Convo: A Visual Analytics Framework for Intent Structure and Optimization in Multi-Turn Prompting

Quynh Ho
Stony Brook University
Stony Brook, New York, USA
quynh.ho@stonybrook.edu

Abstract

As large language models become embedded in everyday programming practice, developers increasingly rely on multi-turn prompting to generate, debug, and refine code. However, existing chat-based interfaces provide limited support for understanding how prompts evolve across an interaction, making it difficult to manage redundancy, track intent, and consolidate effective instructions. I present Convo, a visual analytics system that externalizes multi-turn prompts as persistent, interactive artifacts. Convo enables programmers to spatially organize prompts, inspect relationships among them, and consolidate conversational histories into reusable prompt specifications, supporting reflective and user-controlled prompt refinement rather than automated optimization. A formative qualitative evaluation with programmers shows that Convo helps reduce cognitive overhead when working with long prompt histories, improves visibility of prompt structure, and supports deliberate prompt consolidation. These results suggest that visual representations can play a critical role in supporting prompt-based programming workflows. To facilitate future direction, I open-source my code at https://github.com/quynhho1601/convo_app

CCS Concepts

• **Human-centered computing** → **Visualization**; **User interface design**; • **Software and its engineering** → *Programming environments.*

Keywords

Prompt Engineering, Visual Analytics, Human–AI Interaction, Programming Tools, Large Language Models

1 Introduction and Motivation

Large Language Models (LLMs) have introduced a new interaction paradigm for programming, where developers increasingly rely on natural language prompts to generate, modify, and reason about code. In this paradigm, prompting is not a one-shot action but an iterative process: programmers refine instructions over multiple conversational turns, experiment with alternative formulations, and incrementally shape model behavior. As a result, prompts themselves function as a critical programming artifact, directly influencing system outcomes.

Recent research highlights that prompting has become deeply embedded in real-world software development workflows. Developers routinely revise prompts alongside code, adjusting instructions, constraints, and formatting as features evolve. However, unlike traditional programming artifacts, prompts lack established structure, visibility, and tooling support. They are often scattered across chat histories or embedded as raw strings, making it difficult for programmers to reason about how effective prompts emerge through iteration or how multiple prompts relate to one another.

At the same time, most interaction interfaces for LLMs present prompting as a linear dialogue. This representation obscures the structure of multi-turn prompting and places a heavy cognitive burden on programmers, who must mentally track which prompts express core intent and which serve as supporting or exploratory queries. As conversations grow longer, programmers often resort to ad hoc strategies—such as manual copying, rewriting, or restarting conversations—to consolidate their intent, indicating a lack of support for reflective prompt refinement.

From an HCI perspective, this reveals a critical gap. While prior systems have focused on enabling complex prompt execution or studying prompt evolution at scale, there is limited support for helping individual programmers understand and organize their own prompts during everyday coding tasks. Without external representations that make prompt structure visible, programmers are left to manage increasingly complex prompt workflows internally, limiting reuse, clarity, and control.

Convo is motivated by this gap. Rather than treating prompts as transient text or executable components, Convo treats them as objects for sensemaking. By visually externalizing multi-turn prompts and allowing programmers to spatially organize and consolidate them, Convo supports reflective reasoning about prompting as a programming activity. This work explores how visual analytics can reduce cognitive load and help programmers regain control over iterative prompting workflows.

2 Background and Related Work

PromptChainer [2] introduces a visual programming interface for chaining LLM prompts into executable pipelines, enabling users to decompose complex tasks into sequential steps and debug cascading errors. User studies demonstrate that visual representations improve authoring and debugging of multi-step LLM applications. However, PromptChainer focuses on executable workflow construction for prototyping, whereas Convo supports exploratory, conversational prompting during coding without execution orchestration. Prompting in the Wild [3] provides the first large-scale empirical study of prompt evolution in software repositories. Analyzing over a thousand prompt changes across GitHub projects, the study reveals that prompts are frequently modified during development, often without documentation, leading to prompt–code misalignment and logical inconsistencies. This work establishes prompts as evolving software artifacts requiring systematic maintenance but examines prompting at the repository level rather than during live interaction. Convo complements this by focusing on

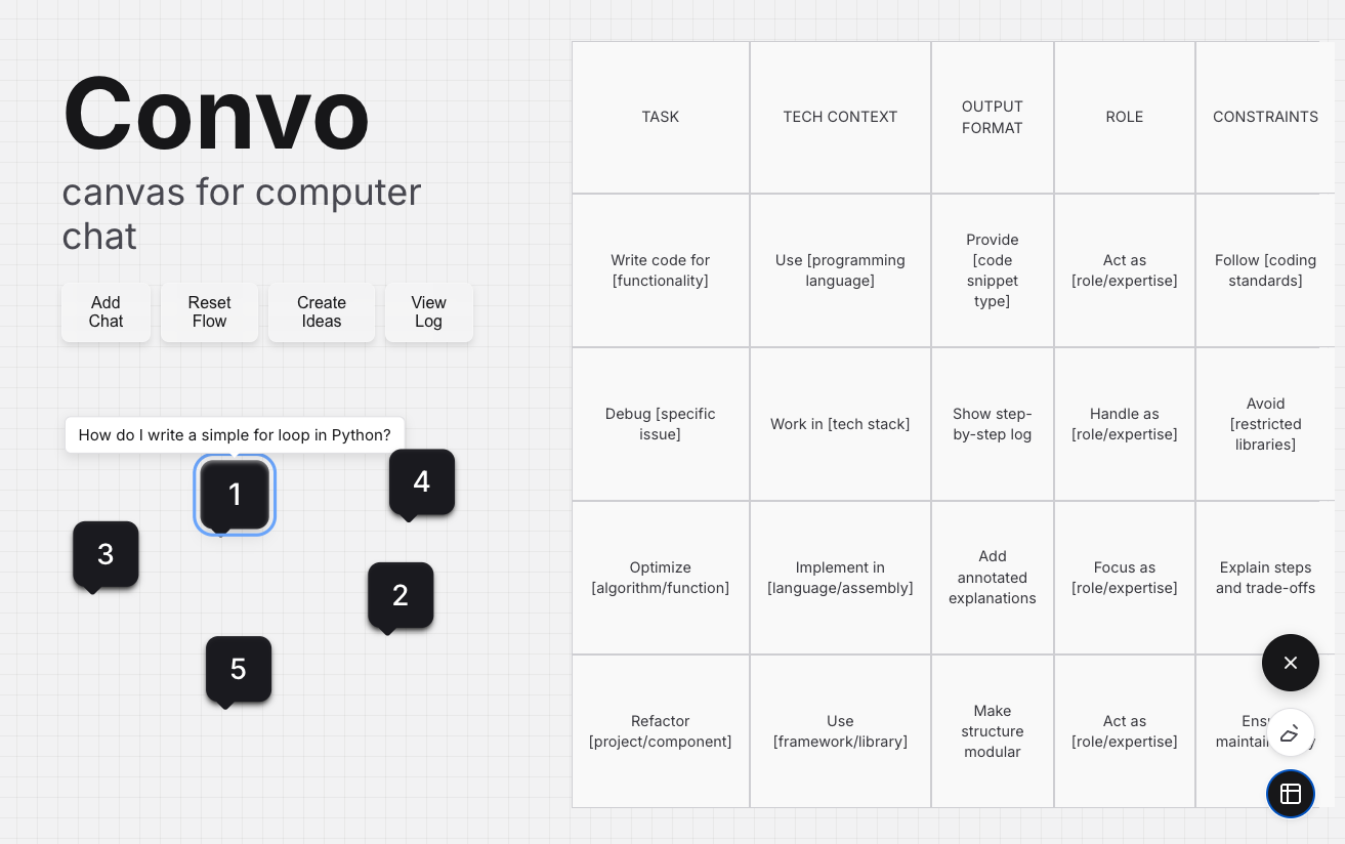

Figure 1: Convo User Interface Overview.

interaction-time sensemaking as programmers iteratively develop prompts.

Together, these works demonstrate that prompting is structurally complex and central to programming workflows. PromptChainer addresses complexity through executable visual pipelines; Prompting in the Wild exposes long-term maintenance challenges. Convo occupies a distinct position: it supports individual programmers in visually organizing and consolidating multi-turn prompts during coding, without imposing execution semantics or repository-scale analysis.

3 Technical Approach

3.1 Prescribing a Prompt Schema for Canvas Design

Prior work in prompt engineering has shown that explicitly structured prompts can substantially improve the reliability, reusability, and clarity of code generation. In contrast to ad-hoc natural language instructions, several systems propose formal prompt schemas that decompose instructions into functional components, analogous to structured programs. These findings motivate the use of an explicit prompt schema as the foundation for Convo's canvas-based design.

Recent work on prompt frameworks for code generation demonstrates that separating prompt content into distinct instructional sections improves model performance while reducing ambiguity. Cruz et al. [4] propose a structured prompting framework that decomposes coding instructions into multiple functional stages, such as task analysis, design decisions, implementation constraints, and quality requirements [4]. Their ADIHQ framework shows that when prompts explicitly separate core task intent from constraints and quality rules, models generate more reliable and test-passing code. Importantly, this framework prescribes a clear internal organization of prompts rather than relying on linear, monolithic instructions.

Similarly, LangGPT introduces a formal, programming-language-inspired prompt schema that organizes prompts into reusable modules such as *Goal*, *Profile*, *Constraint*, *Workflow*, and *Output Format* [1]. LangGPT explicitly argues that prompts benefit from regularized structure, extensibility, and separation of concerns, mirroring how programming languages enforce modularity and clarity. The authors show that such modular prompt design improves both prompt reuse and iterative refinement, especially in complex tasks such as code generation.

Convo's canvas design is directly informed by these structured prompt frameworks. Rather than treating prompts as undifferentiated text blocks, Convo externalizes prompt components as spatially

distinct elements on a two-dimensional canvas. The five canvas columns—*Task*, *Technical Context*, *Role*, *Output Format*, and *Constraints*—are abstractions derived from recurring components prescribed in prior prompt schemas. While Convo does not enforce a rigid template, these columns reflect the functional decomposition advocated by Cruz et al. and LangGPT, making implicit prompt structure explicit without requiring users to learn a formal language [1, 4].

Crucially, this design choice aligns with how developers already refine prompts in practice. Both cited works emphasize that prompt quality emerges through iterative addition of missing components rather than complete rewrites. By mapping prompts onto structural categories, the canvas supports developers in identifying which components are underspecified or missing, reducing redundant trial-and-error interactions. Unlike LangGPT, which prescribes a textual schema, Convo translates prompt structure into a visual interaction model, enabling users to reason about prompt composition spatially.

In summary, prior work demonstrates that structured prompt schemas improve code generation reliability and reusability [1, 4]. Convo builds on these insights by embedding a lightweight, visually guided prompt schema into its canvas design, preserving flexibility while supporting reflective prompt construction and consolidation.

3.2 Design Requirements

The design of Convo is informed by observed practices in prompt-based programming and by gaps identified in prior work on prompt authoring and evolution (Figure 1). Based on these observations, we derive the following design requirements.

Prompt structure visibility Programmers need support for making the structure of multi-turn prompting visible. Since prompts are typically authored through linear chat interfaces, relationships between prompts—such as which ones express core intent versus exploratory or supporting queries—remain implicit. Convo must externalize prompts as first-class objects that can be inspected and compared.

Prompt refinement support Prompt-based programming is iterative and exploratory. Rather than optimizing for execution or automation, the system should support reflection: allowing users to revisit prior prompts, identify redundancy, and synthesize effective prompts from conversational history.

Workflow compatibility For a prompt-support tool to be useful in practice, it must integrate naturally into existing workflows. Convo is designed to operate on prompts already produced through standard LLM interactions, such as chat-based interfaces, without requiring programmers to learn new prompt languages, adopt formal schemas, or restructure their interaction style. By allowing users to import and analyze past conversations, Convo complements existing tools rather than replacing them, minimizing disruption to established prompting practices.

Prompt consolidation support Developers frequently consolidate long prompt conversations into shorter, reusable prompts. Convo should explicitly support this consolidation process, enabling users to transform multiple prompts into a concise optimized prompt while preserving intent.

3.3 System Overview

Convo is a web app that ingests multi-turn prompt conversations and provides an interactive visual workspace for organizing, analyzing, and consolidating prompts. Rather than executing prompts or generating code directly, the system operates as a reflective layer that augments existing LLM interfaces, supporting programmers' sensemaking during iterative prompt development.

The system architecture comprises three interconnected components. The **Prompt Ingestion Module** parses and stores prompt text from multi-turn interactions, preserving conversational structure while extracting individual prompts as discrete artifacts. The **Visual Workspace** renders these prompts as interactive nodes within a two-dimensional canvas, enabling spatial organization and direct manipulation. The **Prompt Consolidation Module** assists users in synthesizing optimized prompts from selected nodes through LLM-mediated rewriting, presenting consolidated results as new artifacts for further refinement.

Critically, Convo is designed to be model-agnostic, operating independently of the specific LLM used to generate original responses. This design choice ensures broad applicability across diverse prompting workflows and LLM providers, positioning the system as a domain-general tool for prompt sensemaking rather than a platform-specific extension.

3.4 Prompt Ingestion Module

The Prompt Ingestion module enables Convo to operate on real multi-turn interactions authored on existing LLM platforms. Users begin by exporting their chat history from the LLM interface they use in practice, such as ChatGPT, Claude, or Gemini. Major platforms already support this workflow, though through different mechanisms—for example, Claude allows users to export chat history directly from its settings and privacy interface, while Gemini provides chat export through Google Takeout. Once exported, the chat transcript can be pasted or uploaded into Convo via an **Add Chat** button (Figure 2, Step 1). The ingestion module parses the transcript to preserve the original conversational structure while extracting individual user-authored prompts as discrete artifacts. Each prompt is assigned a turn index corresponding to its position in the original dialogue, enabling Convo to retain temporal context without enforcing linear navigation. By separating prompt extraction from model execution, the ingestion module allows programmers to revisit and analyze authentic prompting behavior from past coding sessions, grounding Convo's visual workspace in real interaction data rather than synthetic or system-generated prompts.

3.5 Visual Workspace

3.5.1 Prompt Nodes as First-Class Objects. In Convo, each prompt is represented as a node within a two-dimensional workspace, where nodes function as persistent visual artifacts rather than transient chat messages (Figure 2, Step 2). Each node contains the raw prompt text along with minimal metadata, including a numeric identifier corresponding to the turn order in the original dialogue. This turn-based numbering allows programmers to situate prompts within the temporal context of the conversation without enforcing a linear reading order. To further support recall of past interactions,

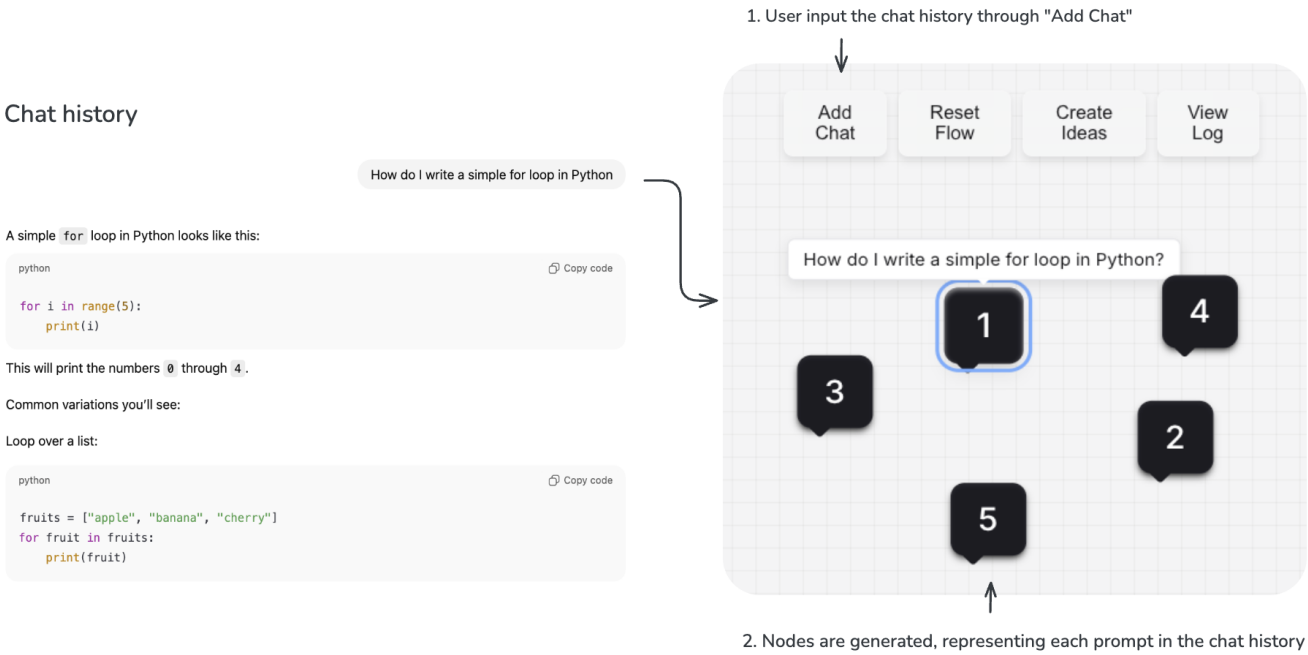

Figure 2: Prompt Ingestion to Visualize Each Prompt as a Node.

users can hover over a node to reveal the full content of the prompt as it appeared in the original dialogue, enabling lightweight inspection without disrupting the spatial organization of the workspace. By decoupling prompt inspection from linear chat history, Convo reduces the need for users to mentally reconstruct prior interactions

Convo employs direct manipulation as its primary interaction paradigm, enabling users to engage with prompt nodes through immediate visual feedback rather than indirect menu-based controls. The system supports three core interactions: spatial reorganization through drag-and-drop manipulation, multi-node selection to express relational groupings, and inline text editing for iterative refinement. These interactions are deliberately designed to be lightweight and reversible, reducing interaction cost and encouraging exploratory experimentation without committing users to irreversible changes.

Nodes are visually uniform to avoid implying hierarchy or correctness. Instead, meaning emerges through spatial arrangement and user interaction. This aligns with prior HCI work showing that spatial organization supports external cognition and reduces memory load.

3.5.2 Spatial Organization. Users freely position prompt nodes within the workspace through direct drag-and-drop interaction, using spatial proximity and layout to communicate semantic relationships such as similarity, refinement, or shared task relevance. Convo relies on user-driven spatial organization, reflecting the inherently exploratory and subjective nature of prompt development in programming tasks. This design choice recognizes that different programmers may conceptualize relationships between prompts in distinct ways, and that imposing rigid structural constraints could prematurely limit interpretive flexibility or constrain problem-solving strategies.

Across sessions, users employ spatial arrangement as an active reasoning strategy rather than a purely organizational action. This allows emergent organizational strategies, including clustering exploratory or incomplete prompts around a small number of central "core" prompts, spatially separating abandoned or superseded iterations from active work areas, and arranging nodes along loose temporal or conceptual progressions to reflect the evolution of their thinking. Users can frequently reposition nodes as their understanding of the task evolved, indicating that spatial layout served as a dynamic external representation of their mental model. These behaviors can be seen as organic interaction rather than instruction, enabling users to iteratively externalize, inspect, and refine their understanding of conversational structure over time.

3.5.3 Prompt Consolidation Workflow. Convo supports prompt consolidation through a four-stage workflow that guides users from exploratory prompt analysis to reusable prompt artifacts. Each stage corresponds to a distinct interaction mode in the system.

Classification Mode. The workflow begins in Classification Mode, which is particularly useful for long conversations containing many user prompts (Figure 3, Step 1). By clicking the classification button, users receive visual suggestions (nodes with pin) that help reduce the effort of manually selecting relevant prompts. Convo visually distinguishes prompts that introduce new information from those that are largely redundant, allowing users to quickly identify which prompts contribute meaningful content and which repeat

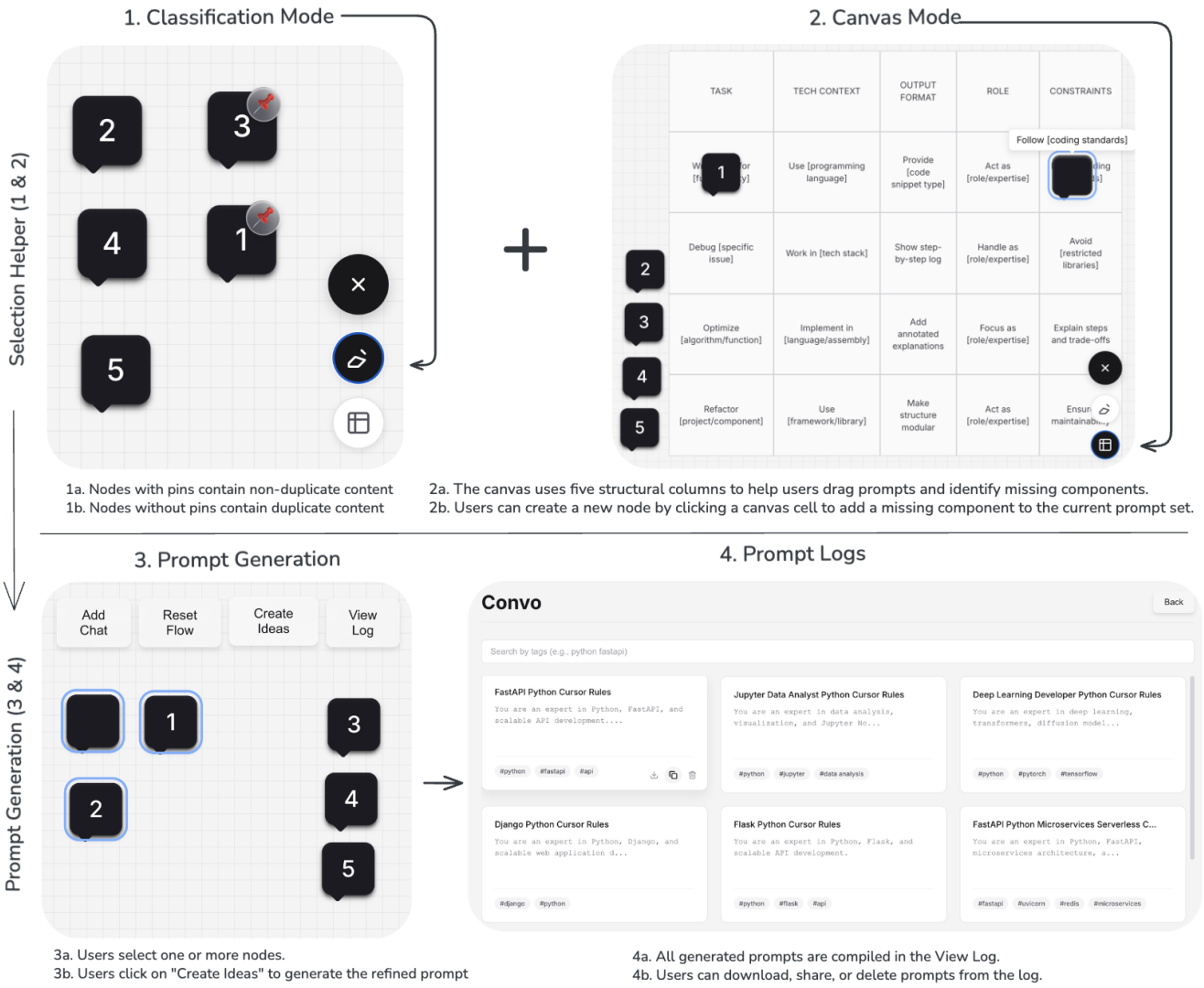

Figure 3: Convo Visual Workspace and Workflows

prior intent. This step helps users efficiently narrow down the set of prompts to carry forward into consolidation.

Canvas Mode. Selected nodes can be organized in Canvas Mode, which externalizes a structured prompt schema derived from prior work on formal prompt design for code generation (Figure 3, Step 2). The canvas is arranged into five columns that correspond to commonly prescribed prompt components, including task specification, technical context, role assumptions, output format, and constraints. Users drag prompt nodes into these columns to make prompt structure explicit and to assess whether required components have been sufficiently specified. Under each column, example cells illustrate the corresponding category and serve as interaction targets; users can click directly on a cell to create a new node within that category, enabling missing information to be added explicitly rather than introduced incrementally through trial-and-error prompting.

Prompt Generation. Once users are satisfied with the selected and structured nodes, they proceed to Prompt Generation. In this stage, users select one or more nodes and trigger the Create Ideas action (Figure 3, Step 3) Convo generates a refined prompt by consolidating the selected content into a concise instruction. The generated prompt is shown in a textbox, allowing users to inspect, edit, save and add tags for further refinement while preserving the original prompts for comparison.

Prompt Logs Finally, all refined prompts produced during consolidation are stored in the Prompt Log (Figure 3, Step 4). The log serves as a persistent collection of reusable prompts compiled across sessions. From this view, users can download prompts for reuse, share them with collaborators, or delete prompts that are no longer relevant. By separating generation from long-term storage,

Convo supports both iterative experimentation and durable prompt reuse.

Together, these stages form a structured yet flexible consolidation workflow that transforms fragmented multi-turn prompting into reusable, inspectable prompt artifacts without discarding the context from which they emerged.

3.6 Prototype Architecture

Convo is implemented as a web-based system with a React and TypeScript frontend that supports interactive node manipulation and canvas-based layouts. The backend is built using Flask to handle chat ingestion, prompt consolidation, and prompt log management. Prompt rewriting leverages the Gemini Flash 2.0 API but is model-agnostic by design, allowing alternative language models to be substituted without changes to the interaction logic. Prompts are stored as simple text entries without tracking execution behavior or dependencies, reflecting Convo's focus on reflection rather than execution. This lightweight architecture enables rapid interaction and iteration while avoiding assumptions about prompt semantics or downstream use.

4 Validation

Evaluating interactive systems that support exploratory and reflective workflows presents inherent challenges. The primary contribution of Convo lies in supporting prompt sensemaking and consolidation rather than optimizing measurable performance outcomes. As such, traditional quantitative benchmarks or accuracy-based metrics are not well suited to assessing its value. Prior HCI research emphasizes that for novel interaction paradigms and early-stage system prototypes, formative qualitative evaluation and analytic validation are appropriate and informative [5] . Following this guidance, we conducted a small-scale, qualitative evaluation to assess whether Convo supports its intended interaction goals.

4.1 Evaluation Setup

I conducted a formative evaluation with three graduate students from Stony Brook University, two in Computer Science (Person 1, Person 2) and one in Data Science (Person 3) who regularly use large language models for programming-related tasks such as code generation, debugging, and refactoring. All participants had prior experience interacting with LLMs through chat-based interfaces and reported frequent use of prompting as part of their coursework or personal projects. None of the participants were involved in the design or implementation of Convo. Given their familiarity with prompt-based workflows and programming tasks, they serve as expert proxy users for evaluating early-stage interaction techniques.

Participants were recruited through personal outreach and evaluated Convo in individual sessions. Prior to the session, participants were asked to identify a recent coding task in which they had relied on multi-turn prompting, or were provided with a representative coding-related chat transcript. At the beginning of each session, participants were given a brief walkthrough of Convo's core features, including node representation, Classification Mode, Canvas Mode, and prompt consolidation. The evaluation then focused on hands-on interaction rather than prompt authoring from scratch.

Each session was structured as a guided walkthrough consisting of three tasks: (1) identifying relevant prompts from a long conversational history, (2) organizing prompts within the canvas to assess coverage and redundancy, and (3) consolidating selected prompts into a reusable instruction. Participants interacted freely with the system while describing their thought process aloud. I observed how participants selected prompts, used hover-based inspection, organized nodes spatially, and refined consolidated prompts. Sessions emphasized iterative use of the interface, including revisiting earlier decisions and modifying prompt organization when deficiencies were identified.

Observations focused on task completion, feature usage, and qualitative feedback rather than performance metrics. Notes were taken on how participants navigated the interface, which features were used to support decision-making, and how participants described the usefulness of visual representations compared to linear chat interfaces. Each session lasted approximately 45 minutes. This evaluation approach aligns with formative, qualitative methods commonly used in early-stage HCI system research to assess interaction design and usability prior to large-scale deployment.

4.2 Experiment Observations

All participants successfully completed the walkthrough tasks and produced consolidated prompts they indicated were suitable for reuse. Across sessions, participants followed similar high-level interaction patterns when working with long prompt histories.

The chat import phase accounted for the longest uninterrupted interaction period prior to canvas use, lasting approximately 8–12 minutes for participants who had not previously exported chat histories from LLM platforms (Person 1, Person 3). These participants spent time locating export settings, selecting relevant transcripts, and preparing content for import. In contrast, the participant (Person 2) with prior experience exporting chat data completed this phase in approximately 3–5 minutes and transitioned more quickly into prompt organization. After prompts were imported, subsequent interaction phases proceeded with fewer pauses and required minimal clarification.

Participants next reduced the prompt set before engaging in detailed organization, typically by activating Classification Mode early in the session. After this step, participants focused their attention on a smaller subset of prompts and did not return to excluded nodes.

Participants consistently relied on hover-based inspection to verify prompt content. Rather than navigating back to the original chat transcript, participants hovered over nodes repeatedly while preserving the spatial layout they had constructed. Turn numbers were used to reason about prompt order and progression, particularly when participants compared earlier prompts with later refinements. Participants referenced temporal position verbally (e.g., "this came later") without engaging in linear scrolling.

In Canvas Mode, participants exhibited iterative spatial reorganization behaviors. Nodes were frequently moved across columns as participants reconsidered relevance or scope. Participants grouped prompts based on perceived contribution to the final instruction rather than original order. When participants identified missing information, they created new nodes directly within the canvas instead of editing existing prompts. These newly created nodes were

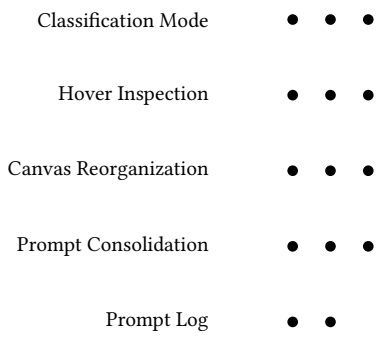

Figure 4: Feature usage frequency across three participants. Each dot represents observed use by one participant during the walkthrough. Frequencies indicate presence of use rather than interaction counts.

typically positioned near related prompts and were later included in consolidation.

During Prompt Consolidation, participants treated generated prompts as intermediate artifacts. All participants edited consolidated prompts after generation, often making multiple revisions before saving. In several sessions, participants generated more than one consolidated prompt from the same node set and compared versions using the Prompt Log before selecting one to retain.

As shown in Figure 4, all participants used Classification Mode, hover-based inspection, canvas reorganization, and prompt consolidation during the walkthrough. Use of the Prompt Log was observed for most, but not all, participants. These patterns suggest that participants converged on a common workflow centered around prompt filtering and spatial organization.

4.3 Qualitative Feedback

Participants consistently reported that Convo made relationships between prompts more visible compared to linear chat interfaces. Two participants noted that when working directly in chat-based systems, they typically rely on rereading large portions of the conversation or manually copying text to reconstruct relevant context. In contrast, Convo allowed participants to reason selectively about prompts by scanning, grouping, and inspecting nodes, reducing the need to revisit the original dialogue in full. Participants described this shift as making it easier to focus on higher-level organization rather than managing conversational detail.

Classification Mode was all cited as helpful for quickly narrowing long prompt histories. All participants claimed being able to reduce the set of prompts requiring attention before engaging in more detailed organization, describing the classification step as a useful "starting point" that prevented them from feeling overwhelmed by large conversations. One participant described the classification step as a "useful starting point" that allowed them to quickly prioritize prompts without feeling overwhelmed by the sheer volume of interactions. This early reduction in the number of prompts to be analyzed was essential for making the subsequent tasks—such as organizing and consolidating prompts—more efficient.

Once the prompt set was narrowed, participants consistently reported that the canvas provided a more structured space for deliberate reasoning about prompt coverage. The visual layout encouraged them to think critically about the content and structure of each prompt, particularly when identifying missing components or areas of overlap. Participants noted that they could easily spot gaps in the task description or constraints by comparing the prompts side by side. This process of organizing prompts into distinct categories within the canvas helped participants identify key components that were either underrepresented or overly generalized. By making this reasoning process explicit, the canvas allowed participants to refine their prompts more effectively, ensuring that all relevant aspects were captured without redundancy.

Participants emphasized that Convo fit naturally into their existing workflows. Because the system operated on exported chat histories, it did not require changes to how prompts were originally authored or how LLMs were used. Two participants noted that they viewed Convo as a complementary tool used after initial prompting, rather than a replacement for chat-based interaction. This separation was seen as beneficial, allowing them to reflect on prompting decisions without disrupting ongoing work.

Importantly, participants described Convo as supporting decision-making rather than automating it. They valued retaining control over which prompts to include, how to organize them, and how to refine consolidated outputs. System-generated prompts were consistently treated as starting points rather than final answers, and participants expressed confidence in making manual adjustments. This emphasis on user control was repeatedly highlighted as preferable to fully automated prompt optimization, particularly for programming tasks where precision and intent are critical.

5 Discussion and Future Work

5.1 Discussion and Limitations

This project explores how visual representations can support programmers' sensemaking in multi-turn prompting workflows, drawing on core HCI principles of external cognition and cognitive load reduction [6]. A key lesson is that prompt-based programming imposes substantial cognitive demands that are poorly supported by linear chat interfaces. Participants relied heavily on spatial organization, selective inspection, and persistent visual representations to reason about prompts, indicating that externalizing information into the environment helped offload memory and attention.

Rather than optimizing prompt generation automatically, Convo emphasizes user-controlled organization and refinement, reflecting the HCI principle of maintaining user agency in complex tasks. Participants' interaction patterns suggest that direct manipulation—such as grouping, repositioning, and selectively inspecting prompts—supported exploratory problem solving in ways consistent with how programmers approach coding tasks.

A second lesson concerns the emergence of structure through interaction. Observations showed that participants reasoned about prompts in terms of missing components or overlapping information, even though such structures are not explicitly represented in standard LLM interfaces. By externalizing prompt components through a canvas, Convo made implicit structure visible, supporting reflection-in-action and deliberate refinement.

Finally, Convo demonstrates the value of treating prompts as persistent artifacts rather than ephemeral chat messages. By enabling users to revisit, compare, and consolidate prompts, the system supports reflective practice and iterative exploration. Participants frequently generated multiple consolidated prompts and compared alternatives before selecting one, underscoring the importance of supporting exploration rather than premature convergence.

At the same time, this work has several limitations. The evaluation involved a small number of participants and focused on short-term interaction, which limits the ability to draw general conclusions about long-term use or broader adoption. Convo currently relies on manual import of chat histories, and this step introduced noticeable friction for some participants, potentially constraining usability without closer integration with existing LLM platforms. Finally, observations of how users engaged with the canvas interface are based on a limited number of data points, and the generalizability of these interaction patterns to other users or tasks remains an open question.

5.2 Future Work

Several directions for future work follow naturally from this study. First, tighter integration with existing LLM interfaces could reduce the overhead of importing chat histories and enable more seamless transitions between prompting and reflection. Second, longitudinal studies could examine how programmers use Convo over time, particularly whether structured visual reflection leads to improved prompt reuse or reduced iteration.

Future work could also explore adaptive or optional prompt schemas that respond to different programming tasks without enforcing a fixed structure. Finally, evaluating Convo with a broader range of users and tasks would help clarify how visual prompt sensemaking scales across domains and experience levels. While these directions are promising, the core contribution of this work lies in demonstrating that visual analytics can meaningfully support prompt-based programming workflows.

6 Conclusion

This work introduced Convo, a visual analytics system designed to support programmers' sensemaking and refinement of multi-turn prompting workflows. By externalizing prompts as persistent visual artifacts and enabling spatial organization, selective inspection, and structured consolidation, Convo addresses limitations of linear chat interfaces in supporting reflective prompt-based programming. A formative evaluation with graduate-level programmers demonstrated that visual representations help users reduce cognitive load, identify redundancy and missing components, and consolidate prompts in a controlled and deliberate manner.

The key takeaway of this work is that effective support for prompt-based programming does not require full automation or rigid schemas. Instead, lightweight visual structure and user-controlled interaction can meaningfully support reflection, exploration, and reuse. These findings suggest that treating prompts as inspectable artifacts—rather than ephemeral chat messages—opens new opportunities for designing human-centered tools for programming with large language models.

References

[1] Ming Wang, Yuanzhong Liu, Xiaoyu Liang, Songlian Li, Yijie Huang, Xiaoming Zhang, Sijia Shen, Chaofeng Guan, Daling Wang, Shi Feng, Huaiwen Zhang, Yifei Zhang, Minghui Zheng, and Chi Zhang. 2023. LangGPT: Rethinking Structured Reusable Prompt Design Framework for LLMs from the Programming Language Perspective. *arXiv preprint.*

[2] 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. In *Proceedings of the CHI Conference on Human Factors in Computing Systems Extended Abstracts (CHI '22 EA).* ACM, New York, NY, USA.

[3] Mahan Tafreshipour, Aaron Imani, Eric Huang, Eduardo Almeida, Thomas Zimmermann, and Iftekhar Ahmed. 2025. Prompting in the Wild: An Empirical Study of Prompt Evolution in Software Repositories. *arXiv preprint arXiv:2412.17298.*

[4] Rogelio Cruz, Jonatan Contreras, Francisco Guerrero, Ezequiel Rodriguez, Carlos Valdez, and Citlali Carrillo. 2024. Prompt Engineering and Framework: Implementation to Increase Code Reliability Based Guideline for LLMs. *arXiv preprint.*

[5] Saul Greenberg and Bill Buxton. 2008. Usability Evaluation Considered Harmful (Some of the Time). In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI).*

[6] Alex Endert, Michael Sedlmair, and colleagues. 2020. Human-Centered Visual Analytics. *IEEE Computer Graphics and Applications.*