4.2.5 Elaborating on a task:** This was a prompting pattern where users expanded on a prompt by providing more context or specific details regarding the task. This was the most commonly identified prompting pattern among the studies evaluated in this paper (most likely because of its intuitive nature). It was a prompting pattern identified among novices and experts [26, 42]. Users used this strategy by either breaking down the task to be performed into steps of what to do (a form of multi-prompting) [7, 10, 22, 42, 52, 57, 86, 89, 91] or by providing general high-level context regarding the kind of output the user wanted to see (a form of single prompting) [22, 52, 57, 61, 91, 102].

As expected, when users elaborated on the task, their prompts were much longer [11] as expected. As noted in a few studies, many users struggled with providing the LLM with the context it needed and this was identified among novices[22, 42, 77] and experts [17, 44] and in studies with mixed participants [86]. Since the struggle for providing context for the LLM seems to be universal, there is a need for tools that guide users towards providing more context.

**4.2.6 Reducing Scope:** In this case, users re-prompted the LLM by reducing the scope of the task to be performed[52]. This could imply that they resorted to executing certain parts of the task themselves.

**4.2.7 Combining Prompting patterns.** : Another thing that we identified from our analysis is that users used a combination of techniques to arrive at their solution [91]. This makes sense and is also inline with the non-deterministic nature of LLMs. Users will use whatever methods they can to arrive at their solution. There was one instance [91], where users went through different prompting patterns very quickly. This could be an indication of a poor understanding of the task at hand.

### 4.3 How users interacted with the LLM to accomplish their task

This section deals with observations of users as they interact with the LLM. This covers other aspects of the interaction beyond prompting the LLM and investigates how users accomplish the given task when given the choice of using an LLM.

**4.3.1 Nature of LLM Interaction Observations:** This category has to do with whether users incorporated other tools into their use of LLMs and how they did it. The nature of the LLM interaction fell into 3 categories; Only LLM interaction, Hybrid LLM interaction and no LLM interaction.

**Only LLM Interaction:** This was a situation where users prompted the LLM for the entire solution[36, 40] and refrained from doing any part of the task manually. Users seem to use this strategy if the LLM response was sufficient [36, 39] or in cases where they do not know how to do the task on their own. This type of interaction can indicate a dependence on the LLM [42, 70]. This dependence on the LLM was discovered in primarily two ways:

- (1) Similarity between final LLM output and what users submitted as their response to the task [42, 101].
- (2) The time users spent reviewing the LLM response [6, 103].

**No LLM interaction:** With this type of interaction, users would solve the task completely on their own and would not opt for other resources [42]. There were instances where participants opted for regular programming [36, 42], potentially as a strategy to alleviate the cognitive burden of understanding LLM-generated code. This type of interaction was rarely found among studies probably because of the inherent nature of the experiments [49]; if you allow someone an opportunity to make their life easier, at the very least, they would try to use it. Hence, why this interaction method is rare.

**Hybrid LLM Interaction:** This was where programmers opted for using other resources alongside the LLM [42, 56, 71, 86, 92, 105]. There were also hybrid interaction cases where the interface by which users were interacting with the LLM was hybrid giving users the freedom of doing the task manually or using LLM prompting. Some users used more Natural Language (NL) programming while in other cases, more of regular programming was done [57, 71]. There was also some indication that experts tend to do more regular programming than novices [36, 70]. There were 3 reasons identified as to why users may opt for this type of interaction:

- (1) To alleviate some of the cognitive burden of reviewing LLM response (most common for code generation cases) [36, 42, 64, 80, 88].
- (2) User already knows what they want to do and thus only needs LLM to generate response for certain components of the task which led to time efficiency [29, 86].
- (3) It could just be a preference to want to do some LLM interaction as well as manual work [35, 57]. For example, in cases where the user desires to learn rather than just getting a response.

**4.3.2 Task Time Allocation Observations:** This pertains to how users manage their time across different stages of interaction with the LLM. Users engage in four main steps when interacting with LLMs: planning and prompt crafting, understanding LLM responses, making changes to LLM code, and accepting or rejecting the LLM response. Mozannar et al. [56] provided a comprehensive taxonomy of the programmer's activities during LLM interaction, from which we derived these steps for brevity. The phases of interaction identified by Mozannar et al. [56] does have some overlap with the types of requests made to the LLM.

Across all studies, more time was spent in the planning and prompt crafting phases and in the understanding LLM responses phases, with most studies indicating that users were spending more than 50% of their time in the prompt crafting and understanding phases[17, 56, 84]. Sometimes, this was based on the length of the LLM response [54]. Out of these two phases of interaction, it seems more time was spent in understanding LLM responses rather than in the prompt crafting phase [10, 12, 56, 84, 95]. This is inline with the understanding that the introduction of LLMs has caused users to shift from code writing to more code reviewing. Additionally, there were instances where participants either skipped understanding the LLM response entirely or deferred it to a later stage [56, 88].

**4.3.3 LLM response Review behaviours:** This theme has to do with the way users reviewed the LLM response. With regular programming, we spend time reading and writing but in the era of LLMs, our time is spent more on reading/reviewing the LLM's response [81]. From the previous sections (4.2, 4.1), we could infer that reviewing of LLM response is an important process in human-LLM interaction, hence why we have this section. Through our synthesis from the papers, we observed 4 LLM response review behaviour types:

- (1) Doing it manually: With this method, users opted for manually making edits to the LLM response in order to test it. Some of these behaviours include; playing around with the variables, temporarily removing the code to test its effectiveness, editing syntax, etc [42].
- (2) Doing it via further prompting: Users would work with the LLM to figure out what was wrong [40, 84]. In response to this kind of method, a tool called Robin was introduced to help with this [16].
- (3) Deferring it for later: In this behaviour pattern, users opted to do the verification process later [56, 68, 88, 101]. In these cases, they would just accept the code initially and may later on reject it if discovered to be incorrect [68].
- (4) No Review at all: In this case, users skipped review of the LLM response all together [42]. It was observed in a study with experts that this rarely happens [36].

There are 3 factors that we observed through our synthesis that informed what LLM response review behaviour that users chose to use.

- (1) Proficiency in task: Users who had more experience with the task at hand were less likely to accept LLM code [37, 56, 83]. There are 2 reasons why this could be happening; the LLM's response is not up to par (maybe due to insufficient information in prompt) or users with more expertise just have a set plan on what they wanted to do and if the LLM defers, they would just reject the LLM response. It is also important to note that there was an instance where proficiency in task did not yield positive gains for the task [60].
- (2) Length of LLM response: The length of the LLM response does play a strong role in whether users accepted it upfront [30, 54, 56, 68, 83]. A larger LLM response takes more time to review and it makes sense that users are less likely to want to review the LLM's response thoroughly.
- (3) Usefulness of Suggestion: This has to do with if the LLM response is in line with the users needs. For example, in one study; code completion tools where strings and comments were suggested, were accepted less by users [83].

With these factors, we identified 3 high level suggestions for reducing the cognitive burden of LLM response review;

- (1) Well-timed and relevant suggestions: LLM creators (as well as interface designers) should make sure that the LLM response is shown at the right time and is relevant to the task at hand for the user [33].
- (2) Segmenting the LLM response: By doing this, users are able to take in as much as they can per time and are not overwhelmed [83]. In one study by MacNeil et al.[54], they found

that code explanations that were separated line by line were viewed more by novice users.

### Summary of RQ1:

We observed several key aspects of users' interactions with LLMs. First, users have specific types of requests and employ diverse prompting strategies to address them. Depending on the perceived usefulness of the LLM responses, users might choose to utilize additional resources or abandon LLM prompting altogether if it proves unproductive. During their interactions, a significant amount of time is spent reviewing responses, and users have developed nuanced behaviors to manage this meticulous task of reviewing LLM responses.

## 5 RQ2: HUMAN ENHANCEMENT EVALUATION

**5.0.1 How we addressed it.** To address this research question, our focus was on discovering and synthesizing the performance metrics associated with the human. We grouped similar metrics together and talk about them under one theme (See Table 6).

**5.0.2 The Results.** Overall, significant improvements were observed across various evaluation metrics, with few instances of notable deterioration.

**Time Productivity:** This metric is an assessment on how much users were able to accomplish in a given time period. We discovered that this was the most studied human performance metric. Across all studies, significant gains were discovered in the amount of tasks users were able to complete [22, 24, 41, 79, 83, 89, 103], the amount of time it took them to complete each task [14, 41, 46, 50, 65, 94, 95, 105], the amount of code they wrote [73], pair programming task time [29, 34]. The number of errors encountered per task were also much less [14, 16, 41, 48]. This is an indication that LLMs are good at code syntax.

There were instances where the results of the time evaluation was not significant [5, 28, 31, 40, 50, 87, 88, 95, 101, 102], as well as instances where the use of LLMs had an opposite effect on time productivity. We discovered that this opposite effect was due to the complexity of the task [17, 51, 59, 84, 91] or the nature of participants expertise on the task [46, 57, 71, 83] which resulted in users spending more time trying to understand, comprehend and review the LLM's response. This negative effect on time was also seen in our synthesis of the interaction data (See Section 4). It is important to note as well that the results of each study when assessing time productivity was not all on one side or the other, a few of the studies had mixed results [17, 24, 28, 50, 83, 95, 102].

**Learning Check:** The metric is an assessment of how much and what students had learnt during their LLM interaction. There seems to be a difference in results based on the test design. When tested using a pre-test and post-test scores design (i.e. users were tested before use of LLM and then tested after use of LLM and score difference was calculated), significant gains were observed between LLM use group and no-LLM groups[41, 53, 61, 105, 106]. In cases where students were given access to the LLM during the course/task, no difference in final grades/learning outcomes was

Metric Theme	Method of evaluation used in Paper	Number of Papers
Time Productivity	Number of Tasks Completed	9
	Task completion Time	18
	Lines of Code Written	1
	Number of Errors Encountered	4
	Productivity	1
Learning Check	Pre-test and Post-test Scores	7
	Pre-test and Post-test Completion Time	1
	Course Performance between groups (as Post-test)	2
	Computational thinking measure (Pre and Post tests)	2

**Table 6:** The human enhancement themes discovered from the user studies along with the various methods of evaluation used within the papers and the number of papers that used a specific metric.

observed [45, 47, 101]. In yet another study, where the pre-test and post-test design was used but the post-test accessed ability to write the code manually, the group without LLM use performed better [38]. All of the other studies tested on a variety of things: code comprehension and maybe some code writing but the results of this study shows the negative effects of using LLMs on learning proper programming syntax. Some might argue that the learning of programming syntax is becoming obsolete.

#### Summary of RQ2

Our synthesis revealed that LLMs significantly enhance time productivity and learning. However, it is important to note that in some instances, the productivity improvements might be negligible.

## 6 RQ3: TASK PERFORMANCE EVALUATION

**6.0.1 How we addressed it.** To address this research question, our focus was on evaluating the final submitted human-LLM response at the end of a user session. Specifically, we focused on retrieving the LLM evaluation metrics used in the studies which include metrics like code quality, readability, program speed, etc. We also grouped these metrics into major themes for the sake of brevity (See 7).

**6.0.2 The Results.** We found a number of metrics used to evaluate the human-LLM response. We discuss them below:

**Collaborative Correctness:** This metric measures how closely the submitted LLM response aligns with the expected output. This was measured by rubrics, unit tests, accuracy of response, etc. We discovered a lot of variance in the accuracy of the final response. There were significant gains observed when LLMs were used for a wide variety of tasks from basic programming tasks [4, 27, 36, 41, 58, 80, 81], algorithmic tasks [19, 50, 95], UML modeling [92], software engineering [3–5] and Data analysis [58]. Even more gains were observed for specialized tools and frameworks that helped users with reviewing the LLM response (more specifically code review) [26, 31, 43, 103]. Moreover, there were instances where the performance of the LLM was average [14, 18, 38, 65, 70, 78, 89, 94, 96, 101] and instances where performance was really bad [4, 5, 19, 21, 34, 58, 59, 66, 82, 110]. What we noticed was that situations where either the task evaluated was very specific or the

rubric was not as rigorous, there were a lot more fluctuations in the measured collaborative correctness. In studies where the measures of correctness were rigorous, accessing a variety of things or using a variety of methods, this metric normalized towards being average.

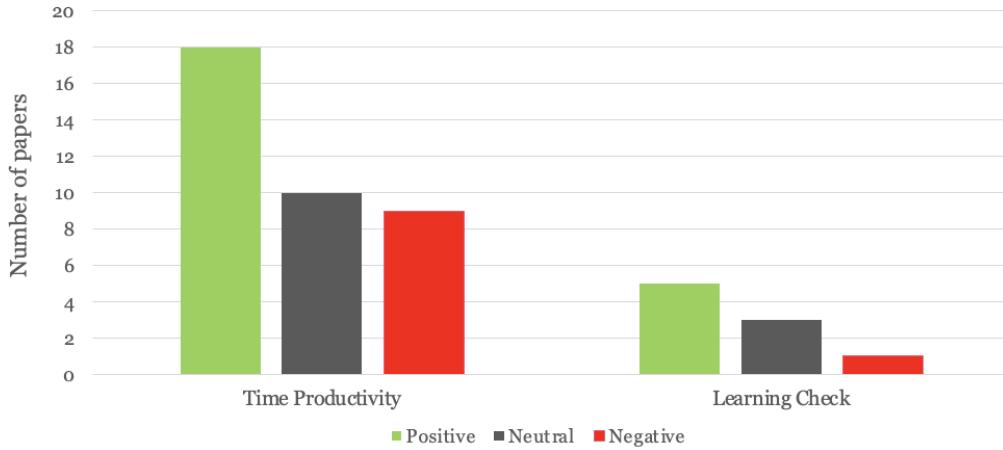
**Code Security and Reliability:** This metric is an evaluation of how secure and reliable the code is under a variety of conditions. It was measured via bugs detected, libraries been used, as well as code smells. The evaluation of this metric from the papers seems to indicate mixed results on the effect of LLMs on code security. A few studies suggest that using LLMs resulted in more secure code being generated [14, 73], while others suggests that the use of LLMs resulted in more insecure code [28, 58, 60, 66]. Still another study suggests that the impact of LLMs on code security are not significant [9]. Despite these results, it is important to recognize that the number of studies where code security and reliability is evaluated explicitly are very few so there is a need for more research into this metric.

**Readability:** This is a measure of how understandable the code is. Using halstead readability metrics, one study did not find any major difference between LLM generated code and human generated code [6]. Another study that looked at this metric suggests though that the readability of a piece of code seems to vary based on the task [58]. There is also some indication that in the code generated by the LLM, there is the possibility of the LLM generating code fragments using concepts unknown to the user [42] thus rendering some parts of the code unreadable to the user without further prompting.

**Program Speed:** This is a measure of how fast the code written with LLM runs. Only one study was found that evaluated this metric and the results of LLM's effects on runtime suggests that LLMs can improve program speed [27].

#### Summary of RQ3

Our synthesis revealed that the effects of LLMs on task performance tends to vary. This is partly due to the diverse nature of tasks evaluated in user studies. More research is needed into this to understand the impact of LLMs on the task performance.



**Figure 1: Human enhancement themes categorized by the number of papers reporting positive, neutral, and negative effects.**

Metric Theme	Method of Evaluation used in Paper	Number of Papers
Collaborative Correctness	Task-Specific Accuracy Score	24
	Success Rate	4
	Unit Tests	2
	Rubric scores	1
	Quality of Solution	3
	Accuracy of Syntax	2
Code Security	Deleted Line counts	1
Readability	Code Smells/Bugs	6
	Halstead measures of Readability	1
	Comprehensibility of Code	1
Program Speed	Complexity of Code	1
	Program Speed	1

**Table 7: The task performance evaluation themes discovered from the studies along with the various methods of evaluation used within papers and the number of papers that used a specific metric**

## 7 RQ4: USER-LLM INTERACTION EFFECTS ON HUMAN ENHANCEMENT AND TASK PERFORMANCE

**7.0.1 How we addressed it.** We address this by looking into studies that reported on human enhancement and task performance and check if any of those studies link an interaction behaviour to positive gains (or losses) in performance.

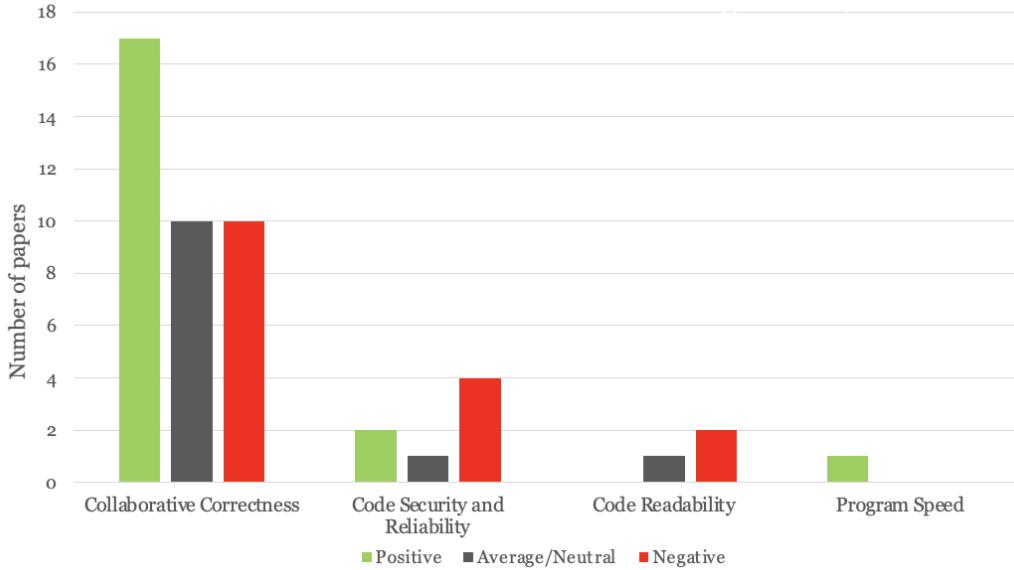
**7.0.2 The Results.** We found some few points indicating the effect of certain interaction behaviours on task performance.

### 7.1 Interaction Effects on Human Enhancement Evaluation

In the analysis of interaction patterns, certain studies observed how specific interaction patterns influenced performance. For instance, users in learning/exploration mode exhibited a learning behavior as expected while using LLMs, although their time productivity decreased. Conversely, users in the implementation mode demonstrated improved time productivity (in some cases).

### 7.2 Interaction Effects on Task Performance

With one re-prompting pattern (elaboration), we found instances where more details yielded positive results [22]. A major interaction behaviour that reflects more positively on the prompting experience is when users are able to identify the errors correctly and also detail the method of rectifying the error [42, 91, 102].



**Figure 2: The LLM response Evaluation metric results as examined in the papers and grouped them into number of papers reporting positive effects, negative and neutral/average effects**

While these findings shed light on the impact of interaction patterns on performance, they remain qualitative and lack methodological rigor. Notably, there is a gap in research regarding the influence of prompting patterns on performance. Further investigation is needed to comprehensively understand how interaction patterns affect performance.

#### Summary of RQ4:

Studies suggest that interaction patterns can influence performance, with learning behaviors decreasing time productivity and implementation behaviors improving it. Additionally, detailed re-prompting and error correction positively affect task performance, but further research is needed to quantify these effects.

## 8 DISCUSSION

Despite the mixed results addressed in this paper, LLMs have been shown to be a powerful tool that aids in programming tasks. In this section, we discuss some of our overall observations as well as opportunities for future research.

### 8.1 A Standard Set of Metrics for Evaluating User-LLM Interactions and Effects

From our analysis in this survey paper, we observed that several studies employed different methods to measure the same metric. For example, time productivity was measured by the number of tasks completed in some studies and by completion time in others. While testing LLMs with users is still relatively new, it would be beneficial to establish a standard set of methods for objectively evaluating each metric. Researchers can then select the method

that best fits their study goals. Based on the findings of this paper, we propose a set of theme metrics along with various evaluation methods for each metric. We also include recommendations on the study design for each method, specifying whether it is suitable for within-subject (WS), between-subject (BS), or both types of designs (see Table 8). We refrain from suggesting task evaluation metrics as these are heavily task-dependent. There are already comprehensive standard metrics available for evaluating code quality (e.g., IEEE software quality metric methodology [1, 2], CERT coding standards [20], etc) and we encourage researchers to use this when assessing task performance (as it relates to code).

### 8.2 Managing the Non-determinism of LLMs

One of the things we know definitively about LLMs is that they are non-deterministic in nature. Thus, figuring out how to interact with LLMs to get the desired response is incredibly valuable.

We've shifted from learning program syntax to learning how to communicate code in natural language, which should come easier since we use natural language daily. However, research shows that users of LLMs struggle with this. It's crucial for humans to understand effective interaction techniques with LLMs to produce the needed response. The section on re-prompting strategies in our response to RQ1 is beneficial for this purpose, serving as a starting point for daily users when interacting with LLMs.

Our analysis in response to RQ1 reveals a paradigm consisting of three considerations when interacting with LLMs:

- (1) **What is the problem you want the LLM to solve?** This directly determines the type of request made to the LLM; whether it is a learning request, implementation request, or error-correcting request.

Metric Theme	Method of evaluation	Definition	Recommended Study Design
Time Productivity	Success Rate	Number of tasks completed	WS, BS
	Completion Time	Amount of time taken to complete the task	BS
	Response Length	Lines of code or number of words in user response	BS
User Frustration	Number of Errors Encountered	A count on the number of instances where users had to deal with an unexpected response from the LLM	WS, BS
Learning Check	Pre-test and Post-test Scores	Test scores taken before (pre-test) and after (post-test) LLM use to assess its impact. The difference in these scores indicates the effect of the LLM.	BS
	Pre-test and Post-test Completion Time	Completion time taken before (pre-test) and after (post-test) LLM use to assess its impact. The difference in these scores indicates the effect of the LLM.	BS
	Computational thinking measure (Pre and Post tests) [111]	Computational thinking score difference before (pre-test) and after (post-test) LLM use to assess its impact.	BS

**Table 8: Suggested human enhancement metric themes along with recommended methods of evaluation. BS means between subjects study design and WS means within subjects study design**

- (2) **How to input the request (or prompt) into the LLM:** Users should choose a prompting style (single or multi-prompt) and a prompting strategy (or strategies) as outlined in 4 as a starting point for prompting or as methods to re-prompt if the LLM fails to produce a correct response. White et al. has also listed out some prompting patterns that can be used as well when trying to get a desired response from the LLM [97].
- (3) In cases where the LLM continues to produce incorrect responses after multiple attempts, users may need consider other resources. The LLM may not have a "correct" answer, or the user's background knowledge on the problem might need improvement to achieve better results.

Starting with these key considerations can enhance the effectiveness of LLM usage. By adopting effective re-prompting strategies and being aware of the nuances in interacting with LLMs, users can better leverage these powerful tools to achieve their desired outcomes. Future research should continue to explore and refine these interaction techniques to maximize the potential of LLMs in various applications.

### 8.3 LLMs for Learning

The research demonstrates that LLMs play a role in facilitating learning about programming techniques and concepts. This literature survey has shown that programmers need not worry excessively

about impeding their learning progress by utilizing LLM tools, as studies indicate that learning still takes place despite their use. However, some of the concerns regarding LLMs often revolve around academic integrity and the potential for plagiarism, prompting educators to seek alternative assessment methods that mitigate these concerns. Presently, educational policies often entail educators prohibiting the utilization of LLMs for assignments. It's essential for students to adhere to existing institutional policies governing LLM usage, although it's worth noting that these policies may evolve over time to reflect a better understanding of LLMs. One approach for educators to address learning concerns is to blend in-class assessments (without LLM usage) with homework assignments where students are permitted to freely utilize LLMs to enhance their understanding of course concepts. Additionally, it is important that learning goals (whether for a course or for self-study) are set properly. There is a difference between using LLMs to learn how to program and using LLMs to understand programming. Users (and educators) need to be cautious of this as the use of LLMs will definitely help students in understanding programs but not necessarily in learning how to program.

## 8.4 Opportunities for Future Work for Researchers

**8.4.1 Investigating the Impact of Interaction Patterns on Performance:** The insights from RQ1 reveal current user-LLM interactions and highlight challenges programmers face. These findings are useful for LLM developers to address these issues. However, a full evaluation of how these interactions impact performance is still missing. Future research can explore the interaction behaviours that influence task performance and human enhancement metrics. Additionally, some studies outside the scope of this paper have discussed LLM interaction patterns, but their effectiveness across different tasks has not been tested [97].

**8.4.2 Validating Findings Across LLM Models:** Additionally, it's essential to acknowledge that certain types of LLMs, such as T5, StarCoder, have not undergone user studies (or at least none were found in our literature survey). Most of the LLMs used in this study were based off ChatGPT or copilot. This underscores the necessity for future research to validate the generalizability of findings across different LLM models. This gap in knowledge presents an opportunity for further exploration to ensure robust insights into LLM usability and performance.

## 9 RELATED WORK

Some literature reviews on LLMs have predominantly addressed the concerns of the machine learning community, focusing on factors such as model size, data quality, and expert tuning, as emphasized by Chen et al. [15]. However, these insights often lack actionable guidance for programmers seeking to integrate LLMs into their workflow efficiently. Similarly, Wong et al. [98] explored downstream tasks enabled by LLMs but lacked a user-centered perspective.

While Zhang et al. [109] extensively covered LLM applications in software engineering, including tasks examined with LLMs, our study takes a distinct approach by delving into user interaction and LLM usability. We specifically investigate task completion abilities and interaction patterns, providing insights into user-centric usability scenarios that complement existing research. Similarly, Yang et al. [104] explored LLM performance metrics and usability studies but did not delve as deeply into non-AI expert usability scenarios as our investigation.

Conversely, Sarkar et al. [74] concentrated on LLM usability for programming tasks, emphasizing user studies and experience reports with a qualitative approach. In contrast, our study employs both qualitative and quantitative methodologies to comprehensively understand how users currently engage with LLMs and identify opportunities for integration into their workflows to enhance task performance.

## 10 THREATS TO VALIDITY

### 10.1 Search Methodology

During our literature survey, we had to choose search terms carefully to find papers on human usability studies. It was tough balancing inclusivity with relevance, as we wanted to avoid getting too many irrelevant papers. However, we managed this challenge by carefully screening search results based on our study criteria.

This helped us find valuable insights while keeping our review on track. We also employed backward snowballing as well as forward snowballing to account for missed search terms. The diversity in the types of papers we found is proof that the strategy is valid.

## 10.2 Qualitative Data Analysis

Aggregating qualitative data from multiple papers presents another potential threat to validity. While there is a risk of overlooking insights, we have addressed this by clearly indicating any patterns with limited evidence and ensuring that omitted insights do not compromise the validity of our results.

## 11 CONCLUSION

In this paper, we conducted a literature survey to examine user studies that assess the interactions between humans and LLMs as well as identify human enhancements and task performance effects as a result of this interaction. Our study addressed four key research questions: the nature of programmers' interactions with LLMs, the enhancement of human capabilities through LLM use, the improvement in task performance due to LLMs, and the interaction behaviors that lead to these enhancements and improvements.

Our analysis revealed diverse objectives and methodologies in existing studies, highlighting the need for standardized metrics to evaluate user-LLM interactions objectively. We identified common human enhancement metrics such as time productivity and learning outcomes and task performance metrics like code quality and correctness. The non-deterministic nature of LLMs necessitates effective interaction techniques, including re-prompting strategies, which as we discussed, are crucial for users to achieve desired responses.

The research also underscored the role of LLMs in facilitating learning about programming techniques and concepts, though concerns about academic integrity remain. Future research opportunities include investigating the impact of specific interaction patterns on performance and validating findings across different LLM models.

In summary, our survey provides valuable insights into user-LLM interactions and their effects on human and task performance. By establishing standard evaluation methods and refining interaction techniques, we can better leverage LLMs' potential in various applications and enhance our understanding of their capabilities and limitations.

## REFERENCES

- [1] 1998. IEEE Standard for a Software Quality Metrics Methodology. <https://standards.ieee.org/standard/1061-1998.html>
- [2] 2014. IEEE Standard for Software Quality Assurance Processes. <https://standards.ieee.org/standard/730-2014.html>
- [3] Mathieu Acher, José Galindo Duarte, and Jean-Marc Jézéquel. 2023. On Programming Variability with Large Language Model-based Assistant. In *Proceedings of the 27th ACM International Systems and Software Product Line Conference - Volume A* (Tokyo, Japan) (*SPLC '23*). Association for Computing Machinery, New York, NY, USA, 8–14. <https://doi.org/10.1145/3579027.3608972>
- [4] Mathieu Acher and Jabier Martinez. 2023. Generative AI for Reengineering Variants into Software Product Lines: An Experience Report. In *Proceedings of the 27th ACM International Systems and Software Product Line Conference - Volume B* (Tokyo, Japan) (*SPLC '23*). Association for Computing Machinery, New York, NY, USA, 57–66. <https://doi.org/10.1145/3579028.3609016>
- [5] Santiago Aillon, Alejandro Garcia, Nicolas Velandia, Daniel Zarate, and Pedro Wightman. 2023. Empirical evaluation of automated code generation for mobile

- applications by AI tools. In *2023 IEEE Colombian Caribbean Conference (C3)*. 1–6. <https://doi.org/10.1109/C358072.2023.10436306>
- [6] Naser Al Madi. 2023. How Readable is Model-Generated Code? Examining Readability and Visual Inspection of GitHub Copilot. In *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering* (Rochester, MI, USA) (*ASE '22*). Association for Computing Machinery, New York, NY, USA, Article 205, 5 pages. <https://doi.org/10.1145/3551349.3560438>
- [7] Ian Arawjo, Chelse Swoopes, Priyan Vaithilingam, Martin Wattenberg, and Elena L. Glassman. 2024. ChainForge: A Visual Toolkit for Prompt Engineering and LLM Hypothesis Testing. In *Proceedings of the CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) (*CHI '24*). Association for Computing Machinery, New York, NY, USA, Article 304, 18 pages. <https://doi.org/10.1145/3613904.3642016>
- [8] Chaitanya Arora, Utkarsh Venaik, Pavit Singh, Sahil Goyal, Jatin Tyagi, Shyama Goel, Ujjwal Singh, and Dhruv Kumar. 2024. Analyzing LLM Usage in an Advanced Computing Class in India. *arXiv:2404.04603 [cs.HC]* <https://arxiv.org/abs/2404.04603>
- [9] Owura Asare, Meiyappan Nagappan, and N. Asokan. 2024. A User-centered Security Evaluation of Copilot. In *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering* (Lisbon, Portugal) (*ICSE '24*). Association for Computing Machinery, New York, NY, USA, Article 158, 11 pages. <https://doi.org/10.1145/3597503.3639154>
- [10] Shraddha Barke, Michael B. James, and Nadia Polikarpova. 2023. Grounded Copilot: How Programmers Interact with Code-Generating Models. *Proc. ACM Program. Lang.* 7, OOPSLA1, Article 78 (apr 2023), 27 pages. <https://doi.org/10.1145/3586030>
- [11] Seth Bernstein, Paul Denny, Juho Leinonen, Lauren Kan, Arto Hellas, Matt Littlefield, Sami Sarsa, and Stephen MacNeil. 2024. "Like a Nesting Doll": Analyzing Recursion Analogies Generated by CS Students using Large Language Models. *arXiv:2403.09409 [cs.HC]* <https://arxiv.org/abs/2403.09409>
- [12] Christian Bird, Denae Ford, Thomas Zimmerman, Nicole Forsgren, Eirini Kalliamvakou, Travis Lowdermilk, and Idan Gazit. 2023. Taking Flight with Copilot: Early insights and opportunities of AI-powered pair-programming tools. *Queue* 20, 6 (Jan 2023), 35–57. <https://doi.org/10.1145/3582083>
- [13] Courtney Byun, Piper Vasicek, and Kevin Seppi. 2023. Dispensing with Humans in Human-Computer Interaction Research. In *Extended Abstracts of the 2023 CHI Conference on Human Factors in Computing Systems* (Hamburg, Germany) (*CHI EA '23*). Association for Computing Machinery, New York, NY, USA, Article 413, 26 pages. <https://doi.org/10.1145/3544549.3582749>
- [14] Sayan Chatterjee, Ching Louis Liu, Gareth Rowland, and Tim Hogarth. 2024. The Impact of AI Tool on Engineering at ANZ Bank An Empirical Study on GitHub Copilot within Corporate Environment. *arXiv:2402.05636 [cs.SE]* <https://arxiv.org/abs/2402.05636>
- [15] Bei Chen, Daoguang Zan, Fengji Zhang, Dianjie Lu, Bingchao Wu, Bei Guan, Yongji Wang, and Jian-Guang Lou. 2023. Large Language Models Meet NL2Code: A Survey. In *ACL 2023*. <https://www.microsoft.com/en-us/research/publication/large-language-models-meet-nl2code-a-survey/>
- [16] Bhavya Chopra, Yasharth Bajpai, Param Biyani, Gustavo Soares, Arjun Radhakrishna, Chris Parnin, and Sumit Gulwani. 2024. Exploring Interaction Patterns for Debugging: Enhancing Conversational Capabilities of AI-assistants. *arXiv:2402.06229 [cs.HC]* <https://arxiv.org/abs/2402.06229>
- [17] Bhavya Chopra, Ananya Singh, Anna Fariha, Sumit Gulwani, Chris Parnin, Ashish Tiwari, and Austin Z. Henley. 2023. Conversational Challenges in AI-Powered Data Science: Obstacles, Needs, and Design Opportunities. *arXiv:2310.16164 [cs.HC]* <https://arxiv.org/abs/2310.16164>
- [18] Rudrajit Choudhuri, Dylan Liu, Igor Steinmacher, Marco Gerosa, and Anita Sarma. 2024. How Far Are We? The Triumphs and Trials of Generative AI in Learning Software Engineering. In *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering* (Lisbon, Portugal) (*ICSE '24*). Association for Computing Machinery, New York, NY, USA, Article 184, 13 pages. <https://doi.org/10.1145/3597503.3639201>
- [19] Bruno Pereira Cipriano and Pedro Alves. 2023. GPT-3 vs Object Oriented Programming Assignments: An Experience Report. In *Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education V. 1* (Turku, Finland) (*ITiCSE 2023*). Association for Computing Machinery, New York, NY, USA, 61–67. <https://doi.org/10.1145/3587102.3588814>
- [20] Computer Emergency Response Team. [n. d.]. CERT Secure Coding Standards. <https://www.securecoding.cert.org/>
- [21] Javier Cámará, Javier Troya, Luis Burgueño, et al. 2023. On the assessment of generative AI in modeling tasks: an experience report with ChatGPT and UML. *Software and Systems Modeling* 22, 3 (2023), 781–793. <https://doi.org/10.1007/s10270-023-01105-5>
- [22] Paul Denny, David H. Smith IV au2, Max Fowler, James Prather, Brett A. Becker, and Juho Leinonen. 2024. Explaining Code with a Purpose: An Integrated Approach for Developing Code Comprehension and Prompting Skills. *arXiv:2403.06050 [cs.HC]* <https://arxiv.org/abs/2403.06050>
- [23] Paul Denny, Juho Leinonen, James Prather, Andrew Luxton-Reilly, Thezyrie Amarouche, Brett A. Becker, and Brent N. Reeves. 2024. Prompt Problems: A New Programming Exercise for the Generative AI Era. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1* (Portland, OR, USA) (*SIGCSE 2024*). Association for Computing Machinery, New York, NY, USA, 296–302. <https://doi.org/10.1145/3626252.3630909>
- [24] Amin Dirin and Teemu Laine. 2024. Examining the Utilization of Artificial Intelligence Tools by Students in Software Engineering Projects. In *CSEDU24*. <https://doi.org/10.5220/0012729400003693>
- [25] S.E. Dreyfus and Hubert Dreyfus. 1980. A Five-Stage Model of the Mental Activities Involved in Directed Skill Acquisition. , 22 pages. <https://apps.dtic.mil/sti/citations/ADA084551#:~:text=In%20acquiring%20a%20skill%20by,%20proficiency%20expertise%20and%20mastery>
- [26] Zachary Englhardt, Richard Li, Dilini Nissanka, Zhihan Zhang, Girish Narayanswamy, Joseph Breda, Xin Liu, Shwetak Patel, and Vikram Iyer. 2023. Exploring and Characterizing Large Language Models For Embedded System Development and Debugging. *arXiv:2307.03817 [cs.SE]*
- [27] Daniel Erhabor, Sreeharsha Udayashankar, Meiyappan Nagappan, and Samer Al-Kiswany. 2023. Measuring the Runtime Performance of Code Produced with GitHub Copilot. *arXiv:2305.06439 [cs.SE]* <https://arxiv.org/abs/2305.06439>
- [28] Sarah Falkhour, Aaditya Naik, Georgios Sakkas, Saikat Chakraborty, and Shuvendu K. Lahiri. 2024. LLM-based Test-driven Interactive Code Generation: User Study and Empirical Evaluation. *arXiv:2404.10100 [cs.SE]* <https://arxiv.org/abs/2404.10100>
- [29] Felicia Li Feng, Ryan Yen, Yuzhe You, Mingming Fan, Jian Zhao, and Zhicong Lu. 2023. CoPrompt: Supporting Prompt Sharing and Referring in Collaborative Natural Language Programming. *arXiv:2310.09235 [cs.HC]*
- [30] Sidong Feng and Chunyang Chen. 2024. Prompting Is All You Need: Automated Android Bug Replay with Large Language Models. In *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering* (Lisbon, Portugal) (*ICSE '24*). Association for Computing Machinery, New York, NY, USA, Article 67, 13 pages. <https://doi.org/10.1145/3597503.3608137>
- [31] Kasra Ferdowsi, Ruanqianqian Huang, Michael B. James, Nadia Polikarpova, and Sorin Lerner. 2023. Live Exploration of AI-Generated Programs. *arXiv:2306.09541 [cs.HC]*
- [32] James Finnie-Ansley, Paul Denny, Brett A. Becker, Andrew Luxton-Reilly, and James Prather. 2022. The Robots Are Coming: Exploring the Implications of OpenAI Codex on Introductory Programming. In *Proceedings of the 24th Australasian Computing Education Conference* (Virtual Event, Australia) (*ACE '22*). Association for Computing Machinery, New York, NY, USA, 10–19. <https://doi.org/10.1145/3511861.3511863>
- [33] Ken Gu, Madeleine Grunde-McLaughlin, Andrew McNutt, Jeffrey Heer, and Tim Althoff. 2024. How Do Data Analysts Respond to AI Assistance? A Wizard-of-Oz Study. In *Proceedings of the CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) (*CHI '24*). Association for Computing Machinery, New York, NY, USA, Article 1015, 22 pages. <https://doi.org/10.1145/3613904.3641891>
- [34] Saki Imai. 2022. Is GitHub Copilot a Substitute for Human Pair-programming? An Empirical Study. In *2022 IEEE/ACM 44th International Conference on Software Engineering: Companion Proceedings* (*ICSE-Companion*). 319–321. <https://doi.org/10.1145/3510454.3522684>
- [35] Dhanya Jayagopal, Justin Lubin, and Sarah E. Chasins. 2022. Exploring the Learnability of Program Synthesizers by Novice Programmers. In *Proceedings of the 35th Annual ACM Symposium on User Interface Software and Technology* (Bend, OR, USA) (*UIST '22*). Association for Computing Machinery, New York, NY, USA, Article 64, 15 pages. <https://doi.org/10.1145/3526113.3545659>
- [36] Ellen Jiang, Edwin Toh, Alejandra Molina, Kristen Olson, Claire Kayacik, Aaron Donsbach, Carrie J Cai, and Michael Terry. 2022. Discovering the Syntax and Strategies of Natural Language Programming with Generative Language Models. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems* (New Orleans, LA, USA) (*CHI '22*). Association for Computing Machinery, New York, NY, USA, Article 386, 19 pages. <https://doi.org/10.1145/3491102.3501870>
- [37] Jong Jing, Hao Wang, Xinyu Chen, et al. 2024. What factors will affect the effectiveness of using ChatGPT to solve programming problems? A quasi-experimental study. *Humanities and Social Sciences Communications* 11, 1 (2024), 319. <https://doi.org/10.1057/s41599-024-02751-w>
- [38] Daniel M. Johnson, William Doss, and Christopher M. Estep. 2024. Using ChatGPT with Novice Arduino Programmers: Effects on Performance, Interest, Self-Efficacy, and Programming Ability. *Journal of Research in Technical Careers* 8, 1 (2024). <https://doi.org/10.9741/2578-2118.1152>
- [39] Breanna Jury, Angela Lorusso, Juho Leinonen, Paul Denny, and Andrew Luxton-Reilly. 2024. Evaluating LLM-generated Worked Examples in an Introductory Programming Course. In *Proceedings of the 26th Australasian Computing Education Conference* (Sydney, NSW, Australia) (*ACE '24*). Association for Computing Machinery, New York, NY, USA, 77–86. <https://doi.org/10.1145/3636243.3636252>
- [40] Ulas Berk Karli, Juo-Tung Chen, Victor Nikhil Antony, and Chien-Ming Huang. 2024. Alchemist: LLM-Aided End-User Development of Robot Applications. In *Proceedings of the 2024 ACM/IEEE International Conference on Human-Robot Interaction* (Boulder, CO, USA) (*HRI '24*). Association for Computing Machinery, New York, NY, USA, 361–370. <https://doi.org/10.1145/3610977.3634969>

- [41] Majeed Kazemitaab, Justin Chow, Carl Ka To Ma, Barbara J. Ericson, David Weintrop, and Tovi Grossman. 2023. Studying the Effect of AI Code Generators on Supporting Novice Learners in Introductory Programming. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems* (Hamburg, Germany) (CHI '23). Association for Computing Machinery, New York, NY, USA, Article 455, 23 pages. <https://doi.org/10.1145/3544548.3580919>
- [42] Majeed Kazemitaab, Xinying Hou, Austin Henley, Barbara J. Ericson, David Weintrop, and Tovi Grossman. 2023. How Novices Use LLM-Based Code Generators to Solve CS1 Coding Tasks in a Self-Paced Learning Environment. arXiv:2309.14049 [cs.HC]
- [43] Majeed Kazemitaab, Runlong Ye, Xiaoning Wang, Austin Zachary Henley, Paul Denny, Michelle Craig, and Tovi Grossman. 2024. CodeAid: Evaluating a Classroom Deployment of an LLM-based Programming Assistant that Balances Student and Educator Needs. In *Proceedings of the CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) (CHI '24). Association for Computing Machinery, New York, NY, USA, Article 650, 20 pages. <https://doi.org/10.1145/3613904.3642773>
- [44] Ranim Khojah, Mazen Mohamad, Philipp Leitner, and Francisco Gomes de Oliveira Neto. 2024. Beyond Code Generation: An Observational Study of ChatGPT Usage in Software Engineering Practice. arXiv:2404.14901 [cs.SE] <https://arxiv.org/abs/2404.14901>
- [45] Nam Wook Kim, Hyung-Kwon Ko, Grace Myers, and Benjamin Bach. 2024. ChatGPT in Data Visualization Education: A Student Perspective. arXiv:2405.00748 [cs.HC] <https://arxiv.org/abs/2405.00748>
- [46] Tae Soo Kim, DaEun Choi, Yoonseoo Choi, and Juho Kim. 2022. Stylette: Styling the Web with Natural Language. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems* (, New Orleans, LA, USA) (CHI '22). Association for Computing Machinery, New York, NY, USA, Article 5, 17 pages. <https://doi.org/10.1145/3491102.3501931>
- [47] Tomaž Kosar, Dragana Ostojić, Yu David Liu, and Marjan Mernik. 2024. Computer Science Education in ChatGPT Era: Experiences from an Experiment in a Programming Course for Novice Programmers. *Mathematics* 12, 5 (2024). <https://doi.org/10.3390/math12050629>
- [48] Kimio Kuramitsu, Yui Obara, Miyu Sato, and Momoka Obara. 2023. KOGI: A Seamless Integration of ChatGPT into Jupyter Environments for Programming Education. In *Proceedings of the 2023 ACM SIGPLAN International Symposium on SPLASH-E* (Cascais, Portugal) (SPLASH-E 2023). Association for Computing Machinery, New York, NY, USA, 50–59. <https://doi.org/10.1145/3622780.3623648>
- [49] Mark Liffiton, Brad E Sheese, Jaromir Savelka, and Paul Denny. 2024. CodeHelp: Using Large Language Models with Guardrails for Scalable Support in Programming Classes. In *Proceedings of the 23rd Koli Calling International Conference on Computing Education Research* (Koli, Finland) (Koli Calling '23). Association for Computing Machinery, New York, NY, USA, Article 8, 11 pages. <https://doi.org/10.1145/3631802.3631830>
- [50] Jinrun Liu, Xinyu Tang, Linlin Li, Panpan Chen, and Yepang Liu. 2023. Which is a better programming assistant? A comparative study between chatgpt and stack overflow. arXiv:2308.13851 [cs.SE] <https://arxiv.org/abs/2308.13851>
- [51] Jiaqi Liu, Fengming Zhang, Xin Zhang, Zhiwen Yu, Liang Wang, Yao Zhang, and Bin Guo. 2024. hmCodeTrans: Human–Machine Interactive Code Translation. *IEEE Transactions on Software Engineering* 50, 5 (2024), 1163–1181. <https://doi.org/10.1109/TSE.2024.3379583>
- [52] Michael Xieyang Liu, Advait Sarkar, Carina Negreanu, Benjamin Zorn, Jack Williams, Neil Toronto, and Andrew D. Gordon. 2023. “What It Wants Me To Say”: Bridging the Abstraction Gap Between End-User Programmers and Code-Generating Large Language Models. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems* (, Hamburg, Germany) (CHI '23). Association for Computing Machinery, New York, NY, USA, Article 598, 31 pages. <https://doi.org/10.1145/3544548.3580817>
- [53] Qianou Ma, Hua Shen, Kenneth Koedinger, and Tongshuang Wu. 2023. HypoCompass: Large-Language-Model-based Tutor for Hypothesis Construction in Debugging for Novices. arXiv:2310.05292 [cs.HC]
- [54] Stephen MacNeil, Andrew Tran, Arth Hellas, Joanne Kim, Sami Sarsa, Paul Denny, Seth Bernstein, and Juho Leinonen. 2023. Experiences from Using Code Explanations Generated by Large Language Models in a Web Software Development E-Book. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1* (, Toronto ON, Canada,) (SIGCSE 2023). Association for Computing Machinery, New York, NY, USA, 931–937. <https://doi.org/10.1145/3545945.3569785>
- [55] Arghavan Moradi Dakhel, Vahid Majdinasab, Amin Nikanjam, Foutse Khomh, Michel C. Desmarais, and Zhen Ming (Jack) Jiang. 2023. GitHub Copilot AI pair programmer: Asset or Liability? *Journal of Systems and Software* 203 (2023), 111734. <https://doi.org/10.1016/j.jss.2023.111734>
- [56] Hussein Mozannar, Gagan Bansal, Adam Fourney, and Eric Horvitz. 2023. Reading Between the Lines: Modeling User Behavior and Costs in AI-Assisted Programming. arXiv:2210.14306 [cs.SE]
- [57] Daye Nam, Andrew Macvean, Vincent Hellendoorn, Bogdan Vasilescu, and Brad Myers. 2024. Using an LLM to Help With Code Understanding. In *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering* (Lisbon, Portugal) (ICSE '24). Association for Computing Machinery, New York, NY, USA, Article 97, 13 pages. <https://doi.org/10.1145/3597503.3639187>
- [58] Mohamed Nejjar, Luca Zacharias, Fabian Stiehle, and Ingo Weber. 2024. LLMs for Science: Usage for Code Generation and Data Analysis. arXiv:2311.16733 [cs.SE] <https://arxiv.org/abs/2311.16733>
- [59] Sydney Nguyen, Hannah McLean Babe, Yangtian Zi, Arjun Guha, Carolyn Jane Anderson, and Molly Q Feldman. 2024. How Beginning Programmers and Code LLMs (Mis)read Each Other. In *Proceedings of the CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) (CHI '24). Association for Computing Machinery, New York, NY, USA, Article 651, 26 pages. <https://doi.org/10.1145/3613904.3642706>
- [60] Sanghak Oh, Kiho Lee, Seonhye Park, Doowon Kim, and Hyoungshick Kim. 2023. Poisoned ChatGPT Finds Work for Idle Hands: Exploring Developers’ Coding Practices with Insecure Suggestions from Poisoned AI Models. arXiv:2312.06227 [cs.CR] <https://arxiv.org/abs/2312.06227>
- [61] Abdessalam Ouazzki, Kristoffer Bergman, and Adrian Holzer. 2023. Leveraging ChatGPT to Enhance Computational Thinking Learning Experiences. In *2023 IEEE International Conference on Teaching, Assessment and Learning for Engineering (TALE)*. 1–7. <https://doi.org/10.1109/TALE56641.2023.10398558>
- [62] Eng Lieh Ouh, Benjamin Kok Siew Gan, Kyong Jin Shim, and Swavek Włodkowski. 2023. ChatGPT, Can You Generate Solutions for My Coding Exercises? An Evaluation on Its Effectiveness in an Undergraduate Java Programming Course.. In *Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education V. 1* (Turku, Finland) (ITiCSE 2023). Association for Computing Machinery, New York, NY, USA, 54–60. <https://doi.org/10.1145/3587102.3588794>
- [63] Omer Said Ozturk, Emre Ekmekcioglu, Orcun Cetin, Budi Arief, and Julio Hernandez-Castro. 2023. New Tricks to Old Codes: Can AI Chatbots Replace Static Code Analysis Tools?. In *Proceedings of the 2023 European Interdisciplinary Cybersecurity Conference* (Stavanger, Norway) (EICC '23). Association for Computing Machinery, New York, NY, USA, 13–18. <https://doi.org/10.1145/3590777.3590780>
- [64] Evan W. Patton, David Y. J. Kim, Ashley Granquist, Robin Liu, Arianna Scott, Jennet Zamanova, and Harold Abelson. 2024. Aptly: Making Mobile Apps from Natural Language. arXiv:2405.00229 [cs.HC] <https://arxiv.org/abs/2405.00229>
- [65] Sida Peng, Eirini Kalliamvakou, Peter Cihon, and Mert Demirer. 2023. The Impact of AI on Developer Productivity: Evidence from GitHub Copilot. arXiv:2302.06590 [cs.SE]
- [66] Neil Perry, Megha Srivastava, Deepak Kumar, and Dan Boneh. 2023. Do Users Write More Insecure Code with AI Assistants?. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security* (Copenhagen, Denmark) (CCS '23). Association for Computing Machinery, New York, NY, USA, 2785–2799. <https://doi.org/10.1145/3576915.3623157>
- [67] Siddhartha Prasad, Ben Greenman, Tim Nelson, and Shriram Krishnamurthi. 2023. Generating Programs Trivially: Student Use of Large Language Models. In *Proceedings of the ACM Conference on Global Computing Education Vol 1* (, Hyderabad, India,) (CompEd 2023). Association for Computing Machinery, New York, NY, USA, 126–132. <https://doi.org/10.1145/3576882.3617921>
- [68] James Prather, Brent N. Reeves, Paul Denny, Brett A. Becker, Juho Leinonen, Andrew Luxton-Reilly, Garrett Powell, James Finnie-Ansley, and Eddie Antonio Santos. 2023. “It’s Weird That It Knows What I Want”: Usability and Interactions with Copilot for Novice Programmers. *ACM Trans. Comput.-Hum. Interact.* (aug 2023). [https://doi.org/10.1145/3617367 Just Accepted](https://doi.org/10.1145/3617367).
- [69] Kevin Pu, Jim Yang, Angel Yuan, Minyi Ma, Rui Dong, Xinyu Wang, Yan Chen, and Tovi Grossman. 2023. DiLogics: Creating Web Automation Programs with Diverse Logics. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology* (, San Francisco, CA, USA,) (UIST '23). Association for Computing Machinery, New York, NY, USA, Article 74, 15 pages. <https://doi.org/10.1145/3586183.3606822>
- [70] Crystal Qian and James Wexler. 2024. Take It, Leave It, or Fix It: Measuring Productivity and Trust in Human-AI Collaboration. In *Proceedings of the 29th International Conference on Intelligent User Interfaces* (Greenville, SC, USA) (IUI '24). Association for Computing Machinery, New York, NY, USA, 370–384. <https://doi.org/10.1145/3640543.3645198>
- [71] Nikitha Rao, Jason Tsay, Kiran Kate, Vincent Hellendoorn, and Martin Hirzel. 2024. AI for Low-Code for AI. In *Proceedings of the 29th International Conference on Intelligent User Interfaces* (Greenville, SC, USA) (IUI '24). Association for Computing Machinery, New York, NY, USA, 837–852. <https://doi.org/10.1145/3640543.3645203>
- [72] Steven I. Ross, Fernando Martinez, Stephanie Houde, Michael Muller, and Justin D. Weisz. 2023. The Programmer’s Assistant: Conversational Interaction with a Large Language Model for Software Development. In *Proceedings of the 28th International Conference on Intelligent User Interfaces* (Sydney, NSW, Australia) (IUI '23). Association for Computing Machinery, New York, NY, USA, 491–514. <https://doi.org/10.1145/3581641.3584037>
- [73] Gustavo Sandoval, Hammond Pearce, Teo Nys, Ramesh Karri, Siddharth Garg, and Brendan Dolan-Gavitt. 2023. Lost at C: A User Study on the Security Implications of Large Language Model Code Assistants. In *32nd USENIX Security*

- Symposium (USENIX Security 23)*. USENIX Association, Anaheim, CA, 2205–2222. <https://www.usenix.org/conference/usenixsecurity23/presentation/sandoval>
- [74] Advait Sarkar, Andrew D. Gordon, Carina Negreanu, Christian Poelitz, Sruti Srinivasa Ragavan, and Ben Zorn. 2022. What is it like to program with artificial intelligence? arXiv:2208.06213 [cs.HC]
- [75] Jaromir Savelka, Arav Agarwal, Christopher Bogart, Yifan Song, and Majd Sakr. 2023. Can Generative Pre-Trained Transformers (GPT) Pass Assessments in Higher Education Programming Courses?. In *Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education V. 1* (Turku, Finland) (ITiCSE 2023). Association for Computing Machinery, New York, NY, USA, 117–123. <https://doi.org/10.1145/3587102.3588792>
- [76] Seth. 2010. Getting smart about the hierarchy of smart. , 12 pages. <https://seths.blog/2010/10/getting-smart-about-the-hierarchy-of-smart/>
- [77] Brad Sheese, Mark Liffiton, Jaromir Savelka, and Paul Denny. 2024. Patterns of Student Help-Seeking When Using a Large Language Model-Powered Programming Assistant. In *Proceedings of the 26th Australasian Computing Education Conference* (Sydney, NSW, Australia) (ACE '24). Association for Computing Machinery, New York, NY, USA, 49–57. <https://doi.org/10.1145/3636243.3636249>
- [78] Abdulhadi Shoufan. 2023. Can Students without Prior Knowledge Use ChatGPT to Answer Test Questions? An Empirical Study. *ACM Trans. Comput. Educ.* (oct 2023). <https://doi.org/10.1145/3628162> Just Accepted.
- [79] Yanqi Su, Dianshu Liao, Zhenchang Xing, Qing Huang, Mulong Xie, Qinghua Lu, and Xiwei Xu. 2024. Enhancing Exploratory Testing by Large Language Model and Knowledge Graph. In *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering* (Lisbon, Portugal) (ICSE '24). Association for Computing Machinery, New York, NY, USA, Article 98, 12 pages. <https://doi.org/10.1145/3597503.3639157>
- [80] Yiming Su, Chengcheng Wan, Utsav Sethi, Shan Lu, Madan Musuvathi, and Suman Nath. 2023. HotGPT: How to Make Software Documentation More Useful with a Large Language Model?. In *Proceedings of the 19th Workshop on Hot Topics in Operating Systems* (Providence, RI, USA) (HOTOS '23). Association for Computing Machinery, New York, NY, USA, 87–93. <https://doi.org/10.1145/3593856.3595910>
- [81] Dongfang Sun, Azzeddine Bouhouaia, Cheng Zhu, et al. 2024. Would ChatGPT-facilitated programming mode impact college students' programming behaviors, performances, and perceptions? An empirical study. *International Journal of Educational Technology in Higher Education* 21, 1 (2024), 14. <https://doi.org/10.1186/s41239-024-00446-5>
- [82] Mario Sänger, Ninon De Mecquenem, Katarzyna Ewa Lewińska, Vasilis Bountris, Fabian Lehmann, Ulf Leser, and Thomas Kosch. 2024. A qualitative assessment of using ChatGPT as large language model for scientific workflow development. *GigaScience* 13 (2024). <https://doi.org/10.1093/gigascience/giae030>
- [83] Xin Tan, Xiao Long, Xianjun Ni, Yinghao Zhu, Jing Jiang, and Li Zhang. 2024. How far are AI-powered programming assistants from meeting developers' needs? arXiv:2404.12000 [cs.SE] <https://arxiv.org/abs/2404.12000>
- [84] Ningzhi Tang, Meng Chen, Zheng Ning, Aakash Bansal, Yu Huang, Collin McMillan, and Toby Jia-Jun Li. 2023. An Empirical Study of Developer Behaviors for Validating and Repairing AI-Generated Code. (3 2023). <https://doi.org/10.1184/R1/22223533.v1>
- [85] Haoye Tian, Weiqi Lu, Tsz On Li, Xunzhu Tang, Shing-Chi Cheung, Jacques Klein, and Tegawende F. Bissyandé. 2023. Is ChatGPT the Ultimate Programming Assistant – How far is it? (2023). arXiv:<https://arxiv.org/abs/2304.11938> [cs.SE]
- [86] Tiffany Tseng, Ruijia Cheng, and Jeffrey Nichols. 2024. Keyframer: Empowering Animation Design using Large Language Models. arXiv:2402.06071 [cs.HC] <https://arxiv.org/abs/2402.06071>
- [87] Priyan Vaithilingam, Elena L. Glassman, Jeevana Priya Inala, and Chenglong Wang. 2024. DynaVis: Dynamically Synthesized UI Widgets for Visualization Editing. In *Proceedings of the CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) (CHI '24). Association for Computing Machinery, New York, NY, USA, Article 985, 17 pages. <https://doi.org/10.1145/3613904.3642639>
- [88] Priyan Vaithilingam, Tianyi Zhang, and Elena L. Glassman. 2022. 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* (New Orleans, LA, USA) (CHI EA '22). Association for Computing Machinery, New York, NY, USA, Article 332, 7 pages. <https://doi.org/10.1145/3491101.3519665>
- [89] Mircea-Serban Vasiliu and Adrian Groza. 2023. Case study: using AI-assisted code generation in mobile teams. In *2023 IEEE 19th International Conference on Intelligent Computer Communication and Processing (ICCP)*. 339–346. <https://doi.org/10.1109/ICCP60212.2023.10398656>
- [90] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems*, I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.), Vol. 30. Curran Associates, Inc. [https://proceedings.neurips.cc/paper\\_files/paper/2017/file/3f5ee243547dee91fb053c1c4a845aa-Paper.pdf](https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fb053c1c4a845aa-Paper.pdf)
- [91] Anqi Wang, Zhizhuo Yin, Yulu Hu, Yuanyuan Mao, and Pan Hui. 2024. Exploring the Potential of Large Language Models in Artistic Creation: Collaboration and Reflection on Creative Programming. arXiv:2402.09750 [cs.HC] <https://arxiv.org/abs/2402.09750>
- [92] Beian Wang, Chong Wang, Peng Liang, Bing Li, and Cheng Zeng. 2024. How LLMs Aid in UML Modeling: An Exploratory Study with Novice Analysts. arXiv:2404.17739 [cs.SE] <https://arxiv.org/abs/2404.17739>
- [93] Chenglong Wang, John Thompson, and Bongshin Lee. 2024. Data Formulator: AI-Powered Concept-Driven Visualization Authoring. *IEEE Transactions on Visualization and Computer Graphics* 30, 1 (2024), 1128–1138. <https://doi.org/10.1109/TVCG.2023.3326585>
- [94] Wei Wang, Hui long Ning, Gaowei Zhang, Libo Liu, and Yi Wang. 2024. Rocks Coding: Not Development—A Human-Centric Experimental Evaluation of LLM-Supported SE Tasks. arXiv:2402.05650 [cs.SE] <https://arxiv.org/abs/2402.05650>
- [95] Justin D. Weisz, Michael Muller, Steven I. Ross, Fernando Martinez, Stephanie Houde, Mayank Agarwal, Kartik Talamadupula, and John T. Richards. 2022. Better Together? An Evaluation of AI-Supported Code Translation. In *27th International Conference on Intelligent User Interfaces* (Helsinki, Finland) (IUI '22). Association for Computing Machinery, New York, NY, USA, 369–391. <https://doi.org/10.1145/3490099.3511157>
- [96] Michel Wermeling. 2023. Using GitHub Copilot to Solve Simple Programming Problems. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1* (, Toronto ON, Canada) (SIGCSE 2023). Association for Computing Machinery, New York, NY, USA, 172–178. <https://doi.org/10.1145/3545945.3569830>
- [97] Jules White, Quchen Fu, Sam Hays, Michael Sandborn, Carlos Olea, Henry Gilbert, Ashraf Elnashar, Jesse Spencer-Smith, and Douglas C. Schmidt. 2023. A Prompt Pattern Catalog to Enhance Prompt Engineering with ChatGPT. arXiv:2302.11382 [cs.SE] <https://arxiv.org/abs/2302.11382>
- [98] Man-Fai Wong, Shangxin Guo, Ching-Nam Hang, Siu-Wai Ho, and Chee-Wei Tan. 2023. Natural Language Generation and Understanding of Big Code for AI-Assisted Programming: A Review. *Entropy* 25, 6 (2023). <https://doi.org/10.3390/e25060888>
- [99] Chunqin Steven Xia, Yuxiang Wei, and Lingming Zhang. 2023. Automated Program Repair in the Era of Large Pre-Trained Language Models. In *Proceedings of the 45th International Conference on Software Engineering* (Melbourne, Victoria, Australia) (ICSE '23). IEEE Press, 1482–1494. <https://doi.org/10.1109/ICSE48619.2023.00129>
- [100] Ruiwei Xiao, Xinying Hou, and John Stamper. 2024. Exploring How Multiple Levels of GPT-Generated Programming Hints Support or Disappoint Novices. In *Extended Abstracts of the 2024 CHI Conference on Human Factors in Computing Systems (CHI EA '24)*. Association for Computing Machinery, New York, NY, USA, Article 142, 10 pages. <https://doi.org/10.1145/3613905.3650937>
- [101] Yuankai Xue, Hanlin Chen, Gina R. Bai, Robert Tairas, and Yu Huang. 2024. Does ChatGPT Help With Introductory Programming? An Experiment of Students Using ChatGPT in CS1. In *Proceedings of the 46th International Conference on Software Engineering: Software Engineering Education and Training* (Lisbon, Portugal) (ICSE-SEET '24). Association for Computing Machinery, New York, NY, USA, 331–341. <https://doi.org/10.1145/3639474.3640076>
- [102] Hao Yan, Thomas D. LaTozo, and Ziyu Yao. 2024. IntelliExplain: Enhancing Interactive Code Generation through Natural Language Explanations for Non-Professional Programmers. arXiv:2405.10250 [cs.HC] <https://arxiv.org/abs/2405.10250>
- [103] Lita Yan, Alyssa Hwang, Zhiyuan Wu, and Andrew Head. 2024. Ivie: Lightweight Anchored Explanations of Just-Generated Code. In *Proceedings of the CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) (CHI '24). Association for Computing Machinery, New York, NY, USA, Article 140, 15 pages. <https://doi.org/10.1145/3613904.3642239>
- [104] Zhou Yang, Zhensu Sun, Terry Zhuo Yue, Premkumar Devanbu, and David Lo. 2024. Robustness, Security, Privacy, Explainability, Efficiency, and Usability of Large Language Models for Code. arXiv:2403.07506 [cs.SE]
- [105] Ryan Yen, Jiawen Zhu, Sangho Suh, Haijun Xia, and Jian Zhao. 2023. CoLadder: Supporting Programmers with Hierarchical Code Generation in Multi-Level Abstraction. arXiv:2310.08699 [cs.SE]
- [106] Ramazan Yilmaz and Fatma Gizem Karaoglan Yilmaz. 2023. The effect of generative artificial intelligence (AI)-based tool use on students' computational thinking skills, programming self-efficacy and motivation. *Computers and Education: Artificial Intelligence* 4 (2023), 100147. <https://doi.org/10.1016/j.caeai.2023.100147>
- [107] Pradyumna YM, Vinod Ganeshan, Dinesh Kumar Arumugam, Meghna Gupta, Nischith Shadagopan, Tanay Dixit, Sameer Segal, Pratyush Kumar, Mohit Jain, and Sriram Rajamani. 2023. Pwr: Exploring the Role of Representations in Conversational Programming. arXiv:2309.09495 [cs.HC]
- [108] J.D. Zamfirescu-Pereira, Richmond Y. Wong, Bjoern Hartmann, and Qian Yang. 2023. Why Johnny Can't Prompt: How Non-AI Experts Try (and Fail) to Design LLM Prompts. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems* (Hamburg, Germany) (CHI '23). Association for Computing Machinery, New York, NY, USA, Article 437, 21 pages. <https://doi.org/10.1145/3545945.3569830>

- 3544548.3581388
- [109] Quanjun Zhang, Chunrong Fang, Yang Xie, Yixin Zhang, Yun Yang, Weisong Sun, Shengcheng Yu, and Zhenyu Chen. 2023. A Survey on Large Language Models for Software Engineering. arXiv:2312.15223 [cs.SE]
- [110] Haquan Zhou and Jingbo Li. 2023. A Case Study on Scaffolding Exploratory Data Analysis for AI Pair Programmers. In *Extended Abstracts of the 2023 CHI Conference on Human Factors in Computing Systems* (, Hamburg, Germany,) (CHI EA '23). Association for Computing Machinery, New York, NY, USA, Article 561, 7 pages. <https://doi.org/10.1145/3544549.3583943>
- [111] Özgen Korkmaz, Recep Çakır, and M. Yaşar Özden. 2017. A validity and reliability study of the computational thinking scales (CTS). *Computers in Human Behavior* 72 (2017), 558–569. <https://doi.org/10.1016/j.chb.2017.01.005>