A Taxonomy for Human-LLM Interaction Modes

Jie Gao
Singapore-MIT Alliance for Research and Technology
Singapore
jie.gao@smart.mit.edu

Erika Lee*
University of California San Diego
San Diego, CA, USA
erl015@ucsd.edu

Zhiyao Shu*
University of California, Berkeley
Berkeley, CA, USA
yaoshu0326@berkeley.edu

Michelle Vaccaro
MIT Center for Collective Intelligence
Massachusetts Institute of Technology
Cambridge, USA
vaccaro@mit.edu

Xiaoxian Zhang*
National University of Singapore
Singapore
xiaoxianzhang@u.nus.edu

Junming Cao
Fudan University
Shanghai, China
21110240004@m.fudan.edu.cn

Thomas Malone
MIT Center for Collective Intelligence
Massachusetts Institute of Technology
Cambridge, USA
malone@mit.edu

ABSTRACT

With the release of ChatGPT, many LLM-powered systems have been designed to expand the ways humans can interact with LLMs, enhancing their capabilities. The interaction between humans and LLMs in these systems represents a paradigm shift compared to traditional human-software interaction. From a software engineering (SE) and human-computer interaction (HCI) perspective, we conducted a systematic literature review across major AI and HCI venues and analyzed the system architectures of human-LLM teams in 267 papers. Through this analysis, we identified the fundamental building blocks: eight key elements and their interfaces, four element behaviors, and a standard human-LLM interaction process. Based on these insights, we developed a taxonomy of seven key interaction modes and 22 submodes. We explored the ideal design space in which these atomic modes can be combined and outlined application scenarios where these modes are applicable. Additionally, we conducted a case study on real-world applications to demonstrate the taxonomy's practical use. We envision this work as a contribution to the theoretical foundations for designing and developing future LLM-powered applications.

CCS CONCEPTS

• **Human-centered computing → HCI theory, concepts and models**.

*This work was done when working as research interns at Singapore-MIT Alliance for Research and Technology.

Conference acronym 'XX, June 03–05, 2018, Woodstock, NY

ACM ISBN 978-1-4503-XXXX-X/18/06
https://doi.org/XXXXXXX.XXXXXXX

KEYWORDS

Large Language Model, Interaction Modes, Human-LLM Interaction, User Interface, Conversational AI, LLM System

ACM Reference Format:
Jie Gao, Erika Lee, Zhiyao Shu, Michelle Vaccaro, Xiaoxian Zhang, Junming Cao, and Thomas Malone. 2018. A Taxonomy for Human-LLM Interaction Modes. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation emai (Conference acronym 'XX)*. ACM, New York, NY, USA, 13 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Every use of computers involves an *interaction mode*—a pattern of interaction between the user and the computer. This concept has evolved significantly, starting from command-line interfaces, advancing to the direct manipulation of on-screen items, and progressing to engaging conversations with chatbots, etc. [51, 52].

In software engineering, patterns are mainly referring to software architecture design patterns [16]–standardized solutions to common problems in software design. These patterns provide reusable blueprints that have proven effective in the past, which developers can readily apply or adapt.

Inspired by software design patterns, we explore the concept of the *human-LLM team*, where humans *interact* with LLMs (via command-line interfaces) or LLM-powered systems (which integrate LLMs as core components). The human-LLM team is similar to software design in that its elements are structured to work together to generate input and output. The key difference is that traditionally, humans are merely users, providing limited and fixed input (e.g., clicking buttons, choosing from dropdown list) to software, relying the software to provide information. In contrast, within an LLM-powered system, humans become both users and elements, generating essential interaction information through prompts. Without prompts, the LLM system can scarcely provide services or follow user guidance.

As both users and integral elements of the system, human-LLM interaction can take a simple form—where humans write prompts, and the LLM processes them to generate responses. However, interactions can also become more complex when LLMs are integrated into systems with additional elements. For example, an LLM may be embedded within a structured user interface that enables users to craft prompts by clicking buttons, minimizing the need to write prompts from scratch.

With the emergence of more LLM-powered systems, such as Cursor[1], Copilot, and Notion AI, a spectrum of interactions has developed, influencing design decisions in the development of these systems. **What are the typical interactions between humans and LLMs in human-LLM teams? Do distinct interaction modes[2] already exist?** While no existing work directly answers these questions, several emerging concepts [17, 36, 53, 60] are highly relevant. Notably, Subramonyam et al. [53] introduce the "design pattern" concept, including many design patterns like "visually track prompts and outputs." This pattern features node-based interfaces that enable users to visualize multiple outputs (nodes), trace their connections (edges), and thus compare and evaluate different input-output variations.

Taking a different perspective from Subramonyam et al., we apply software architecture analysis methods [2, 9] to examine the architecture of human-LLM teams. Our study primarily explores three research questions:

- RQ1: What are the fundamental building blocks of human-LLM teams, including their elements and relationships? (Addressed in Section 4.)
- RQ2: What are the typical ways of organizing these elements into collaborative structures? (Addressed in Section 5.)
- RQ3: How can the taxonomy of interaction modes be applied in practice? (Addressed in Sections 7, 6, and 8.)

To explore our research questions, we conducted a systematic review of papers on LLM-powered systems from major AI, HCI, and SE[3] venues published between 2021 and October 2024. Our analysis of these papers involved three rounds of qualitative coding (1st round: N = 1009; 2nd round: N = 345), resulting in a final corpus of 267 papers. From this, we identified 8 key elements and their interfaces, 4 element behaviors, and a standard human-LLM interaction process. Ultimately, we categorized **7 key interaction modes** and **22 variations** (i.e., submodes) within our taxonomy. We further grouped these modes into 4 primary clusters based on similarities in their usage scenarios. Finally, we explored the design spaces of LLM-powered systems and conducted a case study to demonstrate their potential real-world applications.

In the end, we discuss the conceptual overlap with existing software design patterns. We expect that this taxonomy could serve as a foundation for analyzing the architecture of human-LLM teams, ultimately shaping the design of LLM-powered systems. We envision this work as a theoretical contribution to software engineering for AI. Our data is available in Section 12.

2 DEFINITIONS AND SCOPES

Definitions. To establish a clear communication framework, we define the following key terms used in this paper:

- **LLM-powered system:** A system consisting of the LLM, which provides core intelligence, along with the elements that facilitate user interaction with the LLM.
- **Human-LLM Team:** A team comprising both humans and LLMs, where they interact to complete a task. In this setup, humans are an integral part of the execution process—they must dynamically provide prompts, as the LLM generally cannot proceed with the task autonomously without human input.
- **Human-LLM Interaction**: In a *human-LLM team*, humans and LLMs interact to mutually influence each other's perception of information and collaborate to complete the task.
- **Interaction modes**: The various types of *human-LLM interaction* processes. We intentionally avoid using the term "interaction patterns," as patterns typically refer to reusable, well-established solutions. In contrast, the modes we identify have largely emerged within the past two years. Therefore, we focus specifically on these modes—as they represent current practices—to gain insights into system development and architecture.

Scope on models. While generative AI can be broadly applied to build AI systems, analyzing the entire spectrum of generative AI models is beyond the scope of this work. Therefore, we focus specifically on 1) LLMs and 2) their most relevant models, which we categorize into three groups:

- **Large Language Models (LLMs)**: These models focus on generative AI model which support basic text-to-text generation, like GPT-3, GPT-3.5, Claude, Gemini, Llama 2, OpenAI o1, etc.
- **Multi-modal Large Language Models (MLLMs)**: These models handle interactions that combine text with other modalities, such as images, videos, and audio, like GPT-4, GPT-4o.
- **Models with Similar Input-Output Patterns**: Although this category does not strictly fall under language models, it includes models that share similar input-output patterns with LLMs, such as text-to-text or text-to-image transformations (e.g., Stable Diffusion models). These models are often integrated alongside LLMs in LLM-powered systems.

Scope on Design and Architecture analysis of Human-LLM team. Software architecture typically plays a key role as a bridge between requirements and implementation [19]. Due to differences in focus, there are multiple perspectives for describing software architecture [9]. Our focus is on a high-level conceptual architecture description, similar to the "container diagram" in the C4 Model for visualizing software architecture[4], rather than a detailed, low-level breakdown of specific software implementation components.

3 METHOD

To build our taxonomy of different human-LLM interaction modes, we conducted a systematic literature review following the PRISMA guidelines [39]. Figure 1 illustrates the flow of these steps.

[1]https://www.cursor.com/

[2]While we previously mentioned patterns, we intentionally avoid terms like "interaction patterns" because patterns typically refer to reusable and well-tested modes. Since LLM-powered systems have only recently gained widespread popularity (following the release of ChatGPT), we use "mode" as a more cautious term to describe current interaction dynamics without implying they are fully established or reusable.

[3]But we were unable to find enough relevant papers for our analysis from SE venues.

[4]https://c4model.com/diagrams/component

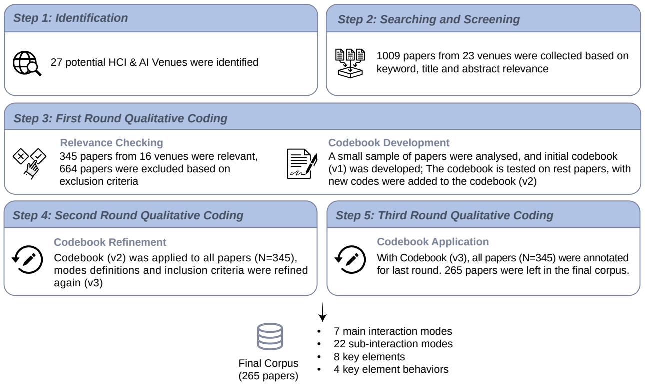

Figure 1: Overview of literature review process.

3.1 Identification process

Given the interdisciplinary nature of studies about human-LLM interactions, we performed our initial search in multiple conferences and journals within a broad list of Software engineering, HCI and AI venues. We found only 27 of them relevant, including major HCI venues like CHI, UIST, CSCW, TOCHI, IUI, DIS, C&C, and CI, and major AI venues like ACL, EMNLP, NAACL, TACL, EACL, IJCAI, CVPR, ICML. While arXiv contains a significant amount of the latest research on LLM-powered applications, we did not include it due to the substantial overlap with papers from other venues. Additionally, we wanted to prioritize peer-reviewed publications.

3.2 Searching and Screening Process

To create the search string used to identify publications, we began by distilling the relevant components of research about human-LLM team. Specifically, these works involve: (1) a human component, and (2) an LLM component, through which the human and LLM interact. We developed a list of synonyms for each component and combined them using Boolean operations. Specifically, we used the following search string:

- **Human**: "Human OR Human-LLM OR Human-AI OR Human-centric OR Human-in-the-loop";
- **LLM**: "Large Language Models OR LLM(s) OR Generative AI OR Prompting OR Prompt(s)";
- **Specific Language Models and Multimodal Language Models**: "GPT OR GPT-4o OR ChatGPT OR Claude (Anthropic) OR Midjourney OR Sora OR Gemini OR DALLE3".

We conducted searches using (1) AND (2) OR (3) across each venue's proceedings or journal issues. We screened titles and abstracts to identify relevant papers. Papers focusing solely on the technical aspects of LLMs without involving human interaction were excluded at this stage. After screening, 1009 papers from 23 venues were included for further eligibility checking. Our final search included papers and posters published between January 2021 and October 2024, following OpenAI's release of GPT-3[5].

3.3 First Round Qualitative Coding (N=1009)

We conducted the first round of qualitative coding for eligibility assessment and the development of coding dimensions, following the collaborative qualitative coding method [46]. Specifically, we began by analyzing a small initial sample of papers to identify preliminary classification categories and primary inclusion criteria. These initial classification categories and inclusion criteria were shared with four authors who participated in qualitative coding, who then performed coding and attended weekly group meetings to resolve conflicts and address ambiguities. Once each author achieved an interrater reliability of over 0.75 (cohen's kappa) with the lead author, the four authors independently coded the remaining papers, discussing ambiguous cases during the weekly team meetings.

In total, 1009 papers underwent first round qualitative coding for relevance checking. At the same time, if a paper was deemed eligible after examining its content, we then performed coding dimension analysis. We summarized the inclusion criteria in Section 3.3.1 and coding dimensions in Section 3.3.2.

3.3.1 Inclusion Criteria. To understand different human-LLM interaction modes, we focused on papers that propose, facilitate, or evaluate processes where humans and LLMs work together to perform a task. Specifically:

- **LLM:** The paper must involve content generation using an LLM or relevant models as described in Section 2. Consequently, papers including new LLM-powered systems were included. In contrast, papers were excluded if they ONLY using traditional language models like BERT for conventional NLP tasks such as text classification or label prediction.
- **Human:** The paper must include human aspects; those that solely evaluate LLM performance based on prompts from non-user sources (e.g., using LLMs for data annotation to assess task performance) were excluded.
- **Interaction:** The paper must describe a clear process for human-LLM interaction, including prompt crafting and response reviewing. Papers without a well-defined interaction process were excluded.

This process of assessing the eligibility of papers resulted in a version of corpus of 345 papers.

3.3.2 Coding Dimensions: We analysed coding dimensions by adopting a software architecture perspective [9, 19] and a process perspective for human-LLM interaction based on the process model concept [37] to understand how humans and LLMs dynamically work and interact to perform tasks. In the end, we identified following key coding dimensions:

- **Element:** A component within an LLM-powered system that possesses autonomy and the ability to perform actions influencing an interaction, e.g., Human, LLM, and other computing components.
- **Behavior:** The specific actions an element can perform when humans and LLMs collaborate on a task. These behaviors manifest through a series of activities within a human-LLM interaction process.
- **Process:** The structured sequence of interactions between humans and an LLM system as they work together to complete a task.
- **Interaction mode:** A distinct type of process through which humans collaborate with LLM systems to accomplish tasks and achieve their goals.

[5]https://openai.com/index/gpt-3-apps/

3.4 Second and Third Qualitative Coding

The **second round of qualitative coding** was conducted because new interaction modes emerged during the initial coding process. This required many initially assigned interaction modes to be reclassified or have their names and meanings revised. As a result, all 345 papers were assigned newly named or reclassified interaction modes.

We then conducted **the third round of qualitative coding** to adopt a multi-mode perspective, allowing each paper to be assigned multiple interaction modes. During the second round of coding, we observed that many papers actually encompassed more than one interaction mode, challenging our initial assumption that each paper had a single primary interaction mode while overlooking others. In the end, each paper was assigned multiple interaction modes as well as multiple elements.

3.5 Final Corpus

After the identification, searching and screening, and three rounds of qualitative coding, we included **267 papers in the final corpus**. Notably, the majority of papers came from ACM CHI, a top conference in the field of human-computer interaction. In the end, a total of 7 main interaction modes and 22 sub-interaction modes were identified. We introduce our main findings in following sections.

4 HUMAN-LLM TEAM BUILDING BLOCKS

We first identify the elements of the human-LLM team, along with their interfaces (inputs and outputs), as well as the behaviors of these elements within the human-LLM interaction process. This examination follows the "view template" used for documenting software architecture [2]. Moreover, we analyze the overall input and output, focusing on how input data—ranging from task goals to LLM-generated responses—is transformed throughout the interaction process. Table 1 showed elements and other details.

4.1 8 Elements and Their Interfaces

First, the elements in the system can be either humans or LLM systems. We identified two key groups: **Humans** (2 elements) and **LLM System** (comprising 2 LLM elements and 4 supporting elements). The primary distinction between humans, LLMs, and other elements lies in their roles within the interaction. Humans and LLMs actively perform behaviors such as crafting prompts, whereas supporting elements primarily assist other elements in completing tasks. For example, components such as the graphical UI, knowledge base, multimodal component, and external tools mainly serve to support the behaviors of the key elements. Table 1 presents their main interfaces, including inputs and outputs.

4.2 4 Key Elements Behaviors

Second, we identified four key behavioral elements in the standard human-LLM interaction process: *planning, generating, evaluating, and revising*. Figure 2 illustrated this process. These behaviors align with those in Norman's seven-stage model of interaction [6]. Since they can occur sequentially in human-LLM interactions, we also use them to represent distinct phases where these behaviors take place. Additionally, as illustrated in Figure 2, the interaction process is often iterative, meaning certain behaviors—such as replanning or regenerating prompts—may repeat. Below, we provide a common definition for each of these four key behaviors:

[6]https://www.educative.io/courses/intro-human-computer-interaction/normans-model-of-interaction

Table 1: Elements and details.

Element	Details
Human	
Single User & User Group	**Input:** Task goals **Output:** Prompts, Revised prompts, Feedback **Behaviors:** Plan, Evaluate, Revise
LLM	
LLM (Single)	**Input:** Prompts, Regeneration requests **Output:** Generated content (text, images, etc.) **Behaviors:** Generate
LLM Team	**Input:** Prompts, Regeneration requests **Output:** Generated content (collaborative generation) **Behaviors:** Generate
Other Components	
Graphical UI	**Input:** User interactions (click, filled content) **Output:** Reformatted output **Behaviors:** (Supporting) Plan, Evaluate, Revise
Knowledge Base	**Input:** Prompts **Output:** Knowledge matched with prompts **Behaviors:** (Supporting) Generate
Multi-modal Component	**Input:** Mixed inputs (text, images, video) **Output:** Multi-modal inputs for LLMs, Readable Content for Humans **Behaviors:** (Supporting) Generate, Evaluate
External Tool	**Input:** Tool usage specifications in prompts **Output:** Enhanced content **Behaviors:** (Supporting) Generate, Evaluate

(1) *Planning*: In this phase, the goal of the prompting interaction is set by the main element (the human), potentially assisted by other elements such as the LLM or a graphical UI, which provides a platform for users to craft prompts. Additionally, multimodal components may help convert certain parts of the user's input into textual information that the LLM can process.
(2) *Generating*: This phase is central to the entire interaction—an LLM or a group of LLMs generates responses to the human's prompts. During this process, they may leverage external tools (e.g., web search) or perform semantic matching with a knowledge base to provide more domain-specific responses.
(3) *Evaluating*: In this phase, users evaluate the LLM's responses and decide whether to accept them, regenerate them, or even replan the prompts. This evaluation may be supported by a graphical UI (e.g., displaying the response), multimodal components (e.g., text-to-audio conversion), or external tools (e.g., a code interpreter) that enhance the presentation of responses.
(4) *Revising*: After evaluating the LLM's response, users may be satisfied with the results but still choose to revise them before use. Even at this stage, users might decide to regenerate responses or replan prompts. This process is often supported by a graphical UI.

4.3 A Standard Human-LLM Interaction Process

With the above key elements and their interfaces and behaviors, we are able to describe a standard or a typical standard human-LLM

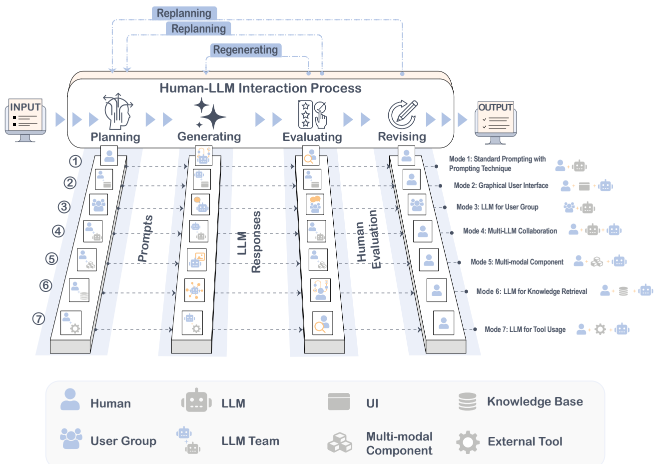

Figure 2: A Standard Human-LLM Interaction Process & Processes for Seven Interaction Modes. This diagram highlights only the key behaviors in their corresponding phases, though each phase may involve additional complexities, such as the inclusion of additional elements. For example, during the planning process, a human might request the LLM to generate prompts.

interaction process, with only a human and an LLM is involved, as opposed to those processes other components are involved. As shown in Figure 2, the process generally begins with inputting a task description, elements (e.g., human user) then plan how the task will be executed, such as proposing prompts to be used in the generation phase. The prompts are then used to generate responses with textual outputs. The human user then compares the generated response against the overall task objectives to determine if it meets the goals. If satisfied with the response, minimal revisions are made, and the results are directly used. If not satisfied, the user may request the LLM to regenerate the response, often revising the prompts to improve subsequent outputs. Finally, the human-LLM interaction concludes, with the evaluated and revised output serving as the final output.

5 TAXONOMY OF INTERACTION MODES

As additional elements are introduced, LLM-powered systems and their interactions with humans become increasingly complex. Since many systems are having mutiple modes as combination modes, we focus on identifying "automatic" modes—those that are independent, indivisible, and mutually exclusive. In particular, Mode 1 represents standard prompting, which does not involve additional elements but serves as the foundational automatic mode for Modes 2–7. Understanding Mode 1 is therefore essential for grasping the entire taxonomy. Just as atoms combine to form molecules, these interaction modes can be integrated to create diverse human-LLM interaction processes. We provide further details in Section 6.

5.1 Mode 1. Standard Prompting with Promting Technique

This mode is exactly the same as the standard human-LLM interaction process. We introduce this mode as a starting point for other interaction modes—much like water serves as the primary component of many other fluids. While it may seem basic, it is crucial for understanding potential variations and extensions of human-LLM interactions.

Elements involved: Human, LLM

Aspects that can vary: The prompting technique used during planning phase.

Variations: The key variations occur in elements behavior during the planning stage, where users can formulate different types of prompts using various prompting techniques. Below, we outline several key variations identified in our literature review:

(1) *Submode 1.1. User/LLM uses reasoning in prompting.* User decomposes the task into logical steps or step-by-step pieces [57, 69]. For example, in a code-repairing task, the process can be divided into code verification and code modification [59, 76]. Similarly, argumentative writing can be broken down into writing a series of individual points, sentences, and paragraphs [73].
(2) *Submode 1.2. User asks LLM to act as a persona.* LLM performance can be enhanced by assigning it a specific role that mimics human expertise, such as an instructor with a sense of humor [30], an expert teacher for communication coaching [21], or even a group discussion participant providing challenging opinions [7].
(3) *Submode 1.3. User incorporates knowledge framework in prompting.* By incorporating a predefined knowledge framework into the prompt, LLM can generate responses that align with that framework. Examples include using a mental health psychology framework [50], a hierarchical storytelling framework that organizes characters, plots, and themes for writing [38], a predefined codebook for qualitative coding [66], and usability heuristics such as Nielsen's heuristics [13].
(4) *Submode 1.4. User is asked for intent clarification by LLM.* LLM can refine its responses by clarifying user intent through iterative questioning. For example, it can progressively adjust generated images to better match user intentions [31] or enhance the accuracy of retrieved knowledge by actively asking clarifying questions to ensure intent alignment [72].
(5) *Submode 1.5. User/LLM designs multiple prompt variations or LLM generates multiple output variations.* The user either designs (or requests the LLM to design) multiple variations of a prompt [4], or the LLM autonomously generates multiple outputs, such as alternative sentence rewrites [23] or qualitative coding suggestions [18].
(6) *Submode 1.6. User gives LLM explicit task context and specific steps.* Context refers to external information beyond the general task description, provided within the prompt to guide the LLM in generating more relevant responses [7]. Prompts that incorporate context are often referred to as the in-context learning technique [10]. This includes specific few-shot examples, structured information about the desired output format, and other explicit instructions. For instance, in AI-assisted code writing tasks [20], specific code contexts are provided to elicit more relevant suggestions from the LLM.
(7) *Submode 1.7. User gives LLM implicit context and general requirements.* In contrast to Submode 1.6, this mode requires the user to provide only minimal context and general requirements as input, relying on the LLM to perform some level of deduction to understand the user's intent. For example, in an in-vehicle

[7] https://www.promptingguide.ai/

conversation scenario, an LLM assistant might infer user intent and take actions such as playing music or adjusting the air conditioning [11].

(8) *Submode 1.8 User iteratively and incrementally refine the output* User uses the modified prompt and output from last step to prompt LLM for further output. For example, users can incrementally refine generated images using previous outputs and modified output [35].

5.2 Mode 2. Interaction with Graphical User Interface

This mode differs from standard prompting because it has additional element, GUI, involved, which increases the complexity of system architecture. This mode differs from standard prompting due to the additional involvement of a GUI, which increases the complexity of the system architecture.

Elements involved: Human, LLM, Graphical User Interface

Aspects that can vary: UI design in the planning stage to support prompt formulation and UI design in the output stage to facilitate evaluation and revision of responses.

Variations: The key variations occur in two stages: planning the input and evaluating and revising the output.

(1) *Submode 2.1. User uses UI for prompting technique design support.* UI design can support prompting technique design in several ways, including: ① *Supporting reasoning*: UI element can enable direct manipulation of intermediate steps in the reasoning chain, allowing human edits before transitioning to the next step. For instance, the UI can provide intermediate suggestions for each bullet point within a prompt chain [63, 64], organize steps through a chain of visual blocks [1], facilitate writing by guiding users from a primary idea to detailed examples or evidence [54, 74], and present solutions with decomposed steps that allow users to repair bugs in intermediate steps [59]. ② *Supporting multiple prompts/outputs generation*: UI design can facilitate the creation of multiple prompts or outputs (Submode 1.5). For example, DesignAID [4] enables an LLM to generate multiple prompts based on a single input, allowing for diverse image generations in later stages. Similarly, UI elements can support iterative writing and experimentation, such as enabling users to explore different story-writing variations using a "tree structure of keywords" [29]. ③ *Supporting knowledge frameworks*: UI can also integrate knowledge frameworks, such as usability heuristics, to facilitate the evaluation of users' design [13].

(2) *Submode 2.2. User uses structured UI for prompt design.* A UI can be designed with structured fields to enable users to customize different parts of a prompt, such as prefixes, settings, and few-shot examples [24, 38]. For example, an LLM system can support personal journaling by allowing users to input keywords and adjust preferences via a slider [28]. Similarly, a system for multimodal generative AI can enable users to customize prompts through various parameter settings, such as color selection and element selection [43].

(3) *Submode 2.3. User uses UI for various output format control.* To enhance user evaluation of LLM outputs, this mode can provide options for customizing visualized code results, such as selecting the preferred size, choosing colors, or adjusting button layouts [25, 26].

(4) *Submode 2.4. User uses UI for collaborative prompt design.* UI design can facilitate collaborative prompt design, allowing multiple users to design prompts together rather than individually [15].

(5) *Submode 2.5. User uses UI for output evaluation, iteration and refinement.* UI design can enable users to evaluate LLM responses and refine or even recraft prompts, supporting the iterative improvement of intermediate or final outputs [3, 33]. For example, in LLM-assisted cooking conversations [70], UI design helps users refine interactions by identifying and labeling errors, as well as regenerating interactions. Additionally, UI design can facilitate the presentation of different evaluation aspects [22].

5.3 Mode 3. Interaction with User Group

This mode differs from standard prompting by incorporating an additional element—User Group—which increases the complexity of both system architecture and interaction.

Elements involved: Human Group, LLM

Aspects that can vary: A group of humans can collaborate with LLM assistance during different stages of the task, such as planning (e.g., facilitating discussions), evaluating and revising (e.g., providing feedback on LLM-generated content).

Variations:

(1) *Submode 3.1. LLM facilitates human group communication.* LLM is designed to facilitate team interactions, such as meeting communication [5, 14, 40], decision-making [75], and collaborative story writing [48].

(2) *Submode 3.2. LLM acts as an equal team member within a human-LLM team.* Instead of directly giving instructions to an LLM, humans and the LLM collaborate as equal partners through natural language communication, with both capable of requesting tasks from each other to achieve the overall goal [61]. In a co-writing task, the LLM and the human can contribute equally to the final output, while the LLM also provide assistance to the user [44].

5.4 Mode 4. Interaction with Multi-LLM Collaboration

This mode differs from standard prompting by incorporating an additional element—a Multi-LLM team—which increases the complexity of both system architecture and interaction.

Elements involved: Human, LLM Group

Aspects that can vary: Humans can collaborate with multiple LLMs instead of a single LLM to perform a task. This approach is often used when a complex task requires multiple LLMs working together, facilitating both the planning and generating phases.

Variations:

(1) *Submode 4.1. User perform tasks by prompting an LLM team.* By assigning each LLM a distinct persona, multiple LLMs can collaborate to handle complex tasks that a single LLM may not perform well [6, 58]. For example, in novel writing, different LLMs can take on roles such as planner, researcher, and writer [6]. In photo editing, multiple LLMs can function as a program manager and a technical expert, each contributing specialized expertise [58]. This approach can even extend to forming a social community of LLM agents [41].

(2) *Submode 4.2. User performs task with teacher and student model team.* This mode leverages the advantages of both small LLMs (student models) and large LLMs (teacher models) by combining the low-cost, lightweight, and locally deployable nature of

student models with the high language understanding and generation capabilities of teacher models. The teacher LLM provides prompt guidance, while the student LLM handles execution. This approach reduces the burden on the remote teacher model and helps maintain data privacy [68, 71].

5.5 Mode 5. Interaction with Multi-modal Components

This mode differs from standard prompting by incorporating an additional element—a multi-modal component—that converts input and output into formats understandable by humans or LLMs, such as processing audio, images, or videos. This increases the complexity of interaction.

Elements involved: Human, LLM, Multi-modal Component

Aspects that can vary: Designing components for LLM input (prompts) and components for converting LLM output into different modalities. This usually happens in planning and evaluating phases.

Variations:

(1) *Submode 5.1. LLM performs task with multimodal components for input, output, or both.* This approach may involve adding a multi-modal encoder to the LLM, enabling it to process and understand visual information [45]. Alternatively, image and audio data can be converted into structured text for the LLM to perform reasoning and predictions [32]. Other techniques include extracting text from images for content analysis with LLM or passing LLM-generated outputs to other generative AI models for image generation [8]. Additionally, an LLM can generate multiple variations of user prompts, which are then used by a stable diffusion model to create diverse images for users [4].

5.6 Mode 6. Interaction with Knowledge Base

This mode differs from standard prompting by incorporating an additional element—a knowledge base—which enables the LLM to respond to human queries using retrieved information from the knowledge base.

Elements involved: Human, LLM, Knowledge Base

Aspects that can vary: Designing different domain-specific knowledge retrieval techniques, such as retrieval-augmented generation (RAG)[8], enables the LLM to generate more expert-level responses by assessing the semantic similarity between the user's prompt and an external knowledge base. This variation primarily occurs during the planning phase.

Variations:

(1) *Submode 6.1. User prompts LLM to retrieve knowledge from knowledge base.* A typical example of default knowledge retrieval is RAG, which can be enhanced with additional prompting techniques, such as actively asking users about their intent in knowledge retrieval to ensure more aligned question answering [72]. Additionally, during the fine-tuning process of an LLM, knowledge retrieval can be used to construct high-quality training dataset, improving the performance of the trained model [67].
(2) *Submode 6.2. User uses knowledge graph to assist LLM knowledge base retrieve retrieval.* Using a knowledge graph to assist LLM responses, where each node represents a concept or entity from the knowledge base [12, 55, 62], reframes knowledge retrieval as a concept-location problem within the knowledge base.
(3) *Submode 6.3. User uses domain model to assist LLM.* Sometimes, the knowledge base is not structured in a traditional format. In such cases, a domain model can be trained on the knowledge base, to provide few-shot examples for LLM prompts through text classification [65]. Additionally, generated knowledge from the LLM can be compared with extracted knowledge from language models to perform knowledge verification [42].

5.7 Mode 7. Interaction with Tool Usage

This mode differs from standard prompting by incorporating an additional element—external tools (e.g., web searching)—which enable the LLM to generate more accurate and up-to-date responses.

Elements involved: Human, LLM, External Tool

Aspects that can vary: Designing different toolsets that involving various functions like web browsing or code execution during the generating process.

Variations:

(1) *Submode 7.1. LLM enhances its output by external tools during generating.* LLM responses are typically generated based on preexisting knowledge, primarily learned from predefined datasets—which may become outdated over time. During the response generation process, whenever the latest information is required (e.g., weather updates or recent tutorials), the LLM can utilize web search results to generate more up-to-date responses [34].

6 TAXONOMY DESIGN SPACE THROUGH COMBINATION OF MAIN MODES

As mentioned in Section 5, our seven interaction modes are atomic, meaning they can be combined to form more complex interaction patterns. As shown in Figure 3, 54.7% of the annotated papers feature multiple interaction modes rather than a single mode per system, highlighting the increasing complexity of LLM-powered system design.

As shown in Figure 4, most combinations occur between submodes under Mode 1 and Mode 2. In particular, Submode 2.2—Structured UI for prompt design—can be integrated with various prompting techniques. For example, a structured UI can help users design prompts by specifying the primary task goal, prefixes, settings, and few-shot examples [24]. It can also be combined with multimodal components, enabling users to manipulate images directly or integrate other modes to support design generation [43].

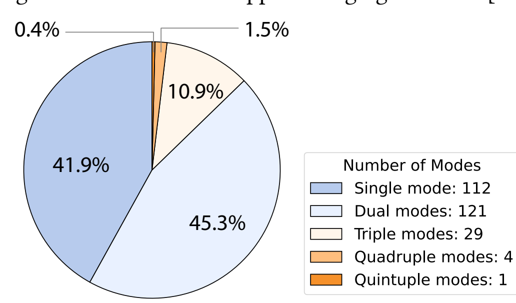

Figure 3: Frequency of combinations

Interestingly, UI is a highly compatible element and can be combined with almost all other elements. For example, users can configure the UI to design multiple prompt techniques supporting multi-LLM teams, integrate multi-modal components, facilitate tool usage, or enable knowledge retrieval.

[8]https://learn.microsoft.com/en-us/dotnet/ai/conceptual/rag

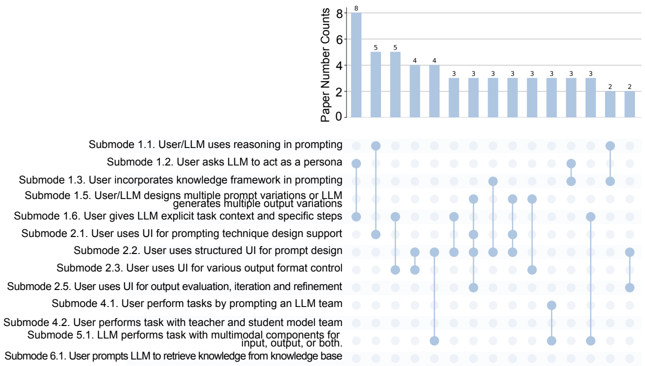

Figure 4: Frequency of combinations.

Moreover, since LLM-powered systems are still evolving, we envision that many more novel mode combinations will emerge, further increasing the complexity of system architecture design and enriching human-LLM interactions.

How many design possibilities exist in the entire design space? If we consider all possible mode combinations, the calculation of the total number of submode combinations can be regarded as a combinatorial enumeration problem of subsets of size 22 (22 submodes). The total number of subsets of a set with 22 elements is given by $\sum_{r=0}^{22}\binom{22}{r} = 2^{22}$. Excluding the empty set (where $r = 0$, $\binom{22}{0} = 1$) and the single-element subsets (where $r = 1$, $\binom{22}{1} = 22$), the total number of possible mode combinations is: $2^{22} - 1 - 22 = 4{,}194{,}304 - 23 = 4{,}194{,}281$. Thus, there are **4,194,281 possible mode combinations**. This number is significantly larger than what is currently observed (assuming no mutually exclusive modes). Consequently, this vast design space could inspire much greater diversity in LLM-powered system design.

7 APPLICATION: TAXONOMY AS A REFERENCE FOR DIFFERENT USAGE PURPOSES

While the 22 submodes were categorized into seven distinct modes, many share similar purposes. We identified four broader usage scenarios to clarify the relationships between different modes. When LLM-powered system developers, designers, product managers, and other potential users of this taxonomy seek suitable submodes, they may start with a basic one and explore alternatives that serve similar purposes. The usage scenarios for interaction modes are illustrated in Figure 5.

7.1 For Personalization

As shown in Usage 1 branch in Figure 5, this usage involves augmenting LLMs in various ways to enhance personalization and task relevance, ensuring they better align with users' goals. This includes incorporating specific domain expertise or tailoring responses based on user preferences.

7.1.1 ***For personalizing user preferences.*** This usage allows users to shape an LLM's behavior and responses through human-like interaction with specific personas, such as preferred communication styles, to better align with their preferences. This may require users to take the initiative in understanding the attributes of the desired persona and articulating them in the prompt.

7.1.2 ***For personalizing LLM expertise.*** This usage aims to augment LLMs with external expertise to generate more informed and expert-aligned outputs. This can be achieved through various methods, such as incorporating knowledge frameworks into prompts, retrieving information from external knowledge bases, utilizing knowledge graph-supported retrieval, or leveraging domain models to provide professional few-shot examples and contextual guidance. However, this approach may require a high level of expertise, as users must understand retrieval mechanisms to construct effective prompts.

7.2 For Iteration

As shown in the Usage 2 branch of Figure 5, iteration usage focuses on improving prompt alignment and refining LLM outputs.

7.2.1 ***For iterating to align input intent.*** This usage ensures that the LLM aligns with the user's intended task goals, making Intent Alignment particularly suitable for users with clear objectives and structured workflows. Alignment can be achieved in several ways: the LLM may request intent clarification, users may provide task context or explicit step-by-step instructions, or users may offer only implicit requirements, relying on the LLM to iteratively refine its responses through trial and error to match their preferences or make decisions.

7.2.2 ***For iterating to improve output.*** To refine results, these modes enable the generation of multiple prompt/output variations, allowing users to compare them synchronously. Alternatively, users can iteratively and incrementally refine outputs, with the system providing mechanisms to decompose tasks upfront and continuously enhance results. This process can also be facilitated through the UI, with improvements typically applied to the LLM's output. Results improvement is particularly beneficial for tasks requiring ongoing adjustments.

7.3 For Collaboration

As shown in Usage 3 in Figure 5, this usage enhances the LLM's capability to facilitate or participate in collaborative workflows, whether among humans, between humans and LLMs, or among LLMs themselves. Collaboration can occur at various stages of a task, such as planning, generating, revising, and evaluating. Developers seeking to explore different roles an LLM can assume—whether as an active participant or an objective facilitator—can consider factors such as which planning phase it engages in, whether humans or the LLM take initiative, or how mixed-initiative interactions unfold at different stages.

These interaction modes share the broad goal of enabling effective communication and cooperative output generation, yet differ significantly in their focus on the roles and contributions of humans and LLMs. We identified three main types of collaboration:

1) One LLM, a group of users. For collaborative prompt design or group communication, LLMs can also serve as objective facilitators, bridging gaps between individuals by summarizing discussions, highlighting key points, and resolving ambiguities to help teams reach a common consensus. This is particularly useful in scenarios such as cross-functional team meetings or decision-making processes, where clarity and agreement are crucial. In such cases, the LLM may need to take a more proactive yet objective facilitation role, actively shaping the team's direction and guiding the problem-solving process.

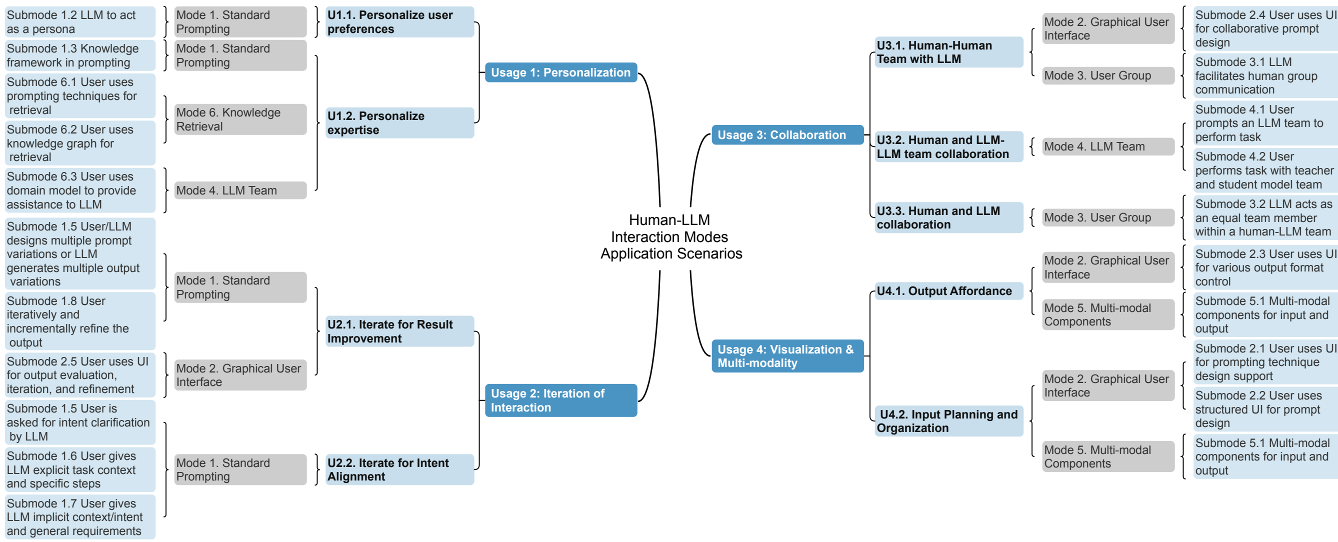

Figure 5: Human-LLM Interaction Modes

2) A group of LLMs, one user. This closely aligns with AI agent concepts, where multiple LLMs take on different roles and collaborate to accomplish tasks.

3) Different Human and LLM collaboration relationship. This has been a highly popular research area in recent years. This usage includes one particularly interesting submode 3.2, where humans and LLMs collaborate as equal team members. This shifts repositions LLMs from being mere tools, as seen in most submodes of Mode 1, to true collaborators in submode 3.2. Exploring different levels of LLM proactivity introduces intriguing variations in system design. For example, a proactive LLM could seek intent clarification from humans and actively contribute as an equal team member in problem-solving tasks (Submode 1.5).

7.4 For Visualization & Multi-modality

As shown in Usage 4 in Figure 5, both input and output can be designed with different variations to enhance visualization and accommodate multiple modalities, such as images and audio.

7.4.1 ***For improving input planning and organization.*** This usage focuses on improving input planning and organization through prompt design on UI and multi-modal components for input format conversion. The UI is designed to streamline the organization process, making it more intuitive for users. Meanwhile, multi-modal components enable the integration of diverse input formats, such as text and images, to establish a richer context for the LLM, enhancing its ability to generate more relevant and informed responses.

7.4.2 ***For improving output affordance.*** As shown in Usage 3.1 in Figure 5, this usage can also be achieved through UI and multi-modal components, but with a focus on output presentation rather than input planning. Here, the UI is primarily designed for displaying results, such as charts, graphs, or infographics, tailored to user needs. This visualization enhances clarity, reduces cognitive load, and improves accessibility for evaluation. Meanwhile, multi-modal components for output allow LLM-generated responses to be converted into images, audio, or a combination of formats, offering a richer presentation than the original text-based output.

8 CASE STUDY: ANALYSING CURRENT LLM SYSTEMS

We selected three widely used LLM applications—ChatGPT, Claude, and Cursor—for a case study, demonstrating the specific application of our taxonomy of interaction modes. ChatGPT and Claude represent the two most popular general-purpose conversational AIs, while Cursor exemplifies a commercially adopted LLM system widely used by developers. Table 2 presents the analysis results, including interaction modes associated with each system and their corresponding system features.

Our analysis reveals that even seemingly simple systems like ChatGPT and Claude rely on multiple interaction modes, each employing different variations that lead to distinct feature designs. Ultimately, these feature designs shape diverse interaction experiences, supporting visualization, personalization, and iteration.

9 DISCUSSION

Concept Connections with Software Design Patterns. When developing the taxonomy, we observed that, in conceptual level, some of our interaction modes have similarities to existing design patterns proposed by the Gang of Four [16, 47]—particularly at a highly abstract conceptual level. We have identified 8 potential sub-interaction modes that can be linked to these original 5 design patterns. We illustrate these connections in Figure 6.

For example, Submode 4.1, the multi-LLM team, can be compared to the composite design pattern—a structural design pattern that "lets you compose objects into tree structures and then work with these structures as if they were individual objects" [16]. Similarly, a multi-LLM team consists of multiple LLMs assigned to different subtasks within a hierarchical structure, collectively working toward a full task. This concept closely aligns with LLM agents [56]. Likewise, the mode where an LLM facilitates human group communication resembles the Mediator pattern, in which a mediator component manages interactions between different components [16, 49]. The key distinction is that these interaction modes are implemented in LLM-powered systems, whereas traditional design patterns were originally derived from conventional software architectures.

Table 2: Comparison of Interaction Modes and Mode-Specific Features for ChatGPT, Claude, and Cursor. We include only clearly identifiable and unambiguous modes, excluding those with varying interpretations. Additionally, a single feature may encompass multiple interaction modes; for example, using DALL·E to generate images involves both Mode 5 and Mode 7.

System	Involved Interaction Modes	Mode-Specific Features
ChatGPT	• **Mode 1: Standard Prompting** – Support most prompting techniques • **Mode 2: GUI** – Submode 2.1: Basic GUI – Submode 2.2: Structured UI – Submode 2.3: Output Format Control • **Mode 5: Multimodal Components** – Submode 5.1: Multimodal I/O • **Mode 6: Knowledge Base** – Submode 6.1: Prompts to perform RAG • **Mode 7: Tool Usage** – Submode 7.1: External Tools	• **Mode 1 (Prompting):** – Support most prompting techniques like reasoning (with o1 model), etc. • **Mode 2 (GUI):** – Basic structured UI fields for prompt composing (My GPTs) – Output formatting (e.g., markdown, code blocks, canvas) • **Mode 5 (Multimodal):** – Plugin support for images or audio (via DALL·E or Whisper plugins) • **Mode 6 (Knowledge Base):** – Uploaded files can be seen as external Knowledge Base • **Mode 7 (Tools):** – Web Searching – Code Interpreter and Canvas
Claude	• **Mode 1: Standard Prompting** – Support most prompting techniques • **Mode 2: GUI** – Submode 2.1: Basic GUI – Submode 2.2: Structured UI – Submode 2.3: Output Format Control – Submode 2.5: UI for output evaluation, iteration, and refinement	• **Mode 1 (Prompting):** – Support most prompting techniques like reasoning, etc. • **Mode 2 (GUI):** – Basic text interface – Allowing users to choose the response style of model – Rendering the code on Canvas, and allowing interactive sharing – Evaluating the results and asking for explanations
Cursor	• **Mode 1: Standard Prompting** – Support most prompting techniques • **Mode 2: GUI** – Submode 2.1: Basic GUI – Submode 2.3: Output Format Control – Submode 2.5: UI for output evaluation, iteration, and refinement • **Mode 5: Multimodal Components** – Submode 5.1: Multimodal I/O • **Mode 6: Knowledge Base** – Submode 6.1: Prompts to perform RAG • **Mode 7: Tool Usage** – Submode 7.1: External Tools	• **Mode 1 (Prompting):** – Best support for explicit context (Submode 1.6), especially when users can choose which pieces of code to include in their queries. • **Mode 2 (GUI):** – Basic code viewing and editing interface – Configurable UI to present intelligent suggestions and bulk modifications – Iterative code generation and refactoring • **Mode 5 (Multimodal):** – Supports images input • **Mode 6 (Knowledge Base):** – Comprehends the entire codebase and the current coding context as a implicit Knowledge Base • **Mode 7 (Tools):** – Supports integration of existing extensions from VS Code

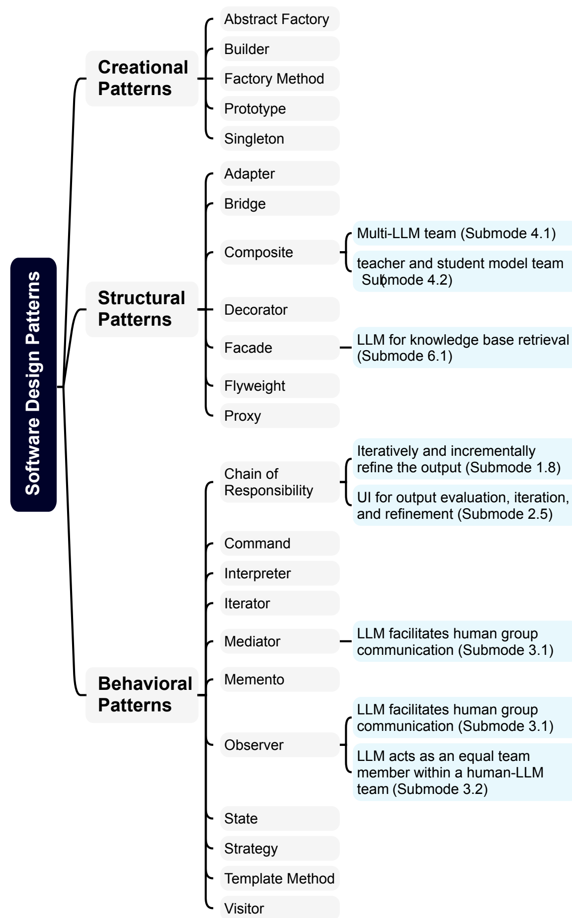

Figure 6: The interaction modes idea aligns with software design patterns, with several demonstrating particularly strong connections.

The development of LLM-powered applications is expanding rapidly, mirroring past technological shifts—such as the software design and development boom of the 1990s. During that period, 1.5 million software applications emerged in the U.S. within a decade, spanning domains such as science, commerce, information technology, and web applications [27]. Jones predicted that the "possible future for software engineering" may involve deriving design patterns from successful, already operational applications.

Similarly, as LLM-powered applications gain popularity, understanding the shift from traditional software design to human-LLM team-based systems presents a valuable potential opportunity to extend classical design pattern theories.

Evaluation factors to evaluate different combination of modes. In Section 6, we highlighted that most LLM systems are composed of different atomic interaction modes, with over four million possible combinations based on the currently identified modes. As more atomic interaction modes are discovered, this space could expand significantly. However, the objective is not merely to increase the number of combinations but to identify reusable patterns in LLM-powered system development.

Thus, an evaluation framework for assessing both individual modes and their combinations is crucial. Such a framework can help bridge the gap between theoretical interaction modes and their real-world implementation. For example, different modes and submodes could be compared based on factors such as resource allocation (e.g., human effort, system complexity, and time constraints) in system development. Some combinations may be cost-effective and require minimal human effort, while others could be resource-intensive without offering additional value. This evaluation would help developers make more informed design decisions.

10 LIMITATIONS AND FUTURE WORK

The taxonomy isn't static, it could be envolved through the progress of the LLM and relevant technology, which we intend to refine continuously. We foresee its expansion to encompass additional LLM interaction modes likely to emerge in the future. Moreover, it is important to note that many current classifications in our taxonomy are not absolute, given the slight overlap between some categories.

Looking forward, we believe that it will be especially valuable to extend the taxonomy to explicitly include different kinds of tasks and different design spaces. For instance, we believe that the capabilities and interaction modes needed to *creating* tasks will likely be systematically different from the capabilities and modes needed to *deciding* tasks. To do that, it could be good to perform the literature review to additional venues or build extra dimensions.

11 CONCLUSION

While conversational LLMs have become the "default" mode of human-LLM interaction, many other interaction methods exist. From a software engineering perspective, we conducted a systematic literature review across major AI and HCI venues and analyzed the system architectures of human-LLM teams in 267 papers. Through this analysis, we identified the fundamental building blocks: elements and their interfaces, behaviors, standard human-LLM interaction processes, and seven typical interaction modes. We discussed potential applications and provided a case study demonstrating the taxonomy's use. In the long run, we envision that this taxonomy could make a theoretical contribution to the SE for AI field.

12 DATA AVAILABILITY

The list of papers and their corresponding labels from our literature review are available on our website https://sites.google.com/view/taxonomymodes.

REFERENCES

[1] Ian Arawjo, Priyan Vaithilingam, Martin Wattenberg, and Elena Glassman. 2023. ChainForge: An Open-Source Visual Programming Environment for Prompt Engineering. In *Adjunct Proc. UIST '23*. 1–3. https://doi.org/10.1145/3586182.3616660

[2] Len Bass, Paul Clements, and Rick Kazman. 2013. *Software Architecture in Practice.* Pearson Education, Harlow.

[3] Stephen Brade, Bryan Wang, Mauricio Sousa, Sageev Oore, and Tovi Grossman. 2023. Promptify: Text-to-Image Generation through Interactive Prompt Exploration with Large Language Models. In *Proc. UIST '23*. 1–14. https://doi.org/10.1145/3586183.3606725

[4] Alice Cai, Steven R Rick, Jennifer L Heyman, Yanxia Zhang, Alexandre Filipowicz, Matthew Hong, Matt Klenk, and Thomas Malone. 2023. DesignAID: Using Generative AI and Semantic Diversity for Design Inspiration. In *Proc. ACM Collective Intelligence Conf. (CI '23)*. 1–11. https://doi.org/10.1145/3582269.3615596

[5] Zhenyao Cai, Seehee Park, Nia Nixon, and Shayan Doroudi. 2024. Advancing Knowledge Together: Integrating Large Language Model-Based Conversational AI in Small Group Collaborative Learning. In *Ext. Abstracts CHI Conf. Human Factors Comput. Syst. (CHI EA '24)*. Article 37, 9 pages. https://doi.org/10.1145/3613905.3650868

[6] Guangyao Chen, Siwei Dong, Yu Shu, Ge Zhang, Jaward Sesay, Börje F. Karlsson, Jie Fu, and Yemin Shi. 2024. AutoAgents: A Framework for Automatic Agent Generation. arXiv:2309.17288 [cs.AI] https://arxiv.org/abs/2309.17288

[7] Chun-Wei Chiang, Zhuoran Lu, Zhuoyan Li, and Ming Yin. 2024. Enhancing AI-Assisted Group Decision Making through LLM-Powered Devil's Advocate. In *Proc. 29th Int. Conf. Intelligent User Interfaces (IUI '24)*. 103–119. https://doi.org/10.1145/3640543.3645199

[8] DaEun Choi, Sumin Hong, Jeongeon Park, John Joon Young Chung, and Juho Kim. 2024. CreativeConnect: Supporting Reference Recombination for Graphic Design Ideation with Generative AI. In *Proc. CHI '24*. Article 1055, 25 pages. https://doi.org/10.1145/3613904.3642794

[9] Paul Clements, David Garlan, Reed Little, Robert Nord, and Judith Stafford. 2003. Documenting software architectures: views and beyond. In *25th International Conference on Software Engineering, 2003. Proceedings.* IEEE, 740–741.

[10] Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Jingyuan Ma, Rui Li, Heming Xia, Jingjing Xu, Zhiyong Wu, Tianyu Liu, Baobao Chang, Xu Sun, Lei Li, and Zhifang Sui. 2024. A Survey on In-Context Learning. arXiv:2301.00234 [cs.CL] https://arxiv.org/abs/2301.00234

[11] Huifang Du, Xuejing Feng, Jun Ma, Meng Wang, Shiyu Tao, Yijie Zhong, Yuan-Fang Li, and Haofen Wang. 2024. Towards Proactive Interactions for In-Vehicle Conversational Assistants Utilizing Large Language Models. In *Proc. IJCAI-24*. 7850–7858. https://doi.org/10.24963/ijcai.2024/869

[12] Weihong Du, Jia Liu, Zujie Wen, Dingnan Jin, Hongru Liang, and Wenqiang Lei. 2024. CARE: A Clue-Guided Assistant for CSRs to Read User Manuals. In *Proc. 62nd Annu. Meet. Assoc. Comput. Linguistics (ACL '24)*. 10795–10811. https://doi.org/10.18653/v1/2024.acl-long.581

[13] Peitong Duan, Jeremy Warner, Yang Li, and Bjoern Hartmann. 2024. Generating Automatic Feedback on UI Mockups with Large Language Models. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) *(CHI '24)*. Association for Computing Machinery, New York, NY, USA, Article 6, 20 pages. https://doi.org/10.1145/3613904.3642782

[14] Wen Duan, Naomi Yamashita, Yoshinari Shirai, and Susan R Fussell. 2021. Bridging Fluency Disparity Between Native and Nonnative Speakers in Multilingual Multiparty Collaboration Using a Clarification Agent. *Proc. ACM Hum.-Comput. Interact.* 5, CSCW2 (2021), 1–31.

[15] Li Feng, Ryan Yen, Yuzhe You, Mingming Fan, Jian Zhao, and Zhicong Lu. 2024. CoPrompt: Supporting Prompt Sharing and Referring in Collaborative Natural Language Programming. In *Proc. CHI '24*. Article 934, 21 pages. https://doi.org/10.1145/3613904.3642212

[16] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. 1995. *Design patterns: elements of reusable object-oriented software.* Addison-Wesley Longman Publishing Co., Inc., USA.

[17] Sundarakrishnan Ganesh and Robert Sahlqvist. 2024. Exploring Patterns in LLM Integration-A study on architectural considerations and design patterns in LLM dependent applications. (2024).

[18] Jie Gao, Yuchen Guo, Gionnieve Lim, Tianqin Zhang, Zheng Zhang, Toby Jia-Jun Li, and Simon Tangi Perrault. 2024. CollabCoder: A Lower-barrier, Rigorous Workflow for Inductive Collaborative Qualitative Analysis with Large Language Models. arXiv:2304.07366 [cs.HC]

[19] David Garlan. 2000. Software architecture: a roadmap. In *Proceedings of the Conference on The Future of Software Engineering* (Limerick, Ireland) *(ICSE '00)*. Association for Computing Machinery, New York, NY, USA, 91–101. https://doi.org/10.1145/336512.336537

[20] 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 *Proc. CHI '24*. Article 1015, 22 pages. https://doi.org/10.1145/3613904.3641891

[21] Kunal Handa, Margarett Clapper, Jessica Boyle, Rose Wang, Diyi Yang, David Yeager, and Dorottya Demszky. 2023. "Mistakes Help Us Grow": Facilitating and Evaluating Growth Mindset Supportive Language in Classrooms. In *Proc. EMNLP 2023*. 8877–8897. https://doi.org/10.18653/v1/2023.emnlp-main.549

[22] Mina Huh, Yi-Hao Peng, and Amy Pavel. 2023. GenAssist: Making Image Generation Accessible. In *Proc. UIST '23*. Article 38, 17 pages. https://doi.org/10.1145/3586183.3606735

[23] Takumi Ito, Naomi Yamashita, Tatsuki Kuribayashi, Masatoshi Hidaka, Jun Suzuki, Ge Gao, Jack Jamieson, and Kentaro Inui. 2023. Use of an AI-Powered Rewriting Support Software in Context with Other Tools: A Study of Non-Native English Speakers. In *Proc. UIST '23*. Article 45, 13 pages. https://doi.org/10.1145/3586183.3606810

[24] Ellen Jiang, Kristen Olson, Edwin Toh, Alejandra Molina, Aaron Donsbach, Michael Terry, and Carrie J Cai. 2022. PromptMaker: Prompt-Based Prototyping with Large Language Models. In *CHI Conf. Human Factors Comput. Syst. Extended Abstracts*. 1–8. https://doi.org/10.1145/3491101.3503564

[25] Ellen Jiang, Edwin Toh, Alejandra Molina, Aaron Donsbach, Carrie J Cai, and Michael Terry. 2021. GenLine and GenForm: Two Tools for Interacting with Generative Language Models in a Code Editor. In *Adjunct Proc. 34th Annu. ACM Symp. User Interface Softw. Technol. (UIST '21)*. 145–147. https://doi.org/10.1145/3474349.3480209

[26] 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 *Proc. CHI '22*. 1–19. https://doi.org/10.1145/3491102.3501870

[27] Capers Jones. 2013. *The technical and social history of software engineering*. Addison-Wesley.

[28] Taewan Kim, Donghoon Shin, Young-Ho Kim, and Hwajung Hong. 2024. DiaryMate: Understanding User Perceptions and Experience in Human-AI Collaboration for Personal Journaling. In *Proc. CHI '24*. Article 1046, 15 pages. https://doi.org/10.1145/3613904.3642693

[29] Tae Soo Kim, Yoonjoo Lee, Minsuk Chang, and Juho Kim. 2023. Cells, Generators, and Lenses: Design Framework for Object-Oriented Interaction with Large Language Models. In *Proc. UIST '23*. 1–18. https://doi.org/10.1145/3586183.3606833

[30] Harsh Kumar, Yiyi Wang, Jiakai Shi, Ilya Musabirov, Norman A. S. Farb, and Joseph Jay Williams. 2023. Exploring the Use of Large Language Models for Improving the Awareness of Mindfulness. In *Ext. Abstracts CHI '23*. 1–7. https://doi.org/10.1145/3544549.3585614

[31] Cassandra Lee and Jessica R Mindel. 2024. Closer and Closer Worlds: Using LLMs to Surface Personal Stories in World-Building Conversation Games. In *Companion Pub. ACM Designing Interactive Systems Conf. (DIS '24 Companion)*. 289–293. https://doi.org/10.1145/3656156.3665430

[32] Jiahao Nick Li, Yan Xu, Tovi Grossman, Stephanie Santosa, and Michelle Li. 2024. OmniActions: Predicting Digital Actions in Response to Real-World Multimodal Sensory Inputs with LLMs. In *Proc. CHI '24*. Article 8, 22 pages. https://doi.org/10.1145/3613904.3642068

[33] 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 *Proc. CHI '23*. 1–31. https://doi.org/10.1145/3544548.3580817

[34] Xingyu 'Bruce' Liu, Vladimir Kirilyuk, Xiuxiu Yuan, Peggy Chi, Alex Olwal, Xiang 'Anthony' Chen, and Ruofei Du. 2023. Experiencing Visual Captions: Augmented Communication with Real-Time Visuals Using Large Language Models. In *Adjunct Proc. UIST '23*. Article 85, 4 pages. https://doi.org/10.1145/3586182.3615978

[35] Atefeh Mahdavi Goloujeh, Anne Sullivan, and Brian Magerko. 2024. Is It AI or Is It Me? Understanding Users' Prompt Journey with Text-to-Image Generative AI Tools. In *Proc. CHI '24*. Article 183, 13 pages. https://doi.org/10.1145/3613904.3642861

[36] Amama Mahmood, Junxiang Wang, Bingsheng Yao, Dakuo Wang, and Chien-Ming Huang. 2025. User Interaction Patterns and Breakdowns in Conversing with LLM-Powered Voice Assistants. *Int. J. Hum.-Comput. Stud.* 195 (2025), 103406. https://doi.org/10.1016/j.ijhcs.2024.103406

[37] Thomas W Malone, Kevin Crowston, and George Arthur Herman. 2003. *Organizing business knowledge: the MIT process handbook*. MIT press.

[38] Piotr Mirowski, Kory W. Mathewson, Jaylen Pittman, and Richard Evans. 2023. Co-Writing Screenplays and Theatre Scripts with Language Models: Evaluation by Industry Professionals. In *Proc. CHI '23*. 1–34. https://doi.org/10.1145/3544548.3581225

[39] David Moher, Alessandro Liberati, Jennifer Tetzlaff, Douglas G Altman, and t PRISMA Group*. 2009. Preferred reporting items for systematic reviews and meta-analyses: the PRISMA statement. *Annals of internal medicine* 151, 4 (2009), 264–269.

[40] Gun Woo Park, Payod Panda, Lev Tankelevitch, and Sean Rintel. 2024. The CoExplorer Technology Probe: A Generative AI-Powered Adaptive Interface to Support Intentionality in Planning and Running Video Meetings. In *Proceedings of the 2024 ACM Designing Interactive Systems Conference*. 1638–1657.

[41] Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. 2023. Generative Agents: Interactive Simulacra of Human Behavior. In *Proc. UIST '23*. Article 2, 22 pages. https://doi.org/10.1145/3586183.3606763

[42] Pat Pataranutaporn, Valdemar Danry, Lancelot Blanchard, Lavanay Thakral, Naoki Ohsugi, Pattie Maes, and Misha Sra. 2023. Living Memories: AI-Generated Characters as Digital Mementos. In *Proc. IUI '23*. 889–901. https://doi.org/10.1145/3581641.3584065

[43] Xiaohan Peng, Janin Koch, and Wendy E. Mackay. 2024. DesignPrompt: Using Multimodal Interaction for Design Exploration with Generative AI. In *Proceedings of the 2024 ACM Designing Interactive Systems Conference* (Copenhagen, Denmark) *(DIS '24)*. Association for Computing Machinery, New York, NY, USA, 804–818. https://doi.org/10.1145/3643834.3661588

[44] Zhenhui Peng, Xingbo Wang, Qiushi Han, Junkai Zhu, Xiaojuan Ma, and Huamin Qu. 2023. Storyfier: Exploring Vocabulary Learning Support with Text Generation Models. In *Proc. UIST '23*. Article 46, 16 pages. https://doi.org/10.1145/3586183.3606786

[45] Renjie Pi, Jiahui Gao, Shizhe Diao, Rui Pan, Hanze Dong, Jipeng Zhang, Lewei Yao, Jianhua Han, Hang Xu, Lingpeng Kong, and Tong Zhang. 2023. DetGPT: Detect What You Need via Reasoning. arXiv:2305.14167 [cs.CV] https://arxiv.org/abs/2305.14167

[46] K Andrew R Richards and Michael A Hemphill. 2018. A practical guide to collaborative qualitative data analysis. *Journal of Teaching in Physical education* 37, 2 (2018), 225–231.

[47] Douglas C Schmidt, Michael Stal, Hans Rohnert, and Frank Buschmann. 2013. *Pattern-oriented software architecture, patterns for concurrent and networked objects*. John Wiley & Sons.

[48] Hanieh Shakeri, Carman Neustaedter, and Steve DiPaola. 2021. SAGA: Collaborative Storytelling with GPT-3. In *Companion Pub. CSCW '21*. 163–166. https://doi.org/10.1145/3462204.3481771

[49] Alan Shalloway and James R Trott. 2004. *Design patterns explained: a new perspective on object-oriented design*. Pearson education.

[50] Ashish Sharma, Kevin Rushton, Inna Wanyin Lin, David Wadden, Khendra G Lucas, Adam S Miner, Theresa Nguyen, and Tim Althoff. 2023. Cognitive reframing of negative thoughts through human-language model interaction. *arXiv preprint arXiv:2305.02466* (2023).

[51] Ben Shneiderman and Catherine Plaisant. 2004. *Designing the User Interface: Strategies for Effective Human-Computer Interaction (4th Edition)*. Pearson Addison Wesley.

[52] Mads Soegaard. 2015. Interaction Styles. Retrieved March 6, 2025 from https://www.interaction-design.org/literature/book/the-glossary-of-human-computer-interaction/interaction-styles.

[53] Hari Subramonyam, Roy Pea, Christopher Pondoc, Maneesh Agrawala, and Colleen Seifert. 2024. Bridging the Gulf of Envisioning: Cognitive Challenges in Prompt-Based Interactions with LLMs. In *Proc. CHI '24*. Article 1039, 19 pages. https://doi.org/10.1145/3613904.3642754

[54] Sangho Suh, Bryan Min, Srishti Palani, and Haijun Xia. 2023. Sensecape: Enabling Multilevel Exploration and Sensemaking with Large Language Models. In *Proc. UIST '23*. Article 1, 18 pages. https://doi.org/10.1145/3586183.3606756

[55] Yuqian Sun, Ying Xu, Chenhang Cheng, Yihua Li, Chang Hee Lee, and Ali Asadipour. 2023. Explore the Future Earth with Wander 2.0: AI Chatbot Driven by Knowledge-Base Story Generation and Text-to-Image Model. In *Ext. Abstracts CHI '23*. Article 450, 5 pages. https://doi.org/10.1145/3544549.3583931

[56] Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, Wayne Xin Zhao, Zhewei Wei, and Jirong Wen. 2024. A survey on large language model based autonomous agents. *Frontiers of Computer Science* 18, 6 (March 2024). https://doi.org/10.1007/s11704-024-40231-1

[57] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. arXiv:2201.11903 [cs.CL]

[58] Yuxi Wei, Zi Wang, Yifan Lu, Chenxin Xu, Changxing Liu, Hao Zhao, Siheng Chen, and Yanfeng Wang. 2024. Editable Scene Simulation for Autonomous Driving via Collaborative LLM-Agents. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 15077–15087.

[59] Jiaxin Wen, Ruiqi Zhong, Pei Ke, Zhihong Shao, Hongning Wang, and Minlie Huang. 2024. Learning Task Decomposition to Assist Humans in Competitive Programming. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Lun-Wei Ku, Andre Martins, and Vivek Srikumar (Eds.). Association for Computational Linguistics, Bangkok, Thailand, 11700–11723. https://doi.org/10.18653/v1/2024.acl-long.629

[60] 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]

[61] Guande Wu, Chen Zhao, Claudio Silva, and He He. 2024. Your Co-Workers Matter: Evaluating Collaborative Capabilities of Language Models in Blocks World. In *Findings of the Association for Computational Linguistics ACL 2024*, Lun-Wei Ku, Andre Martins, and Vivek Srikumar (Eds.). Association for Computational Linguistics, Bangkok, Thailand and virtual meeting, 4941–4957. https://doi.org/10.18653/v1/2024.findings-acl.294

[62] Jiageng Wu, Xian Wu, and Jie Yang. 2024. Guiding clinical reasoning with large language models via knowledge seeds. *arXiv preprint arXiv:2403.06609* (2024).

[63] Tongshuang Wu, Ellen Jiang, Aaron Donsbach, Jeff Gray, Alejandra Molina, Michael Terry, and Carrie J. Cai. 2022. PromptChainer: Chaining Large Language Model Prompts through Visual Programming. http://arxiv.org/abs/2203.06566 arXiv:2203.06566 [cs].

[64] Tongshuang Wu, Michael Terry, and Carrie Jun Cai. 2022. AI Chains: Transparent and Controllable Human-AI Interaction by Chaining Large Language Model Prompts. In *CHI Conference on Human Factors in Computing Systems*. ACM, New Orleans LA USA, 1–22. https://doi.org/10.1145/3491102.3517582

[65] Yiquan Wu, Siying Zhou, Yifei Liu, Weiming Lu, Xiaozhong Liu, Yating Zhang, Changlong Sun, Fei Wu, and Kun Kuang. 2023. Precedent-Enhanced Legal Judgment Prediction with LLM and Domain-Model Collaboration. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, Houda Bouamor, Juan Pino, and Kalika Bali (Eds.). Association for Computational Linguistics, Singapore, 12060–12075. https://doi.org/10.18653/v1/2023.emnlp-main.740

[66] Ziang Xiao, Xingdi Yuan, Q. Vera Liao, Rania Abdelghani, and Pierre-Yves Oudeyer. 2023. Supporting Qualitative Analysis with Large Language Models: Combining Codebook with GPT-3 for Deductive Coding. In *28th International Conference on Intelligent User Interfaces*. ACM, Sydney NSW Australia, 75–78. https://doi.org/10.1145/3581754.3584136

[67] Xinxin Yan and Ndapa Nakashole. 2021. A Grounded Well-being Conversational Agent with Multiple Interaction Modes: Preliminary Results. In *Proceedings of the 1st Workshop on NLP for Positive Impact*, Anjalie Field, Shrimai Prabhumoye, Maarten Sap, Zhijing Jin, Jieyu Zhao, and Chris Brockett (Eds.). Association for Computational Linguistics, Online, 143–151. https://doi.org/10.18653/v1/2021.nlp4posimpact-1.16

[68] Yao Yao, Zuchao Li, and Hai Zhao. 2024. GKT: A Novel Guidance-Based Knowledge Transfer Framework For Efficient Cloud-edge Collaboration LLM Deployment. In *Findings of the Association for Computational Linguistics ACL 2024*, Lun-Wei Ku, Andre Martins, and Vivek Srikumar (Eds.). Association for Computational Linguistics, Bangkok, Thailand and virtual meeting, 3433–3446. https://doi.org/10.18653/v1/2024.findings-acl.204

[69] Ori Yoran, Tomer Wolfson, Ben Bogin, Uri Katz, Daniel Deutch, and Jonathan Berant. 2023. Answering Questions by Meta-Reasoning over Multiple Chains of Thought. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, Houda Bouamor, Juan Pino, and Kalika Bali (Eds.). Association for Computational Linguistics, Singapore, 5942–5966. https://doi.org/10.18653/v1/2023.emnlp-main.364

[70] 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*. ACM, Hamburg Germany, 1–21. https://doi.org/10.1145/3544548.3581388

[71] Kaiyan Zhang, Jianyu Wang, Ermo Hua, Biqing Qi, Ning Ding, and Bowen Zhou. 2024. CoGenesis: A Framework Collaborating Large and Small Language Models for Secure Context-Aware Instruction Following. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Lun-Wei Ku, Andre Martins, and Vivek Srikumar (Eds.). Association for Computational Linguistics, Bangkok, Thailand, 4295–4312. https://doi.org/10.18653/v1/2024.acl-long.235

[72] Shuo Zhang, Liangming Pan, Junzhou Zhao, and William Yang Wang. 2024. The Knowledge Alignment Problem: Bridging Human and External Knowledge for Large Language Models. In *Findings of ACL 2024*. 2025–2038. https://doi.org/10.18653/v1/2024.findings-acl.121

[73] Zheng Zhang, Jie Gao, Ranjodh Singh Dhaliwal, and Toby Jia-Jun Li. 2023. VISAR: A Human-AI Argumentative Writing Assistant with Visual Programming and Rapid Draft Prototyping. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology (UIST '23)*. Association for Computing Machinery, New York, NY, USA, 1–30. https://doi.org/10.1145/3586183.3606800

[74] Zheng Zhang, Jie Gao, Ranjodh Singh Dhaliwal, and Toby Jia-Jun Li. 2023. VISAR: A Human-AI Argumentative Writing Assistant with Visual Programming and Rapid Draft Prototyping. *arXiv preprint arXiv:2304.07810* (2023).

[75] Chengbo Zheng, Yuheng Wu, Chuhan Shi, Shuai Ma, Jiehui Luo, and Xiaojuan Ma. 2023. Competent but Rigid: Identifying the Gap in Empowering AI to Participate Equally in Group Decision-Making. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*. ACM, Hamburg Germany, 1–19. https://doi.org/10.1145/3544548.3581131

[76] Li Zhong, Zilong Wang, and Jingbo Shang. 2024. Debug like a Human: A Large Language Model Debugger via Verifying Runtime Execution Step by Step. In *Findings of the Association for Computational Linguistics: ACL 2024*, Lun-Wei Ku, Andre Martins, and Vivek Srikumar (Eds.). Association for Computational Linguistics, Bangkok, Thailand, 851–870. https://doi.org/10.18653/v1/2024.findings-acl.49