Semantic Commit: Helping Users Update Intent Specifications for AI Memory at Scale

Priyan Vaithilingam
SEAS
Harvard University
Cambridge, Massachusetts, USA
pvaithilingam@g.harvard.edu

Munyeong Kim
Montréal HCI
Université de Montréal
Montréal, Quebec, Canada
kim.munyeong@umontreal.ca

Frida-Cecilia Acosta-Parenteau
Université de Montréal
Montréal, Quebec, Canada
frida-cecilia.acosta-parenteau@umontreal.ca

Daniel Lee
Adobe Inc.
San Jose, California, USA
dlee1@adobe.com

Amine Mhedhbi
Polytechnique Montréal
Montréal, Quebec, Canada
amine.mhedhbi@polymtl.ca

Elena L. Glassman
SEAS
Harvard University
Cambridge, Massachusetts, USA
glassman@seas.harvard.edu

Ian Arawjo
Montréal HCI
Université de Montréal
Montréal, Quebec, Canada
ian.arawjo@umontreal.ca

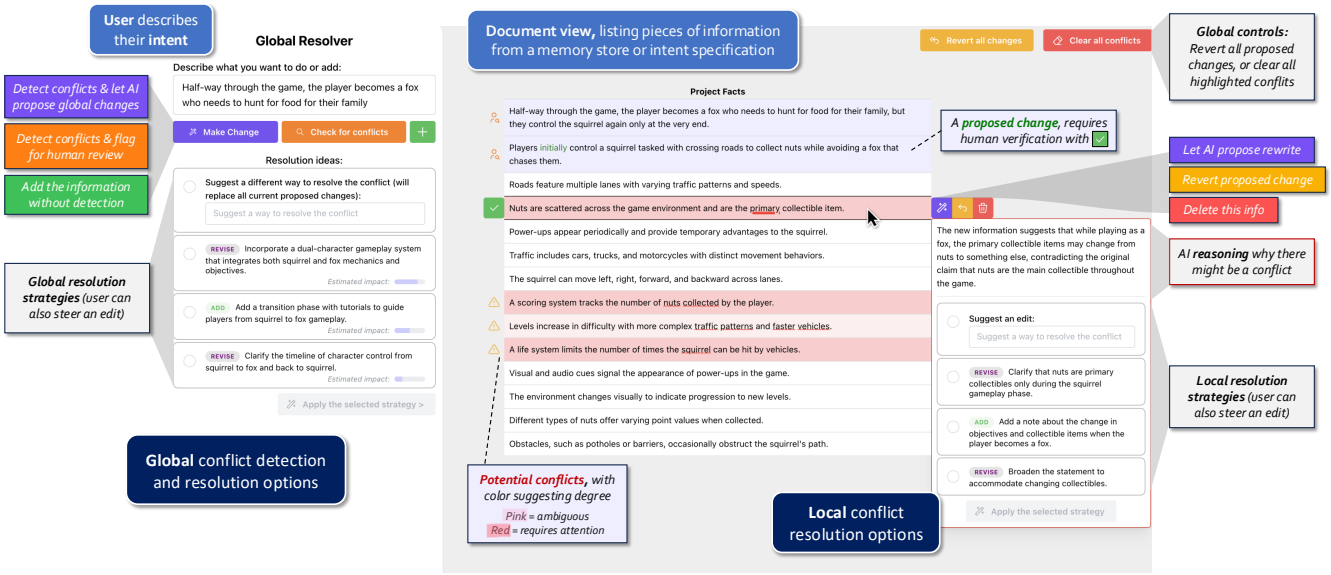


Figure 1: Our SemanticCommit interface, providing users myriad ways to detect and resolve conflicts at global and local levels. Our prototype was used as a probe to better understand the needs of users for integrating new information into lists of prior information akin to AI agent memory or requirements lists. The screenshot depicts a short list describing a "Squirrel Game," where the user is integrating a new feature. Potential conflicts are highlighted in red and pink to mark degree, and the AI has added a new piece of information to the store and proposed an edit to another piece, both marked for human verification.

UIST '25, Busan, Republic of Korea

ACM ISBN 979-8-4007-2037-6/25/09
https://doi.org/10.1145/3746059.3747778

Abstract

As AI agents increasingly rely on memory systems to align with user intent, updating these memories presents challenges of semantic conflict and ambiguity. Inspired by impact analysis in software engineering, we introduce SemanticCommit, a mixed-initiative interface to help users integrate new intent into intent

specifications—natural language documents like AI memory lists, Cursor Rules, and game design documents—while maintaining consistency. SemanticCommit detects potential semantic conflicts using a knowledge graph-based retrieval-augmented generation pipeline, and assists users in resolving them with LLM support. Through a within-subjects study with 12 participants comparing SemanticCommit to a chat-with-document baseline (OpenAI Canvas), we find differences in workflow: half of our participants adopted a workflow of impact analysis when using SemanticCommit, where they would first flag conflicts without AI revisions then resolve conflicts locally, despite having access to a global revision feature. Additionally, users felt SemanticCommit offered a greater sense of control without increasing workload. Our findings indicate that AI agent interfaces should help users validate AI retrieval independently from generation, suggesting that the benefits from improved control can offset the costs of manual review. Our work speaks to the need for AI system designers to think about updating memory as a process that involves human feedback and decision-making.

CCS Concepts

• **Computing methodologies** → **Intelligent agents**; • **Human-centered computing** → *Natural language interfaces*; User studies; • **Software and its engineering** → **Requirements analysis**; • **Information systems** → *Information integration.*

Keywords

memory management, AI agents, large language models, impact analysis, human-AI grounding, intent specification

ACM Reference Format:

Priyan Vaithilingam, Munyeong Kim, Frida-Cecilia Acosta-Parenteau, Daniel Lee, Amine Mhedhbi, Elena L. Glassman, and Ian Arawjo. 2025. Semantic Commit: Helping Users Update Intent Specifications for AI Memory at Scale. In *The 38th Annual ACM Symposium on User Interface Software and Technology (UIST '25), September 28–October 01, 2025, Busan, Republic of Korea.* ACM, New York, NY, USA, 18 pages. https://doi.org/10.1145/3746059.3747778

1 Introduction

In the near-future, people may coordinate with AI agent systems through project-specific documents that represent accumulations of user intent [66, 92, 100]—lists that we call **intent specifications**. These human-readable accumulations of design requirements, user goals and preferences reify common ground [8, 19, 92] between humans and AI systems, grounding AI decision-making by keeping track of details and goals, surfacing implicit assumptions made by AI, and acting as a intermediate representation of an AI system's 'understanding' which the user can inspect and edit (Figure 2).

We dream of a world in which people can make **semantic commits**: committing ideas and details to projects like they commit code, and dealing with the "merge conflicts" that may occur. One key challenge standing in the way of this paradigm shift is ***integration***: how to responsibly and verifiably integrate new information into a repository of natural language [92], e.g., to update an AI agent's memory of user intent in a reviewable, concise, and accurate manner, such that the memory remains aligned. How can technology assist with the integration of a new piece of information into an existing repository *at scale* (e.g., a design document, a requirements list, documentation, a wiki, novel, etc.)? The new information may conflict with prior information—something may become *inconsistent* or *contradictory*. Changing existing information can incur the same effect. We frame this challenge for the community as ***semantic conflict detection*** and ***semantic conflict resolution***, since it operates at the level of *semantics* and *concepts*, unlike past techniques that operate on pre-defined structure and syntax.

Over brief time-frames and short documents, simple methods—such as using LLMs to regenerate entire documents or apply string replace operations [56]—can perform edits, but as humans interact with agents over long time-frames and complex projects, these methods cease to function at scale. Simple vector store architectures, seen in retrieval-augmented generation (RAG), also face challenges, since detecting semantic conflicts frequently requires multi-hop reasoning, a well-known failure mode [25, 36]. *How* to resolve a conflict is also often subjective [14, 49], and therefore a problem for HCI, as for example, particular conflict resolutions may incur *cascades* where solving one problem creates another. Systems thus need ways not only of *identifying* conflicts and inconsistencies efficiently and accurately at scale, but of *interactively assisting users in conflict resolution* in a way that a) helps users reflect and b) foresee the impact of changes, c) only makes the necessary changes without touching other information, and d) minimizes user effort while maximizing changes' alignment with user intent. Downstream AI systems could use conflict detection results to, e.g., decide whether to perform grounding acts [83, 85] such as request for clarification.

To help researchers better understand the problem of updating AI memory of user intent in an aligned manner, in this paper, we provide several contributions to the literature. We:

(1) Define the term *intent specification* to name grounding documents that coordinate with AI agents, such as user-defined "memory" lists for Claude Code [2].
(2) Provide design goals for AI-assisted interfaces for semantic conflict detection and resolution, inspired by related literature such as impact analysis in software engineering.
(3) Develop an interface, SemanticCommit, iterating its design over two pilot studies. Our system implements a range of affordances for conflict detection and resolution and is intended as a probe of user behavior.
(4) Introduce an architecture for semantic conflict detection at scale. Our approach uses induced knowledge graphs, adapting emerging architecture in retrieval-augmented generation (RAG) [36]. To test and compare our approach to prior approaches in the literature, we also provide an initial evaluation dataset ("evals" [87]) across three domains.
(5) Provide empirical insights from a within-subjects user study, examining how users detect, understand, and resolve conflicts when updating intent specifications for AI memory, comparing SemanticCommit to OpenAI's ChatGPT Canvas.

Our findings suggest that AI agent interfaces should enable users to perform ***impact analysis***, separating retrieval from generation—steps that are currently conflated in many AI-powered software engineering IDEs. Surprisingly, although users appeared more engaged when using SemanticCommit, they did not report

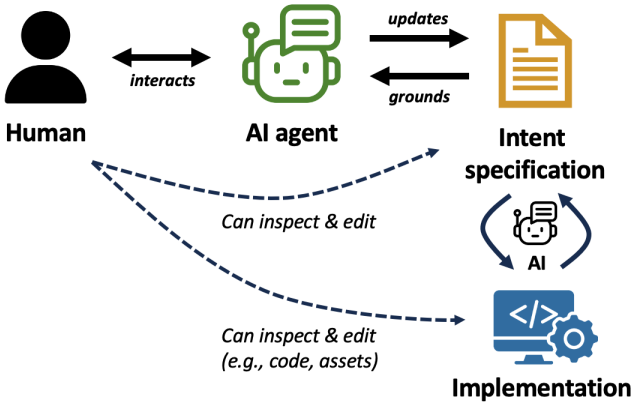


Figure 2: A high-level depiction of our envisioned interaction between humans and AI assistants for long-term projects. The human-readable *intent specification* serves as an intermediate layer for enhancing common ground between the human and the AI, and grounds the AI's decision-making. We assume future AI agents will have a similar intent specification layer. Our project squarely concerns how the AI *updates* this memory in a robust, verifiable manner, and in the process might surface conflicts to the user to get their feedback in resolving them.

significantly higher workload than the more automated Canvas UI. This suggests that **the benefits of improved control can offset the cost of manual review,** possibly by shifting user workload away from metacognitive demands [89] that users face when prompting in open-ended chat, towards the demands of the actual task, such as reviewing conflicts.

2 Motivation: Intent Specifications Ground Human Coordination with AI Agents

Humans are increasingly managing and validating the outputs of AI systems that implement entire software systems like games, websites, and apps. To reduce risk and align AI decision-making in user preferences, human-readable documents are emerging as a mechanism to create and maintain common ground [19] between humans and AI systems acting on their behalf [86].

Numerous examples are emerging of this interaction paradigm. The AI-powered programming IDE Cursor, for instance, can ground its behavior in user-made "cursor rules"—markdown documents that AIs read to ground their behavior in user preferences—at both project-specific and global levels [21].[1] Rules range from sweeping commands, like "never use apologies," to the highly particular, like "use vectorized operations in pandas and numpy for improved performance." Users develop these rules over time across many interactions. Anthropic has also adopted this paradigm: with the Claude Code agent, users create `CLAUDE.md` files listing project- and global-level directives; Anthropic's own "memory best practices" tell users to format memories as "bullet points" and reminds them to manually "update memories as your project evolves" [2]. These "memories" help Claude Code "remember project conventions, architecture decisions, or coding standards that we want to reference across sessions" [96]. Not to be outdone, the CEO of Windsurf—a competitor to Cursor—just announced a yet-to-be-implemented "auto-generated memories" feature where these memories of user intent are automatically updated by an AI, which will inevitably encounter the very challenges we discuss here.[2]

Everyday users are also increasingly coordinating with AI systems through lists of requirements expressed in natural language. For instance, users are generating games from specs that resemble lists of software requirements. Here is an excerpt from a real user,[3] to give readers a sense of how these rules appear in practice:

- The dog barks when the player clicks or taps on the screen, making the sheep move faster
- Sheep should react realistically to the dog's presence
- When frightened, the flock should scatter

This user's example, which in total has 27 requirements, is only the *start* of an interaction with an AI agent. As the user interacts and projects grow in complexity, future AI systems will need to assist in the extension and updating of these rules and details.

We call these lists—cursorrules, `CLAUDE.md` files, user directives, AI memory of user intent, etc.—**intent specifications**, adapting and broadening the notion of *requirement specifications* in software engineering.[4] Intent specifications are ***evolving, comprehensible documents of user intent that ground AI decision-making and reify common ground between humans and AI systems***. We introduce *intent specification* to underscore that such documents may not only cover design details or software requirements, but how the AI should communicate to the user, who the user is, the user's goals and dreams, etc. Said differently, an intent specification is not only a description of user intent, but may also include information that helps an AI agent *assume* user intent—i.e., background, assumed preferences—accelerating the establishment of common ground. However, unlike a general memory store—which could be an extensive collection of all interactions—intent specifications' purpose is to be reviewable, comprehensible and digestible, to be inspected and edited by humans. In response to edits, the AI will adjust its behavior, such as revising an implementation; the AI may also amend the specification in response to the user or to better reflect new implementation details and assumptions [92, 100] (Fig. 2).

As we mention in our introduction, the *integration* of new information into an intent specification is not (always) straightforward. People and ideas change. New information may conflict with prior information, especially as projects and user interactions stretch from days to months and years. Proposed approaches to memory with RAG architectures, which store all memories verbatim, do not account for these potential conflicts (e.g., [1, 44, 74]).

To illustrate the nuances of semantic conflict resolution, consider two chunks of information regarding a warp drive in a game. One chunk states that "the warp drive can exceed the speed of light,"

[1] People have started crowd-source these rules: the "Awesome CursorRules" repository and CursorList.com include hundreds of rules lists, contributed by everyday users, indexed by programming language, libraries, and use cases. See, e.g., https://github.com/PatrickJS/awesome-cursorrules.

[2] https://x.com/vitrupo/status/1900146068030914740

[3] https://github.com/vnglst/when-ai-fails/blob/main/shepards-dog/README.md

[4] Leveson [57] introduced the term "intent specification" in the context of software engineering to track requirements. Our definition of intent specification is broader, and more loosely defined to support a wide variety of documents and scenarios.

while the other chunk specifies that "when the ship's warp drive is activated, it first moves slowly and then suddenly operates at a very high speed." If a new chunk stating "no material can move faster than the speed of light" is added, then a contradiction arises with the first chunk. However, it is unclear whether it contradicts the second about a "very high speed"—which in the full context implies, but does not clearly state, faster-than-light travel. Any attempt at fully automated semantic conflict resolution is subject to debate in such situations because of the ambiguity of natural language.

Our chief insight is that integration of new information into an AI memory store is a *process* that can require *interactive*, human-in-the-loop feedback for aligned resolution. Both users and AI systems need support for semantic conflict *detection*—to understand when a conflict has taken place, with what information, and how—as well as *resolution*, as resolving conflicts could involve the revision, addition, or deletion of existing information in a manner that may add or change details. To resolve conflicts, practical assistance may require *clarification of ambiguities*, *constructive negotiation of ideas* [92], or *delegation of tasks* [88, 100]. However, it remains unclear how users update, and want to update, intent specifications in practice. **What affordances should AI memory interfaces have for the process of integration? How do users think about semantic conflicts and what needs do they have for resolving them with confidence? How can we help users update intent specifications like `CLAUDE.md` files with confidence?** Before returning to these questions, we first connect to existing literature that can help shed light on this emerging paradigm.

2.1 Related Work

2.1.1 ***Design documents to coordinate work in human teams.*** The rise of intent specifications mirrors what human teams already do to coordinate actions. Across many domains—from product design, to game development, software engineering, UX design, and animation—people standardize the vision (look, feel, goals, plans, etc) of a project in documents that are often called "design documents" [11]. These documents serve to establish and maintain *common ground* between parties [19], ensuring each member's actions remain *grounded* in shared understanding and objectives. In animation, the design document takes the form of model sheets [69], which standardize how to draw characters and other assets. Game developers use "game design documents" (GDDs) to keep development grounded across a team [20]. In software engineering (SE) and UX research, need-finding processes produce a "system requirements specification" that is passed off to the software team [39, 63]. Programmers develop "coding style guides," or norms around naming conventions, comments, and writing tests, as well as "contributing guidelines" that establish rules for external contributors. These documents serve to externalize, standardize, and coordinate the high-level *intent* of a team—its objectives, details, procedures, and feel—and are revised as the project proceeds [20].

Intent specifications, while less formal than code, are a lot like software: they encode dependencies among ideas that constrains future evolution. These dependencies may be "sequential" (i.e., a custom term is defined then used later on) or heterarchical. As interactions continue, teams "commit" new information to the intent specification, and must resolve outdated or inconsistent dependencies. Maintaining consistency is paramount, because the very purpose of these documents is to enforce consistency and define standards. For instance, from a study of game designers: *"[She] writes the GDD as she is designing the game... taking anything out of... the GDD that conflict with the consistency of her plot. [She]... wrote her entire GDD... as a list, which she frequently added and deleted from as she designed the game"* [20, p.9]. The field of requirements engineering in SE also stresses the importance of clarity, conciseness, completeness, and unambiguous requirements [23], with "commission (inclusion of irrelevant or incorrect details) and omission (exclusion of necessary details)" as additional concerns [66]. However, revising requirements accurately may require consideration of intent information outside formal specifications [57].

2.1.2 ***Impact analysis in software engineering.*** In software engineering, visibility on the ramifications of a feature change or addition is called **impact analysis**: identifying what parts of the shared context (code repository) will need to be amended, for the change to occur [6]. Impact analysis "predict[s] the system-wide impact of a change request before actually carrying out modifications to the system... so that appropriate decisions related to the change request can be made, such as planning, scheduling and resourcing... The potential impacts are then interactively validated by the user" [38, p. 174-181]. Impact analysis is complemented by feature localization (retrieving relevant context to inform impact detection) and followed by change propagation (actually making changes to code) [24]. In our non-coding context, we might interpret these steps as first retrieving relevant info, then detecting what information is impacted, then helping users make changes. Note that impact analysis is a *sense-making* task, less a coding one: impact analysis primarily surfaces system entities and dependencies that may be affected by a proposed change, although some tools do help users change the underlying code [50].

2.1.3 ***Conflict detection and resolution techniques and interfaces.*** Conflict detection and resolution are classic problems in computing, usually arising in contexts of collaborative information processing to merge asynchronous changes. Engineers have developed techniques such as version control [70], groupware platforms and database synchronization [55, 61, 75], and concurrency-control systems [26, 37]. The 'git' command-line interface [76], for instance, is a popular version control system where users make "commits"—a change to a file repository, alongside a pithy message—to keep track of changes. To help users understand differences between versions, many interfaces present "diffs" [46]. Numerous LLM writing tools have been proposed that incorporate diffs, spanning various areas such as story writing [17, 18, 99], screenplay writing [68], poetry [32], dictation [60], and argumentative [94, 101] and scientific [29] writing. InkSync [56], for example, is a prototype for executable and verifiable text editing with LLMs, which shows LLM edits as diffs on the document. To make diffs, InkSync uses string-matching: it relies upon the LLM to reproduce extracts of text to change, and then specify the replacement; this method is also used by Anthropic Artifacts [78].

These conflict algorithms operate on syntax, rather than semantics. Current interfaces provide little to no support for users to see

the *semantic* ramifications of their changes on the rest of the document. An example is editing a scene in a novel: would changing a lunch between two characters to a dinner setting impact something hundreds of pages later? These semantic conflicts require dedicated support to detect, visualize, and resolve. Semantic conflict resolution interfaces must go beyond visualizing what changes *were* made, to what changes *could be* made, *where* they should be made, and *what the effects* might be. This resembles feed*forward*: affordances that help the user foresee the impact of an action [67, 93].

2.1.4 ***Human-AI collaboration grounded in shared, intermediate representations.*** HCI has, in a sense, always been about communicating to machines through shared representations [4, 41]. However, past shared representations had to be strictly formalized—into programming languages, domain-specific languages (DSLs), schema, etc.—to ensure deterministic outcomes. These shared representations helped negotiate agency between humans and machines [40], but ultimately could only go so far, as end-users always resigned some agency to the representation designer(s) [58, 92].

Today with LLMs, we are less limited by this constraint, and solutions to the problem of human-machine communication might be better found in cybernetics theory [9] than static formalism. Effective human-AI communication relies upon tight feedback loops [100], but also offering humans *control* in the form of transparency over AI understanding and context. Along these lines, emerging HCI research envisions that AI systems will be grounded by shared representations of a more informal nature—lists of directives expressed in natural language [66, 92, 100]. Some researchers even argue that these informal expressions of intention will be "all you need" [80, 81]. For instance, Vaithilingam et al. imagine a hypothetical AI game design assistant where the AI "[integrates user] choices into the project plan" [92], while Zamfirescu et al. explore an iterative design loop with an AI agent that "tracks decisions that the human has made" and "surfaces decisions the LLM has implemented in the code" in a running list [100]. Ma et al. define "requirement-oriented prompt engineering," helping users generate a "clear, complete requirements" list prior to prompting an AI to implement software. They stress that making a good list requires skill and support [66]. These projects speak to the need for targeted support for updating intent specifications that ground AI behavior. Ensuring alignment with user intent (e.g., by reducing inconsistencies) is critical: miscommunications are the chief reason for breakdowns with AI agents [88], and the potential of failure compounds as communication continues without addressing misunderstandings [85].

2.1.5 ***Natural language inference, reference ambiguity, and knowledge graphs.*** Finally, the technical side of our work relates to natural language inference (NLI), a research area in NLP [49] that concerns the classification task: Given two sentences—a premise sentence and a hypothesis–does the hypothesis sentence follow from (*entailment*), *contradict*, or bear a *neutral* relationship to the premise? HCI scholars have applied NLI to data annotation [95], in-situ summaries [62], and LLM response consistency [16]. Our discussion of NLI provides additional context for our system design.

Detecting conflicts is by no means an objective task; human annotators frequently disagree [14, 49]. Jiang & de Marneffe [49] investigated reasons for human disagreement during NLI classification and argue for a fourth category, "complicated," which increased model recall. Their goal was "not necessarily to maximize accuracy. A model that can recall the possible interpretations is preferred to a model that misses them" [49, p. 1365]. Chen et al. [14] also introduced a fourth category, "ambiguous," to denote situations where "it is unclear whether the claim and the evidence refer to the same context... [i.e.,] there exist multiple possible assignments or interpretations." The authors refer to this as *reference ambiguity*—when the two sentences *could* coexist, but it is unclear—and found that it explained many annotator disagreements [14].

NLI appears in recent discussions on the future of SE, which propose that LLMs may be used for formal requirements analysis [7, 90]; e.g., Lubos et al. [65] studied how LLMs can provide quality feedback on requirements, while Fantechi et al. [27] analyze an LLM's ability to detect inconsistencies. Importantly, Fantechi et al.'s method simply fed in the entire list into the LLM and asked it to detect conflicts; they found that LLMs could only process "short requirement documents" this way. They conclude that despite lower accuracy compared to humans, "manual detection of inconsistencies is more expensive," growing quadratically with list size, "whereas examining [LLM] answers to distinguish true from false positives is a much lighter task" [27, p. 338]. Fazelnia et al. [28] also trained an NLI model to analyze requirements lists, and concluded that NLI models suffered in multi-hop conflict detection.

To better capture dependencies among requirements, SE researchers proposed ontology extraction, where a system generates a knowledge graph [3, 23] capturing relationships between requirements, a method introduced by Kaiya and Saeki [51]. For instance, Hsieh et al. [45] extract a domain-specific ontology by mining information from textbooks; while Dermeval et al. [23] use a web ontology as a visualization technique to help SWEs in writing more "correct," "complete," "consistent," and "unambiguous" software requirements. Such graph-based visualizations have also supported impact analysis; for instance, *Wolf* [30] shows the impact of proposed changes via a dependency graph. This work informed our decision to use knowledge graphs (Section 5).

3 Design Goals

Here we chronicle our initial design goals for SemanticCommit, as well as our revised goals as the result of two pilot studies.

We wanted to design a prototype to better understand what interface affordances users need to facilitate robust and trustworthy updates to intent specifications in a manner that 1) maintained their alignment with user intent and 2) kept unrelated information untouched. We thus went for a kitchen-sink approach: to include a variety of features that users may, or may not, choose to engage in, that seemed reasonable based on past conflict resolution interfaces, and observe what features users find most important and how they use these features in concert. Based on our review of past literature on conflict detection and impact analysis, we identified an initial set of design requirements for SemanticCommit:

- **Foresee impact**: Literature on impact analysis highlights the importance of letting users predict the effects of a change [6, 24, 30]. Our system should allow users' to *foresee* potential downstream impact without actually proposing any changes.

- **Detect conflicts**: The system should assist the user in *detecting* potential conflicts or contradictions, between existing information and the new information being introduced [26, 27].
- **Understand conflicts**: The system should provide explanations to help the user understand and reason about the conflicts. Explanations support impact analysis sense-making [50] and have been shown to improve trust in intelligent systems [59].
- **Assist conflict resolution**: Like some impact analysis tools help users make code changes [38], our tool should help users resolve semantic conflicts at both global (i.e., entire document) and local levels. The AI should suggest possible *resolution strategies*, while leaving users free to manually edit at any time.
- **Leave non-conflicting information unchanged**: Integrating new information should only touch pieces of information that are in conflict, and leave others unchanged.
- **Support local changes**: Users should be able to inspect proposed changes in situ and decide whether to accept, reject, or further revise (such as via a "diff" view). Design guidelines on human-AI agency and prior work also stress on giving users' fine-grained control over AI suggestions [40, 56].
- **Revert changes**: Due to the stochastic nature of LLMs, and the complexity of semantic commit task, it is vital to provide ways to recover from AI failure [10, 98]. Proposed changes (edits) should be able to be reverted at global and local levels (i.e., to cancel specific revisions or back out from a wide-scale change).
- **Work at scale**: To support a wide range of contexts and long-term usage, the system should operate at scale — handling lengthy intent specifications without introducing latency. Users should not have to worry about document length.

Note that there are other design goals which are important to *general* user interfaces for managing AI memory—such as version control, branching, and navigation (see *Memolet* [97])—but we do not consider them here.[5]

It is critical to note that while some design goals overlap with document editing interfaces, *a primary goal of our research is to produce design implications for situations where there may be no manual document view*—e.g., situations where the user is communicating entirely through a chat UI, where the AI is managing the intent specification for them and may surface conflicts in a different, constrained manner (and decide whether, when, and how to do so). We intend that semantic commit will eventually be *a programmatic API* for helping developers update intent specifications that ground AI agent systems. Thus, we designed our interface to purposefully constrain editing to separate pieces of information—"memories," details or rules—rather than enabling the user to perform freeform writing tasks (i.e., think OpenAI ChatGPT's memory store [73], rather than Microsoft Word).

3.1 Early Prototype and Pilot Feedback

Our explorations went through substantial iterations and prompt prototyping over a period of eight months, evolving in response to two pilot studies and progressing from a card-based interface to a list of texts. We chronicle our early design and formative studies.

From our design goals, we built an initial prototype, where pieces of information were written on cards akin to post-its and could be freely moved. This interface was limited to prompting our conflict detection feature, and studied how users would integrated changes into (a chunked version of) the game design document for the unpublished LucasArts game Labyrinth [31]. In this early prototype, cards were only marked as either in conflict or not.

We ran one pilot study with five users of our card-based interface, and a second with four users of a revised interface. Key takeaways:

- The color-coding of cards marked as conflicts drew user attention sometimes entirely away from manual inspection of non-marked cards. Possibly in reaction, all pilot users **preferred higher recall over precision**. They viewed false negatives (missed detections of true conflicts) as catastrophic, while false positives were easily handled with a quick skim.
- When asked, participants expressed a preference for a **structured, sequential document view,** over the cards interface. One reason may be that users became fixated on sorting the cards, another could be that documents are more familiar.[6]
- Users wanted finer-grained insight into the **degree of conflict**. Users wanted a quick visual way to understand where they should spend their limited attention.
- Participants would **iterate on their prompts** to the conflict detector and resolver, in case the output did not exactly match their intent. It seemed less important that AI sometimes made mistakes, and more that they were easily fixable.
- In post-interviews, users suggested that **the degree to which they trusted the AI depends on their degree of investment in the information**. If they felt invested, they would trust the AI *less* to make direct changes.

In response, we added more design goals to our initial list:

- **Recall-first**: Favor recall over precision for conflict detection.
- **List view**: The system should prefer more standard document views, which present manageable chunks of information sequentially, than open-ended diagramming canvases.
- **Visualize degree**: The system should help users understand the degree or importance of a conflict at a glance.
- **Help user recover from AI errors** [71]: The system should support fast iteration, in case of AI mistakes, by allowing the user to steer the detector or resolver with a prompt.

Based on these goals and feedback, we adjusted our interface and study protocol. The most important change we made was how strict our conflict detection retriever and filtering prompt was: we loosened it considerably, to enhance recall at the expense of precision. We also added a third classification, "ambiguous," to imply a lesser "degree" of conflict, a decision solidified after review of papers in NLI [14, 49]. Ambiguous conflicts appear as a softer pink color to imply reduced importance, directness, or confidence that the information is truly in conflict.[7] This prompt engineering was a delicate balance: too restrictive and the system tends to only rarely include ambiguous options; too generic and it flags almost all pieces

[5] In particular, in real-life intent specifications like Cursor Rules, users sometimes group lines together; we chose a simple list to avoid complexity in our initial design.

[6] This preference seems to map to the "cursorrules"-like situations of editing Markdown documents, which weren't popular at the time of our pilot.

[7] As we rely upon LLMs, this is not an exact science. Indeed, the aforementioned NLI papers also show that even with human annotators, there is little consistent reason why something is categorized as "ambiguous" or "complicated" [14, 49].

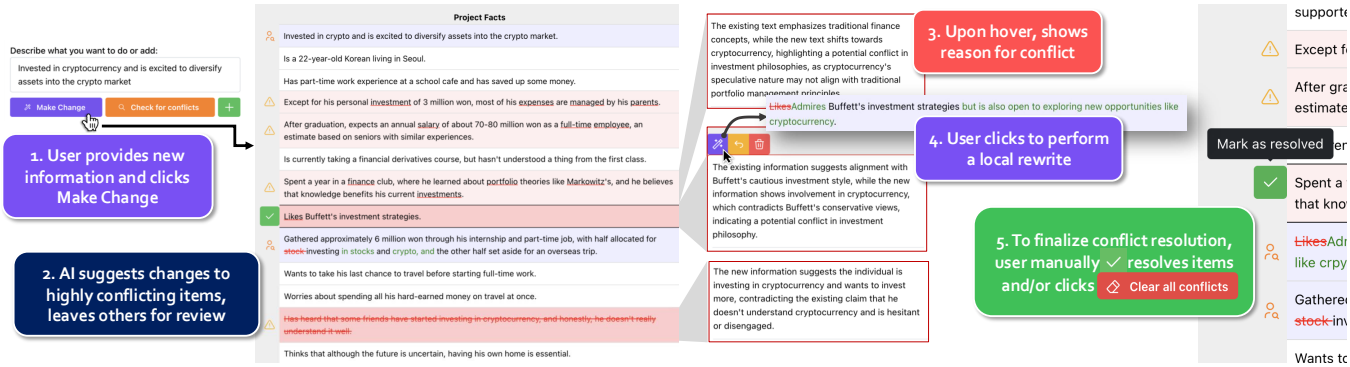


Figure 3: Example of our SemanticCommit workflow, showing one process of integrating new information into an AI memory of the financial habits of a South Korean student. 1. The user has described a new piece of information and pressed Make Change. 2. SemanticCommit detects conflicts and suggests changes to items it deems the most conflicting, leaving other conflicts for human review. 3. The user hovers over conflicting items to view the AI's reasoning. 4. For one item, they click a button to let the AI make a local rewrite. The user can continuing editing, manually revising, reverting suggested changes, or deleting items at will. 5. When they feel done, they manually resolve items and/or clear remaining conflicts with a global action. (Alternatively, the user could have clicked Check for Conflicts to only perform detection, then handled conflicts locally.)

of information as potentially conflicting. We iterated on our system decision choices with more confidence by validating changes against a custom evals dataset, which we discuss in Section 5.

4 SemanticCommit User Interface

Here we overview our final design and walkthrough examples of usage. Figure 1 shows our prototype, with global operations:

- Check for conflicts The Check for Conflicts button provides the ability to perform *impact analysis* [6], which only highlights potential conflicts without suggesting changes, allowing the user to get a sense of how much effort a change might require. They may choose to manually resolve each conflict, or back out and decide upon a different course of action.
- Make Change The Makes Changes button performs Check for Conflicts then lets the AI propose a rewrite. The back-end uses the same procedure as check for conflicts, then performs a global rewrite of all detected conflicts in order to incorporate new information. Critically, the LLM can decide not to rewrite information, even after it has been flagged (this is to avoid redundant changes); flagged conflicts that were not changed remain highlighted for human review.
- The Add Info button allows the user to manually add a piece of information.

More features are shown in Fig. 1. Local conflict resolution options include letting the AI rewrite, steering a rewrite, applying a suggested resolution strategy, reverting a change, and deleting the information. Global conflict resolution options complement this, allowing the user to steer a global rewrite via a prompt or choose a suggested resolution strategy. Users can also perform global actions to Revert All proposed changes, or Clear All Conflicts (putting all pieces of information back into a neutral state). Finally, red underlines are an experimental feature that suggests words which contributed the most to the conflict (in Fig. 1, "primary" is bold-underlined to imply that nuts are likely no longer the primary collectible when the player is a fox).

The only feature missing from our figure is a "request intent clarification" pop-up that appears when the AI classifies a user request as potentially resulting in many changes (Section 4.1.3). We observed that high-impact changes, like changing a game's setting from Mars to Venus, could incur many second-order effects and deserves an additional clarification round before proceeding with (relatively more costly) conflict detection.[8]

4.1 Walkthrough of Usage

Let's walk through three examples of system usage in different domains: an investment advisor agent, updating the directives for an AI software engineer, and updating a game design document.

4.1.1 ***Updating memory of an investment advice agent.*** As a simple example, imagine an AI agent for investment advice has accumulated a memory of the user, a South Korean college student, after many chat sessions. These include details such as financial goals, life events, employment history, etc. Now this user invests in a cryptocurrency and expresses excitement about diversifying more assets into crpyto. Using SemanticCommit, we add this piece of information to the list, and the system detects potential semantic conflicts which may require human review (Figure 3). A user clicks the "Make Change" button, which adds a new piece of information (deducing that it should do so, which is not always done), detects conflicts, then proposes changes to ensure the memory remains consistent with the new information. One line it proposes deleting entirely, another it rewrites, and others it flags for review.

Notice how semantic conflict detection leveraged the LLM's general knowledge: a mention that the user likes Warren Buffet's investment strategies is highlighted as a potential conflict. Buffet, a

[8] Our prompt to the AI for this step is simple and more of a prototype: here, we simply feed the entire context in alongside the user's change, and ask the AI to provide a question if it decides the change is high-impact enough to deserve clarification.

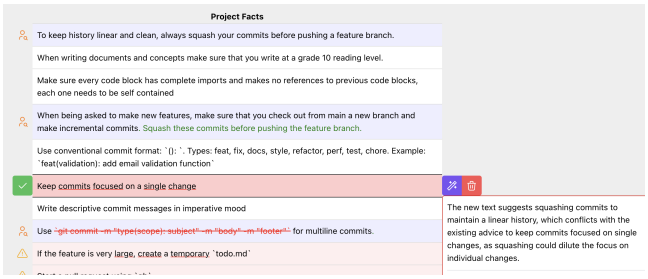
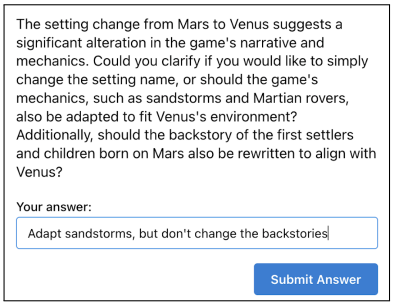


Figure 4: Cursor Rules [90] adapted from the Instructor library [48], loaded into our SemanticCommit UI. The user has added a new directive to squash commits before pushing a feature branch. The system adds the new rule to the top, makes a revision, and flags other lines as potential conflicts. One change is in error, which the user can spot and revert.

famous investor, avoids cryptocurrencies and has declared them "rat poison squared" [54]. Clicking the Let AI Propose Change button on the *local* information, a slight rewrite is proposed where the claim is softened (Step 4 in Fig. 3).

4.1.2 ***Updating rules for an AI software engineer.*** Consider a user has a list of Cursor Rules, describing how an AI software engineering agent should behave in a code repository (Figure 4). (Here we use real cursor rules adapted from `Instructor` API's open-source repository [48].) The user adds a new directive, common to software engineering practice: "To keep history linear and clean, always squash your commits before pushing a feature branch." SemanticCommit highlights "Keep commits focused on a single change" in red, indicating direct conflict, and "If the feature is very large, create a temporary `todo.md`" in pink, indicating an ambiguity. The first is unclear how to resolve: removing it seems unwise, but keeping it unchanged incurs confusion. The AI has also added a mention of squashing commits, after the line, "When being asked to make new features, make sure that you check out from main a new branch and make incremental commits."

4.1.3 ***Changing a game design document.*** Finally, imagine a game designer has a design document for a game set on Mars, which an AI agent implements. After some playtesting, they decide that Mars is overused in sci-fi narratives, and communicate that they want to switch the setting to Venus. Here, the AI has estimated that the change is significant enough to request further clarification from the user before continuing:



The user provides clarification, and conflict detection proceeds. The AI makes the most obvious changes—changing the term "Mars" to "Venus," mainly—while flagging other potential conflicts for review. A subtle semantic conflict, that Mars has sandstorms but Venus does not, is detected and changed to a more generic "storm", steered by the user's clarification:

~~Mars~~Venus's ~~sandstorm~~storm events greatly impact terrain, facilities, and crops each time they occur, but the player can unlock devices to prevent damage through gameplay.

These examples illustrate that conflicts: a) may require general world-knowledge to detect, b) may be hard to resolve, and c) *how* to resolve a conflict can be a matter of creative decision-making. Resolving even a single change accurately is important, as unresolved conflicts can cascade as more changes are made. Using this system, we also *learned* why some conflicts occur—the Buffett example above was not something we were aware of—or could be forced to reckon with second-order effects, such as re-thinking the sandstorm mechanic to better fit the planetary conditions of Venus.

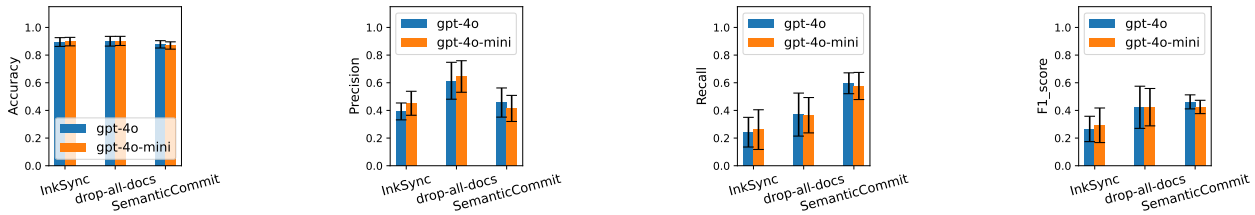
Note finally that our system *does* may mistakes—conflicting information can be missed, as our technical evaluation shows; conflict detection and retrieval are stochastic; reasoning can sometimes be superfluous; and in practice, some knowledge base domains can benefit from adding a temporal feature to information (i.e., a limited duration where a rule holds). However, we believed the system was strong enough to run a user study in order to better understand where further efforts should be directed.

4.2 Implementation

SemanticCommit is implemented in React and TypeScript, with a Flask Python backend for our knowledge graph-based retrieval architecture (described in Section 5). We iterated on prompts using ChainForge [5] by setting up an evaluation pipeline against our datasets, which allowed us to observe the effects of prompt changes and model choices. There are *many* prompt-based functions, from the user intent router, to conflict detection, local and global revision, underlining "highly conflicting" words, and suggesting resolution strategies. We chose GPT-4o for performance and latency reasons, as it performed optimally against our evals. Further details on our development process and system are in Supplementary Material.

5 Back-End for Semantic Conflict Detection

We implement a back-end system to drive the interface of SemanticCommit. The back-end's primary goal is to enable conflict resolution at scale. During early prototyping, we found that simple methods—giving the entire context to the LLM, or generating string-replace operations [56]—were prone to missing conflicts. These techniques rely on a single prediction, which takes the entire memory store and produces either a rewritten version or a set of suggested edits. Rewriting the document frequently introduced superfluous changes unrelated to conflict resolution and can have large latency due to output size. As an alternative, we considered vector store retrieval from simple retrieval-augmented generation (RAG), but this method is known to perform sub-optimally in cases requiring multi-hop reasoning [25, 36], where dependencies among chunks are heterarchical (understanding one chunk can depend upon considering it alongside several others) .



(a) Accuracy (Mean ± StdDev) **(b) Precision (Mean ± StdDev)** **(c) Recall (Mean ± StdDev)** **(d) F1 Score (Mean ± StdDev)**

Figure 5: Comparison of SemanticCommit using a knowledge graph with PageRank relevance assessment and then classification to two baselines: (i) DropAllDocs: takes all documents in context to classify them without a retrieval stage; and (ii) InkSync [56] implementation, reformulating the prompt to our context. The comparison is across all benchmarks in Table 1, averaged with st. dev. bars, for the `GPT-4o` and `GPT-4o-mini` models. Our method, *kg-pagerank*, achieves higher recall with similar accuracy.

To tackle these limitations, we implement the back-end using a knowledge-graph (KG) RAG architecture [36] consisting of two phases: pre-processing and inference. The pre-processing phase constructs a KG by extracting entities from a collection of input documents in the memory store and linking them. Each of the entities keeps track of the relevant document from which it was extracted. The inference phase detects semantic conflicts using a multi-stage information retrieval (IR) pipeline. The IR pipeline takes an edit action (whether it is an insertion or a modification to the memory store) and produces a list of chunks of information in conflict. It contains two stages: (i) *retrieval*: finds relevant chunks of information using the KG in a single-step to avoid error propagation. In order to minimize relevance assessment issues, we apply a PageRank-based relevance ranking over the KG, akin to HippoRAG [36]; and (ii) *conflict classification*: identifies from the retrieved chunks of information which ones are in conflict with the edit. Based on NLI literature (2.1.5) and our pilot studies (3.1), our detection prompt includes a classification of "ambiguous" (Appendix A).

In the rest of this section, we give an overview of our design considerations and their rationale through an technical evaluation. We highlight that our prototype back-end system, achieves higher recall than the simple methods with similar accuracy.

5.1 Technical Evaluation

Our goal is to technically validate key aspects of our design decisions. We compare our end-to-end system against two simpler methods: (i) DropAllDocs, which adds all documents to the context for conflict classification; and (ii) InkSync [56] which generates a JSON list of string-replace operations. These comparisons allow us to analyze the impact of separating conflict detection from resolution, separating retrieval from conflict classification, and evaluating the performance of different LLMs.

5.1.1 Evals. To conduct our evaluation, we created four small evaluation datasets on three distinct domains:

- **Game Design**: We use two game design documents. The first is from Labyrinth [31] by LucasArts (1986). The second includes excerpts from an original by one coauthor, describing a fictional game set on Mars about the first generation of children born there. The documents are chunked into paragraphs and referred to as the *Labyrinth* and *Mars* datasets, respectively.

Benchmark	Ch	M	CS (Min, Median, Max)
Labyrinth	35	17	(0, 4, 10)
Mars	30	25	(0, 2, 14)
FinMem	30	17	(0, 4, 10)
CursorRules	65	19	(0, 3, 25)

Table 1: Benchmark details including number of chunks (Ch), number of prepared modifications (M), and conflict statistics (CS) (min, median, max) across modifications.

- **Financial Advice AI Agent Memory**: AI agent memory in the style of OpenAI's ChatGPT memories, about the investment strategies, financial situation, and background of a fictional college student living in South Korea (prepared by a South Korean coauthor). We refer to this dataset as *FinMem*.
- **Coding Assistant Rules**: Rules for the Cursor IDE [21], which are intent specifications for coding assistants. A subset of the rules was adapted from the awesome-cursorrules GitHub repository. We refer to this as the *CursorRules* dataset.

Eval domains were chosen to cover three major types of intent specifications from Sec. 2.1. Four coauthors created the evals, and two coauthors manually double-checked all conflicts, a process that took several days. During this inductive process, they discussed with two coauthors on how to classify conflicts into direct, ambiguous, and non-conflicts, and adjusted classifications accordingly (see Appendix F for full details).

For each of these datasets, we introduce updates as insertions or modifications to chunks of information, intentionally introducing varying numbers of conflicts. Table 1 summarizes each of the evals including the number of chunks, the number of updates to apply as part of the eval, and statistics on the number of conflicting chunks per update (min, max, and median). These initial evals served as a foundation for prototyping our approach and preparing user studies.

5.1.2 Experiments and discussion. We compare our approach with the two baselines: DropAllDocs and InkSync. We run end-to-end on our four eval datasets using `GPT-4o` and `GPT-4o-mini` and report the mean ± stddev for accuracy, precision, recall, and F1 scores for the three approaches in Figure 5.

Our results show that SemanticCommit achieves higher recall (1.6× and 2.2× higher) than DropAllDocs and InkSync, respectively, while retaining similar accuracy. This better addresses

user preferences mentioned in our pilot studies and Related Work (2.1.5), reducing risk of false negatives. Additionally, our system matches the F1 score of DropAllDocs, outperforming InkSync by 1.6×. While its *precision* is comparable to that of InkSync and 1.6× lower than DropAllDocs, we consider this an acceptable trade-off given our emphasis on maximizing recall. Note that our evals are rather skewed with highly targeted conflicts (on average, only a few ground truth items in conflict when integrating new information), and accuracy can be misleading in such a setup, as assigning non-conflict to all documents would still yield high accuracy.

Overall, InkSync performs worst likely due to its combination of both conflict detection and resolution in a single prediction. In contrast, both SemanticCommit and DropAllDocs benefit from task decomposition, achieving similar F1 scores. SemanticCommit's additional decomposition intro retrieval and conflict classification enables independent optimization contributing to the higher recall. This decomposition proves beneficial even when it is possible to fit all documents into the context window, as we observe worse conflict classification as the false positive rate (FPR) increases. Filtering down the chunks of information remains preferable.

We also ran evaluations of model latency and classification performance under varying false positive rates for the following LLMs by OpenAI: `GPT-4o`, `GPT-4o-mini`, and `o3-mini`. We selected `GPT-4o` for its slightly better performance, comparable latency to `GPT-4o-mini` and for being twice the speed of `o3-mini`. Additional details on FPR sensitivity and a comparison with `o3-mini` are provided in Appendix B of our Supplementary Material.

6 User study

To understand how users integrate new information in practice, we conducted a controlled within-subjects study with mixed methods, comparing SemanticCommit with a baseline interface. We had the following research questions:

- Which interface affordances do users prefer (use most often) when performing an integration of new information?
- How do users think through the process of integrating new information into an AI's existing memory store, with regards to detecting and resolving potential conflicts?
- Does SemanticCommit make users feel more in control of the integration process, over a more open-ended one?
- Does SemanticCommit's required manual review increase user workload compared to a more automated method?

We compared with a baseline to better understand: 1) any interface affordances our structured environment might miss, compared to an open-ended one; 2) how users might currently use popular AI-based tools to handle the process of integration, in the absence of targeted support. We chose OpenAI's ChatGPT Canvas as a baseline for five reasons: (i) it is a popular, commercially available tool, hence it is likely familiar to users; (ii) it provides a document editing view, where users can select text and ask GPT to rewrite it, or chat with an AI to make global edits; (iii) it employs a similar class of model (GPT-4o); (iv) it supports similar editing features as SemanticCommit like inline text selection, conflict highlighting, and a diff view, while adding free-form editing; and (v) similar interfaces like Anthropic Artifacts tended to rewrite the specification entirely, and did not offer Canvas's "diff" view to allow for a fair comparison.[9]

Participants. We recruited 12 participants (7 female, 5 male) through the mailing lists of two research universities and one multinational technology company. All the participants were familiar with GenAI tools. Ten participants used GenAI tools daily, and the other two at least weekly. ChatGPT was the most commonly used tool, alongside others, e.g., Gemini, MS Copilot, Claude, Perplexity, and Deepseek. Seven participants had previously used Canvas-like tools, and eight had used persisting memories (or preferences) with AI tools. Of these eight, four participants actively manage their memories either by adding, editing, or deleting them. Participants received a $25 Amazon Gift Card as compensation.

Tasks. We adapted two intent specifications from our evals: *Mars Game Design Document* and *Financial Advice AI Agent Memory*, as these tasks mapped to the two paradigmatic types covered in Sections 2 and 2.1 (design documents, and AI memory of the user). We ensured each list was 30 items long as our pilot studies suggested this was long enough that manual detection starts to become unwieldy (users need to scroll up and down the document), but short enough that participants could become familiar in a short period. For each task, participants were tasked with integrating three new pieces of information into the memory, one at a time ("sub-tasks"). We told participants to only change pieces of information that conflict with the new information, and that otherwise they were free to make additions, edits, and deletions as they saw fit. One of our tasks directly asks users to imagine they are an information management system that is managing memories about the user, in order to mimic how automated memory management systems will need to be conservative in changing information. More details on our tasks are provided in Supplementary Material.

Procedure. We hosted SemanticCommit online, allowing participants to access it via their web browser. For access to Canvas, we provided credentials for a ChatGPT account specifically created for the study to control for model and feature differences. With participant consent, we recorded audio and screen-casts, and participants were encouraged to think aloud. In each study session, the participant completed one of the two tasks each (each task containing 3 sub-tasks) using both the tools. Both the order of task assignment and tool assignment were counterbalanced and randomly assigned. Before each task, participants received a tutorial on the assigned tool and were given five minutes to explore it using a test document. We also provided a summary of the task document and time to read through it before starting. Each condition had a time limit of 15 minutes, after which the participant completed a post-task survey. After both tasks were completed, participants filled out a final survey to compare the two conditions. Finally, we conducted an informal interview about their experience.

Measurement and Analysis. For each task, we measured the success or failure of each sub-task the participant was required to perform. A sub-task was considered a failure if the participant was unable to complete it within the time limit. For condition

[9]We focused on AI-assisted conditions because our ultimate goal (and anticipation) is that AI will keep track of user intent, especially as the intent specification grows lengthy and unwieldy. Even within our limited evals, we encountered how time-consuming conflict detection can be: manually identifying conflicts for a single new piece of information could easily take 10 minutes, if one was being precise.

Figure 6: Participants' self-reported cognitive load and preference scores that directly compare the two conditions.

using SemanticCommit, we recorded all instances of *edits*, *check for conflicts*, *make change*, *local*, and *global resolution* actions using telemetry. In the post-task surveys, we collected self-reported NASA Task Load Index (TLX) scores, Likert-scale ratings for ease of use, and responses on how well the AI helped participants identify, understand, and resolve semantic conflicts. In the post-study survey, completed after both tasks, we recorded participants' self-reported tool preferences and a modified NASA TLX focused on comparing their experiences between the two tools. For qualitative analysis, the first author performed open coding on participant responses and audio transcripts to identify themes, which were used to interpret the qualitative results. To measure statistical significance, we used Mann–Whitney–Wilcoxon tests and report the p-values.

6.1 Findings

6.1.1 ***Preferred Workflow.*** Participants employed distinct workflows with each tool. We recount three characteristic workflows of SemanticCommit first, then compare to user behavior in Canvas.

Impact analysis first. Six participants (P1, P2, P4, P5, P9, P10) always began with *Check for Conflicts*, gaining insight on the impact of the change before integrating any changes. Participant P2 explained why they prefer *check for conflicts* by saying "*I really like the* check for conflicts *action – it still gives me control, and it feels collaborative instead of me kind of scrolling through the whole thing and trying to find it [referring to Canvas]. It highlights points of issues where I can plug this in.*" Participant P7 explained, "*I know where to make the edit, but I will use the global check so that I can find other places I might have to change*". All but one participant in this group proceeded exclusively with localized edits afterward.

Immediate changes with conflict review. Five participants (P3, P6, P7, P11, P12) always started the task with the *Make Change* feature to see the conflicts and the potential changes at once. They then followed up with local changes. P3 said "*This one has a lot of changes, so I'm going to use the global option. I'm just going to* make change*, and then figure out what to keep.*"

Skim to resolve false positives before proceeding. A method adopted within the two workflows, four participants (P3, P6, P9, P10) using SemanticCommit first quickly perused all the conflicts to resolve the false positives[10] and then proceeded to spend time resolving the *actual* conflicts.

In Canvas, users instead lean heavily on global rewrites. When using Canvas, eight participants (P1, P2, P4- P6, P10 - P12) predominantly utilized global prompts, instructing ChatGPT to perform edits throughout the entire document, while four participants preferred starting with global edits and subsequently performing local rewrites by selecting specific lines. As we recount below, this behavior intersected with frustrations from lack of control and the metacognitive demands [89] of prompting.

Workflow choice can depend on context. When asking participants how they pick between local vs. global resolution, they gave two major reasons—complexity of change and familiarity with the document. For example, P9 mentioned they would use global resolution techniques when they perceive the impact is higher—"*There is a lot of information here, it is much harder to go through it one by one. So I wanted to check for all the conflicts with the doc and then change it [collectively].*" The choice also depends on how familiar they are with the contents of the document. P12 said "*And I'm gonna go to [SemanticCommit] and put this as a global change. And I'm gonna say, first check for conflicts before making a change because I haven't read the complete document thoroughly.*"

6.1.2 ***Improved ability to catch semantic conflicts.*** Nine participants (P1 - P4, P7, P8, P10 - P12) explicitly stated that SemanticCommit was better at identifying conflicts compared to Canvas. In the post-study survey ranking, participants additionally report a higher level of *task success* with SemanticCommit compared to Canvas (μ=2.42; σ=1.5, where 1 indicates full preference for SemanticCommit), higher levels of success in *identifying semantic conflicts* (μ=2.08; σ=1.5) and in *understanding semantic conflicts* (μ=2.25; σ=1.95). As P4 noted "*It feels like you can identify inconsistencies easier in [SemanticCommit], which is what I liked a lot. So I favor that more. I'd feel I'd be a lot faster at getting work done.*"

This preference stemmed from two primary reasons. First, six participants (P2, P3, P4, P8, P11, P12) explicitly mentioned that when using SemanticCommit, the granularity of information and the red-colored highlights enabled easy conflict identification. P12 explained this in terms of context for the AI by saying "*I think the [SemanticCommit] tool is great in finding conflict, that's because it discretizes information, it's much more granular. It doesn't club all the context together.*" Second, except P2, all the other participants heavily relied on the *rationale* provided by SemanticCommit when understand why a conflict occurred. P8 explained this by saying "*With [SemanticCommit]... there is stronger explanation provided as to why that conflict is occurring.*"

Inconsistent conflict detection in Canvas leads to frustration and flailing. In contrast, nine participants (P1 - P3, P5 - P9, P11) noted that Canvas often missed conflicts or failed to understand the changes they wanted to make. Across 18 cases involving 10 participants (P1, P2, P3–P7, P9–P12), *Canvas failed to detect even a single conflict during the task.* In 9 of these cases, participants accepted the results without further checks; in the others, they had to either manually spot the issues or retry with more specific prompts. We highlight some of the observations below.

[10]Participants considered "false positives" as the conflicts flagged by the system that, in their subjective judgment, did not require meaningful intervention.

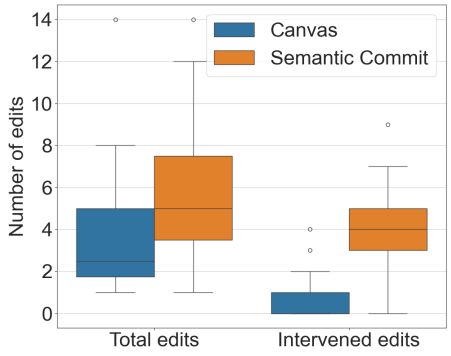


Figure 7: Participants using SemanticCommit made significantly more edits and intervened edits compared to Canvas.

In one instance, P5 had explicitly asked Canvas to find conflicts in the document. When the tool failed, the participant manually pointed out a conflict by quoting the text, and the AI model came up with a convoluted reason as to why it was not a conflict. P5 retorted by saying "*It is giving me an excuse.*" In a different task, P5 exclaimed "*Looks like it just added one statement, and there is no conflict. [5 seconds later] Oh wait! the GameBoy aesthetics is conflicting*"—catching a false negative manually in real time. In another instance, P9 prompted the Canvas tool three times to identify conflicts and make a change, but each attempt failed. Frustrated, they exclaimed, "*It didn't change it the way that I wanted. Maybe I'll delete this and do it myself and specify what I want to be changed*" before proceeding to manually make the change.

There were eight instances with six participants (P1, P2, P5, P7, P11, P12), where Canvas drastically changed the contents of the document either by replacing all the contents or by making heavy modifications. We then instructed the participants to restore to a previous version using version history.

6.1.3 ***Greater sense of control with SemanticCommit.*** A recurring theme among participants was the strong sense of control they felt while using SemanticCommit. Nine participants (P2, P3, P4, P6, P7, P8, P10, P11, P12) explicitly mentioned that SemanticCommit offered them more control over the integration process compared to Canvas. In the post-study survey ranking, participants additionally report a higher level of *control* with SemanticCommit compared to Canvas (μ=2.08; σ=1.36, where 1 indicates full preference for SemanticCommit), as well as a higher level of success in *resolving semantic conflicts* (μ=2.17; σ=1.34). This perception of control emerged due to several reasons mentioned below.

Granular insights into conflicts: Six participants (P2, P3, P4, P8, P11, P12) emphasized that the fine-grained presentation of information in SemanticCommit made it easier to identify, understand, and resolve conflicts—particularly for localized edits. The piece-by-piece breakdown gave users a clear sense of what was being altered and why. As P11 explained, "*you have some concept of a line—every element is aligned, so you probably have more granularity to control the elements that are being changed. That was really nice... I never had to worry that the entire document is going to get changed here and there.*" This precision allowed participants to maintain a stronger grasp over editing and focus their attention where it mattered.

Conflict reasoning encourages critical reflection: The tool's detailed breakdown of conflicts and its reasoning behind proposed changes encouraged users to think more critically about edits. P12 described how this led them to re-evaluate parts of the content they might have otherwise overlooked: "*So yeah, [SemanticCommit] improved the conflict finding even more... there were some parts in the document I would have ignored if I was doing it on my own. I wouldn't have considered some graphic design aspects of the game, but [SemanticCommit] provided its reason on why it has raised this as a conflict made me reconsider my decision. I like that part, because I would have easily ignored it, and that would have led to more iterations with more discussions.*"

Forced review enhanced sense of control over process: Another factor that reinforced a sense of control was the editing workflow itself. Unlike Canvas, which applied changes automatically, SemanticCommit required users to first review conflicts, make changes, and then manually click the resolve button to validate them. This structure helped participants feel like they were directing the process. As P10 observed, "*In [SemanticCommit] it was a step by step process to see the conflict, before making any changes whereas in [Canvas] there was no decision making on my behalf and it did the changes all by itself whether I agree with it or not.*" Similarly, P4 noted, "*Making changes [with SemanticCommit] was my favorite, because it walks you through every line, highlighting recommendations like revise, delete, change, add, or nothing.*"

This workflow—of reviewing conflicts, followed by local and/or global resolution—also could make the task feel more collaborative. Three participants (P1, P2, P5) described the process with SemanticCommit as collaborating with AI. P2 said "*With [SemanticCommit] you could ask it to look for conflicts. So you're sort of partnering like it would get the conflicts for you, and then you would move through them systematically... I felt like with [Canvas], you didn't have that middle ground. It was either make the change or don't.*"

Loss of control breeds insecurity. Due to Canvas not identifying conflicts and understanding instructions from participants, combined with sudden and drastic changes to the document, eight participants (P1-P6, P10, P11) explicitly mentioned they have doubts and insecurity when using Canvas to make any changes to a document. P2 said "*Using [Canvas] was really uncertain. You know, you just kind of felt like you're guessing, and you didn't know what was gonna happen.*" P6 also explained this by saying "*The downside of [Canvas] will be you just take it as it is, so you may not notice there's a part that should or shouldn't be changed. You may just skip it, pass it, and never notice the mistake the AI tool made.*"

Responsive UI with many local resolution options: Participants also appreciated the responsive nature of the interface during local resolution. As P11 described, "*The [SemanticCommit] tool I found quite intuitive, especially with the responsive nature where you put your mouse on it and there's a color code, and there's a green resolve button. The right-hand side gives you options to revise, reject, delete, edit, or suggest a new revision, etc. That is really good.*"

Ease of local reversibility: Like diff interfaces, participants also valued the ability to manually review changes and locally undo or dismiss them. P11 noted the friction in Canvas 's reversal process: "*With [Canvas], if you want to reject changes, then you probably have to undo and restore to the previous version, which seems a little*

cumbersome. It's not as simple as in [SemanticCommit] where you could accept a change or reject right there in that line."

Tradeoffs between control and efficiency: While many appreciated the explicit approval mechanisms in SemanticCommit, a few also noted potential tradeoffs. P3 acknowledged that the confirmation steps could feel excessive in low-conflict scenarios: "*I think sometimes it was overkill, if there were a pretty low number of conflicts detected. But otherwise, I think it was nice to confirm.*" P12 framed this as a tension between control and usability: "*I think it's important to do if you want finer control, but it really depends on the application you want to package it as. If you want better user experience, and you do not want them to spend more time, you would have to give them less control.*"

6.1.4 ***Perceived cognitive load.*** In the post-study survey, participants' preferences were measured using a 7-point Likert scale, where 1 indicated a strong preference for SemanticCommit and 7 indicated a strong preference for Canvas. Participants reported slightly higher levels of *mental demand* (μ=4.67; σ=1.56), *hurry* (μ=4.75; σ=1.14), and perceived *effort* (μ=4.5; σ=1.62) when using Canvas compared to SemanticCommit. They also reported slightly greater feelings of *annoyance* (μ=5; σ=1.2) with Canvas.

However, when comparing post-task questionnaires, we observed no statistically significant difference between conditions regarding mental demand, sense of hurriedness or frustration, effort exerted, or perception of success (all p-values are 0.45 and above). This null result was surprising to us, as we had expected *higher* workload in the SemanticCommit condition due to the increased demand as users manually click to resolve conflicts.

6.1.5 ***Task time and completion rates.*** On average, participants took 4 minutes and 7 seconds (σ= *117 seconds*) to complete tasks using the control tool compared to *5 minutes and 41 seconds* (σ= *123 seconds*) using the experimental tool. This difference is statistically significant ($p \approx 0.004$). It is important to note that task completion time does not capture task performance, as tasks encouraged participants to spend additional time holistically integrating document changes. We observed no significant difference in task completion rates between the two conditions. Four participants failed to complete one sub-task with Canvas compared to five participants with SemanticCommit, with all failures attributed to insufficient time.

6.1.6 ***Participants made significantly more edits when using SemanticCommit.*** Measuring participant engagement in controlled lab studies is challenging. Counting *edits*—with more edits typically indicating higher engagement—is useful, but AI tools can easily automate extensive editing, reducing reliability of metrics. To address this, in addition to studying number of edits overall (human- or AI-made), we also studied *intervened edits*—edits explicitly triggered by participants one at a time, whether manual or with AI. These metrics give a more comprehensive picture.

Participants using SemanticCommit demonstrated significantly higher engagement across both measures. They made an average of 5.83 edits (σ=3.21), compared to 3.5 edits with control (σ=2.85; $p \approx 0.001$). This contrast was even stronger for *intervened edits*, where participants using SemanticCommit averaged 4 edits (σ=1.94) per task, while participants using Canvas averaged just 0.65 (σ = 1; p<0.001; Figure 7). Finally, when using SemanticCommit, participants made an average of 2.93 *localized edits* per task, significantly (p<0.001) higher than an average of 0.28 localized edits per task when using Canvas. Participants extensively used the different kinds of local resolution strategies such as *revise*, *add*, and *delete* suggested by SemanticCommit. These differences highlight the participants' willingness to make more edits when using SemanticCommit. This also helps explain the higher average task completion time presented earlier—showing participants invested more time in understanding and making more deliberate changes.

6.1.7 ***Participant trust and over-reliance.*** Trust emerged as a complex and sometimes contradictory theme in how participants interacted with the AI tools. While many participants expressed skepticism toward AI-generated changes, their actual behavior revealed moments of over-reliance—particularly when changes appeared seamless or were not flagged as conflicts by the tool.

A majority of participants (P3, P4, P6, P7, P8, P10, P11, P12) explicitly stated that they did not trust the AI to make changes without their manual verification. As P10 firmly noted, "*No, I don't trust any AI blindly to make full and final changes to the result accurately. I always verify manually to spot any mistakes or misinterpretations by AI.*" This sentiment reflects a baseline level of caution we expected the participants to carry throughout the tasks. When comparing the two tools, six participants (P1, P2, P5, P6, P11, P12) explicitly reported greater trust in SemanticCommit over Canvas. They cited better contextual understanding and more transparency in the editing process as reasons for this preference. For example, P2 said, "*With [Canvas] I was very skeptical. I don't think I would trust it without doing a full read myself. With [SemanticCommit], I trusted it more. I felt like it seemed to understand the context better. But no matter the tool, I need to make sure that everything was good, so I would still read it over again.*"

Despite these widespread claims of skepticism, however, participants occasionally over-relied on both tools. As noted earlier, in nine instances where Canvas failed to identify any conflicts, participants accepted the output without further review. A similar pattern emerged with SemanticCommit: five participants skipped reviewing parts of the document that were not flagged as conflicting. This points to a potentially risky dependency on the AI and underscores our decision to improve recall at the expense of precision—if the model fails to detect a conflict (false negatives), users may miss critical issues simply because they trust the system's silence.

7 Discussion

7.1 Implications

7.1.1 ***AI Agent Interfaces Should Help Users Perform Impact Analysis.*** Our findings contribute to growing line of HCI research that emphasizes proactivity, presence, and just-in-time steering in AI agents acting on user's behalf [15, 52, 53, 67, 77, 82, 88]. The most surprising finding was participants' preference for performing *impact analysis*: finding conflicts first before making any edits. Instead of automatically applying changes and prompting users to verify afterward (like Canvas), this suggests AI agent systems should encourage users to first understand the impact of the change and only *then* choose to explicitly suggest and/or trigger changes. Our

findings indicate higher trust and satisfaction when users actively initiate changes, reducing uncertainty and increasing perceived control. Surprisingly, **the benefits from increased control seem to offset the cost of AI output validation**, as our results on perceived workload suggest. Not all users will use impact analysis in every context, but highlighting what aspects of an artifact will be considered and/or modified can help enhance user trust and control, especially in high-stakes situations.

This bears important implications for current AI agent interfaces, which tend to first let the AI make changes, and then have users *validate* them. For instance, in AI-powered programming IDEs like Cursor and Visual Studio, the agent makes changes across documents and then presents the revisions for human review. Instead, **our findings call upon designers of AI agent systems to provide affordances for *impact analysis*: helping users *foresee* the impact or location of AI changes, *before necessarily suggesting concrete changes***. This reflects the principle of *feedforward* [67, 93] in communication theory—"a needed prescription or plan for a feedback, to which the actual feedback may or may not confirm" [79]—where a communicator provides "the *context* of what one was planning to talk about" [64, p. 179-80] in order to "*pre-test* the impact of [its output]" on the listener [34, p. 65]. This returns control to the user and explicitly separates *retrieval* and *generation*, steps which are currently conflated in many agent interfaces. Such an affordance might also address a growing pain-point where unrelated data are deleted without approval.[11]

Note that impact analysis is not simply about pausing before enacting a change. It is also about weighing how extensive a change might be, the work required, and unintended side-effects. Users can use impact analysis to *back out* of an in-progress change, before the damage is done or they are overloaded by AI slop—an *AI resilient* [33, 35] affordance that helps users preemptively judge and respond to AI decisions. The reflective nature of impact analysis could also help users better understand potential conflicts, even inspire new ideas and areas for improvement.

7.1.2 ***Let the User Walk the Spectrum of Control.*** When designing mixed-initiative systems [43] where the users and AI collaborate, there is a trade-off between control (retaining it due to distrust in AI) and efficiency (completely delegating). SemanticCommit's affordances for adjustable autonomy [12], or blended agency [82], enabled the user to dynamically select their preferred balance between automation and manual oversight depending on the context, complexity of tasks, trust in the AI, or familiarity with the content, whereas users experienced loss of agency in the baseline condition. This suggests that AI agent interfaces should offer both highly controlled (step-by-step approvals like local resolution in SemanticCommit) and streamlined (global changes) workflows to adapt to varying user needs. Our participants appreciated detailed explanations about identified conflicts and recommended resolutions, which empowered them to make informed decisions. Transparency also appeared to reduce anxiety and frustration, promoting critical evaluation rather than passive acceptance.

7.1.3 ***Start Global, Then Accelerate Local Review.*** We implemented a range of elements into SemanticCommit, not knowing what users would prefer. We found that though users started globally, they preferred to then make local edits, and liberally used a range of local options—local steering, AI rewrites, etc—rather than global steering prompts and global resolution strategies. In the baseline Canvas condition, it was the exact opposite: users appeared resigned to global steering in chat and became frustrated by lack of granular control. This suggests **future interfaces for semantic conflict resolution should better support and accelerate local review,** rather than focusing on features for global steering after the initial interaction. The workflow of 4 participants to first dismiss false positives, and only *then* focus on handling conflicts, was also telling. Interfaces might explore explicitly separating stages of "double-checking" AI outputs versus resolving.

7.2 Limitations

Our comparison to ChatGPT Canvas yields an informative, "best-available" contrast, but Canvas differs from SemanticCommit in two directions: it lets users perform arbitrary free-form edits, yet it lacks SC's structured memory pane. These mismatches could inflate or deflate measured advantages. Our within-subjects study is also subject to demand characteristics [47]. Although we counterbalanced order, familiarity effects or social desirability bias could still surface. Seven participants also had previous Canvas experience, which might bias them to trust or prefer that interface. We therefore interpret subjective preference data cautiously and emphasize differences in observed workflows in our conclusions.

Another limitation is our treatment of AI memories as a list of unweighted facts. Emerging LLM frameworks can attach metadata such as model confidence, provenance, or temporal scope. Future iterations of SemanticCommit might consider color gradients computed from confidence deltas, yielding a continuous rather than three-band classification, and resolution suggestions could be ranked by expected reduction in global uncertainty.

7.3 Future Work and Connections

7.3.1 ***Interfaces and APIs for management of AI memory of user intent.*** We mentioned earlier that our intention is for SemanticCommit to become an API that helps users make "semantic commits": committing ideas and details to projects like we commit code, where the integration work is assisted by AI. Our UI was mainly a vehicle to see what users would do, were they given full control over the integration process. Left to their own devices to prompt chat models, our findings show that users are prone to miss conflicts or accept unwarranted rewrites of entire memory stores. Developers who utilize these simple one-shot prompting methods will be prone to similar failure modes. Tools like Claude Code provide users quick command-line directives to update memory, but simply append the directive to the end of the intent specification [2].

What would a more *assistive* command-line interface for memory updates look like? Could we automatically surface the conflicts that users care about, anticipating and correcting misalignments before they happen—potentially saving thousands of wasted inference calls? As AI agent systems grow in popularity, it becomes critical to explore interfaces and APIs that help users and developers alike

[11] There are many examples of this, from forums (https://news.ycombinator.com/item?id=43298275) to memes (https://x.com/daniel_nguyenx/status/1909184057755496571).

manage, *inspect*, and *update* AI memory of user intent in a manner that is *non-destructive*, *transparent*, and *controllable*.

The hard question is what to do when we do *not* have the luxury of a graphical UI—when intent integration is an API, part of a larger system. When and how to raise conflicts for user review? What rises to the level of "direct conflict" that must be addressed, versus an ambiguity that the AI could still proceed under? This goes back to our initial discussion on NLI and ambiguity, where human annotators had subjective differences in resolving conflicts [14, 49]—in many cases, these differences emerged from humans holding *different frames of reference*. To align conflict detection to specific users, we might consider two mechanisms—first, grounding acts like request for clarification [83, 85], triggered contextually. Vaithilingham et al. [92] suggest that the benefits of negotiation increase with the level of abstraction: AI agents should engage users in discussion for high-impact decisions, while avoiding it for low-impact ones. Second, a more passive mechanism might use memories to help model a particular user's classification of "conflict," aligning it over repeated interactions [84, 87]. Future research could investigate how to align conflict detectors to *specific* humans' ontological understanding of conflicts in their task domain.

7.3.2 ***Cognitive forcing functions to mitigate over-reliance.*** A line of research argues that to mitigate the risk of users becoming complacent or overly reliant on AI, systems should incorporate *cognitive forcing functions* [13, 22] —interface mechanisms that deliberately encourage active user involvement. In SemanticCommit, we do this by requiring explicit user approval when a conflict is detected or a change is made by the AI. Such mechanisms foster sustained cognitive engagement and reduce the likelihood of critical oversights resulting from blind trust in AI-generated outputs.

However, mitigation of over-reliance is not elimination. Our work reflects the tension between automation and agency [40, 82], embodied by our efforts to enhance recall to reduce false negatives. Drawing user attention to conflicts—even "ambiguous conflicts"—shows that users are liable to over-rely upon the AI to the extent of not checking any non-marked information. One further mitigation may be to mirror the kinds of divergences human annotators face when detecting conflicts [14, 49] by querying multiple LLMs in parallel and adopting a majority voting or ensembling scheme [91]. The "degree" of conflict might then correlate with consistency and number of votes, and divergences in LLM judges could be visualized.

7.3.3 ***Interfaces to support requirements-oriented prompting.*** Ma et al. [66] introduced a process for prompting AI agents that focuses on supporting users in creating a good initial set of requirements. They argue that in the age of "requirements-oriented" prompting, HCI will need to focus on training users to be good requirements engineers. Although not entirely focused on requirements lists, our interface can help users update requirements to reduce conflicts, inconsistencies, and ambiguities. Future studies might explicitly study the performance of an AI agent following the user's intentions after changes are made.

7.3.4 ***Semantic commits for long-form writing.*** One of the impetuses for this work was inspired by the challenges a coauthor faced when performing developmental editing for a long fiction novel. Developmental editing [72] assesses the overall content and structure of a document with regards to consistency, plot, and flow. Changed or removed scenes, even one-off conversations, could have impacts much later in a novel, and an author must keep all of this information in their head or manually reread to detect inconsistencies. A review by Zhao et al. [102] found that little HCI research focused on helping writers perform developmental editing. In the future, NLI-like AI-powered interfaces might help writers of long documents detect and resolve inconsistencies that emerge as a result of revisions. Much like *Portrayal* [42] shows writers birds-eye views of characters across a novel, might a similar interface help users to visualize "plot holes"? Our work suggests these semantic commit interfaces should heavily prioritize recall over precision, as a missed conflict across a 100k+ word novel may be catastrophic, compared to lightly reviewing false positives.

Acknowledgments

We would like to thank Professor Jin L. C. Guo for the helpful pointer to impact analysis literature in software engineering.

References

[1] LangChain AI. 2025. LangMem. https://langchain-ai.github.io/langmem/ Accessed: 2025-04-07.
[2] Anthropic. 2025. Claude 3.7 Sonnet and Claude Code. https://www.anthropic.com/news/claude-3-7-sonnet. Accessed: 2025-04-07.
[3] Grigoris Antoniou and Frank van Harmelen. 2004. *Web Ontology Language: OWL*. Springer Berlin Heidelberg, Berlin, Heidelberg, 67–92. doi:10.1007/978-3-540-24750-0_4
[4] Ian Arawjo. 2020. To Write Code: The Cultural Fabrication of Programming Notation and Practice. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) *(CHI '20)*. Association for Computing Machinery, New York, NY, USA, 1–15. doi:10.1145/3313831.3376731
[5] Ian Arawjo, Chelse Swoopes, Priyan Vaithilingam, Martin Wattenberg, and Elena L. Glassman. 2024. ChainForge: A Visual Toolkit for Prompt Engineering and LLM Hypothesis Testing. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) *(CHI '24)*. Association for Computing Machinery, New York, NY, USA, Article 304, 18 pages. doi:10.1145/3613904.3642016
[6] Robert S. Arnold. 1996. *Software Change Impact Analysis*. IEEE Computer Society Press, Washington, DC, USA.
[7] Chetan Arora, John Grundy, and Mohamed Abdelrazek. 2024. *Advancing Requirements Engineering Through Generative AI: Assessing the Role of LLMs*. Springer Nature Switzerland, Cham, 129–148. doi:10.1007/978-3-031-55642-5_6
[8] Gagan Bansal, Jennifer Wortman Vaughan, Saleema Amershi, Eric Horvitz, Adam Fourney, Hussein Mozannar, Victor Dibia, and Daniel S. Weld. 2024. Challenges in Human-Agent Communication. arXiv:2412.10380 [cs.HC] https://arxiv.org/abs/2412.10380
[9] Stafford Beer. 2002. What is cybernetics? *Kybernetes* 31, 2 (2002), 209–219.
[10] Thomas Berlage. 1994. A selective undo mechanism for graphical user interfaces based on command objects. *ACM Trans. Comput.-Hum. Interact.* 1, 3 (Sept. 1994), 269–294. doi:10.1145/196699.196721
[11] Guy A. Boy. 1997. Active design documents. In *Proceedings of the 2nd Conference on Designing Interactive Systems: Processes, Practices, Methods, and Techniques* (Amsterdam, The Netherlands) *(DIS '97)*. Association for Computing Machinery, New York, NY, USA, 31–36. doi:10.1145/263552.263572
[12] Jeffrey M. Bradshaw, Paul J. Feltovich, Hyuckchul Jung, Shriniwas Kulkarni, William Taysom, and Andrzej Uszok. 2004. Dimensions of Adjustable Autonomy and Mixed-Initiative Interaction. In *Agents and Computational Autonomy*, Matthias Nickles, Michael Rovatsos, and Gerhard Weiss (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 17–39.
[13] Zana Buçinca, Maja Barbara Malaya, and Krzysztof Z Gajos. 2021. To trust or to think: cognitive forcing functions can reduce overreliance on AI in AI-assisted decision-making. *Proceedings of the ACM on Human-computer Interaction* 5, CSCW1 (2021), 1–21.
[14] Sihao Chen, Chaitanya Malaviya, Alex Fabrikant, Hagai Taitelbaum, Tal Schuster, Senaka Buthpitiya, and Dan Roth. 2025. On Reference (In-)Determinacy in Natural Language Inference. In *Findings of the Association for Computational Linguistics: NAACL 2025*, Luis Chiruzzo, Alan Ritter, and Lu Wang (Eds.). Association for Computational Linguistics, Albuquerque, New Mexico, 8066–8078. doi:10.18653/v1/2025.findings-naacl.450

[15] Valerie Chen, Alan Zhu, Sebastian Zhao, Hussein Mozannar, David Sontag, and Ameet Talwalkar. 2025. Need Help? Designing Proactive AI Assistants for Programming. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems (CHI '25)*. Association for Computing Machinery, New York, NY, USA, Article 881, 18 pages. doi:10.1145/3706598.3714002

[16] Furui Cheng, Vilém Zouhar, Simran Arora, Mrinmaya Sachan, Hendrik Strobelt, and Mennatallah El-Assady. 2024. RELIC: Investigating Large Language Model Responses using Self-Consistency. 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 647, 18 pages. doi:10.1145/3613904.3641904

[17] John Joon Young Chung, Wooseok Kim, Kang Min Yoo, Hwaran Lee, Eytan Adar, and Minsuk Chang. 2022. TaleBrush: Sketching Stories with Generative Pretrained Language Models. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems* (New Orleans, LA, USA) *(CHI '22)*. Association for Computing Machinery, New York, NY, USA, Article 209, 19 pages. doi:10.1145/3491102.3501819

[18] Elizabeth Clark, Anne Spencer Ross, Chenhao Tan, Yangfeng Ji, and Noah A. Smith. 2018. Creative Writing with a Machine in the Loop: Case Studies on Slogans and Stories. In *Proceedings of the 23rd International Conference on Intelligent User Interfaces* (Tokyo, Japan) *(IUI '18)*. Association for Computing Machinery, New York, NY, USA, 329–340. doi:10.1145/3172944.3172983

[19] Herbert H. Clark and Susan E. Brennan. 1991. *Grounding in communication.* American Psychological Association, USA, 127–149. doi:10.1037/10096-006

[20] Richard Colby and Rebekah Shultz Colby. 2019. Game design documentation: Four perspectives from independent game studios. *Communication Design Quarterly Review* 7, 3 (2019), 5–15.

[21] Cursor Documentation Team. 2025. *Rules for AI.* Cursor. https://docs.cursor.com/context/rules-for-ai Accessed: 2025-04-08.

[22] Sander de Jong, Ville Paananen, Benjamin Tag, and Niels van Berkel. 2025. Cognitive Forcing for Better Decision-Making: Reducing Overreliance on AI Systems Through Partial Explanations. *Proc. ACM Hum.-Comput. Interact.* 9, 2, Article CSCW048 (May 2025), 30 pages. doi:10.1145/3710946

[23] Diego Dermeval, Jéssyka Vilela, Ig Ibert Bittencourt, Jaelson Castro, Seiji Isotani, Patrick Brito, and Alan Silva. 2016. Applications of ontologies in requirements engineering: a systematic review of the literature. *Requirements engineering* 21 (2016), 405–437.

[24] Bogdan Dit, Meghan Revelle, Malcom Gethers, and Denys Poshyvanyk. 2013. Feature location in source code: a taxonomy and survey. *Journal of software: Evolution and Process* 25, 1 (2013), 53–95.

[25] Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, Dasha Metropolitansky, Robert Osazuwa Ness, and Jonathan Larson. 2024. From Local to Global: A Graph RAG Approach to Query-Focused Summarization. arXiv:arXiv:2404.16130

[26] Hongfei Fan and Chengzheng Sun. 2012. Supporting semantic conflict prevention in real-time collaborative programming environments. *ACM SIGAPP Applied Computing Review* 12, 2 (2012), 39–52.

[27] Alessandro Fantechi, Stefania Gnesi, Lucia Passaro, and Laura Semini. 2023. Inconsistency Detection in Natural Language Requirements using ChatGPT: a Preliminary Evaluation. In *2023 IEEE 31st International Requirements Engineering Conference (RE)*. IEEE, Germany, 335–340. doi:10.1109/RE57278.2023.00045

[28] Mohamad Fazelnia, Viktoria Koscinski, Spencer Herzog, and Mehdi Mirakhorli. 2024. Lessons from the Use of Natural Language Inference (NLI) in Requirements Engineering Tasks . In *2024 IEEE 32nd International Requirements Engineering Conference (RE)*. IEEE Computer Society, Los Alamitos, CA, USA, 103–115. doi:10.1109/RE59067.2024.00020

[29] K. J. Kevin Feng, Kevin Pu, Matt Latzke, Tal August, Pao Siangliulue, Jonathan Bragg, Daniel S. Weld, Amy X. Zhang, and Joseph Chee Chang. 2024. Cocoa: Co-Planning and Co-Execution with AI Agents. arXiv:2412.10999 [cs.HC] https://arxiv.org/abs/2412.10999

[30] Mayara C. Figueiredo and Cleidson R. B. de Souza. 2012. Wolf: supporting impact analysis activities in distributed software development. In *Proceedings of the 5th International Workshop on Co-Operative and Human Aspects of Software Engineering (CHASE '12)*. IEEE Press, Zurich, Switzerland, 40–46.

[31] David Fox. 1986. Labyrinth. Internal game design document, Lucasfilm Ltd. Games Division. Game design document of an unpublished game Labyrinth by LucasArts in the 1908s related to the film of the same name. This document was accessed via Web Archive at the url: http://www.wilmunder.com/Arics_World/Games_files/.

[32] Marjan Ghazvininejad, Xing Shi, Jay Priyadarshi, and Kevin Knight. 2017. Hafez: an Interactive Poetry Generation System. In *Proceedings of ACL 2017, System Demonstrations*, Mohit Bansal and Heng Ji (Eds.). Association for Computational Linguistics, Vancouver, Canada, 43–48. https://aclanthology.org/P17-4008/

[33] Elena L. Glassman, Ziwei Gu, and Jonathan K. Kummerfeld. 2024. AI-Resilient Interfaces. arXiv:arXiv:2405.08447

[34] EM Griffin. 2006. *A first look at communication theory.* McGraw-hill, USA.

[35] Ziwei Gu, Ian Arawjo, Kenneth Li, Jonathan K. Kummerfeld, and Elena L. Glassman. 2024. An AI-Resilient Text Rendering Technique for Reading and Skimming Documents. 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 898, 22 pages. doi:10.1145/3613904.3642699

[36] Bernal Jiménez Gutiérrez, Yiheng Shu, Yu Gu, Michihiro Yasunaga, and Yu Su. 2025. HippoRAG: neurobiologically inspired long-term memory for large language models. In *Proceedings of the 38th International Conference on Neural Information Processing Systems* (Vancouver, BC, Canada) *(NIPS '24)*. Curran Associates Inc., Red Hook, NY, USA, Article 1902, 38 pages.

[37] ZHU Haibin. 2005. Conflict resolution with roles in a collaborative system. *International Journal of Intelligent Control and Systems* 10, 1 (2005), 11–20.

[38] J. Han. 1997. Supporting impact analysis and change propagation in software engineering environments. In *Proceedings Eighth IEEE International Workshop on Software Technology and Engineering Practice incorporating Computer Aided Software Engineering*. IEEE, Australia, 172–182. doi:10.1109/STEP.1997.615479

[39] Rex Hartson and Pardha Pyla. 2012. *The UX Book: Process and Guidelines for Ensuring a Quality User Experience* (1st ed.). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

[40] Jeffrey Heer. 2019. Agency plus automation: Designing artificial intelligence into interactive systems. *Proceedings of the National Academy of Sciences* 116, 6 (2019), 1844–1850.

[41] Jeffrey Heer, Joseph M Hellerstein, and Sean Kandel. 2015. Predictive Interaction for Data Transformation.. In *CIDR*. Citeseer, CIDR, Asilomar, California, 1–7.

[42] Md Naimul Hoque, Bhavya Ghai, Kari Kraus, and Niklas Elmqvist. 2023. Portrayal: Leveraging NLP and Visualization for Analyzing Fictional Characters. In *Proceedings of the 2023 ACM Designing Interactive Systems Conference* (Pittsburgh, PA, USA) *(DIS '23)*. Association for Computing Machinery, New York, NY, USA, 74–94. doi:10.1145/3563657.3596000

[43] Eric Horvitz. 1999. Principles of mixed-initiative user interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Pittsburgh, Pennsylvania, USA) *(CHI '99)*. Association for Computing Machinery, New York, NY, USA, 159–166. doi:10.1145/302979.303030

[44] Yuki Hou, Haruki Tamoto, and Homei Miyashita. 2024. "My agent understands me better": Integrating Dynamic Human-like Memory Recall and Consolidation in LLM-Based Agents. In *Extended Abstracts of the CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) *(CHI EA '24)*. Association for Computing Machinery, New York, NY, USA, Article 7, 7 pages. doi:10.1145/3613905.3650839

[45] Shang-Hsien Hsieh, Hsien-Tang Lin, Nai-Wen Chi, Kuang-Wu Chou, and Ken-Yu Lin. 2011. Enabling the development of base domain ontology through extraction of knowledge from engineering domain handbooks. *Advanced Engineering Informatics* 25, 2 (2011), 288–296. doi:10.1016/j.aei.2010.08.004 Information mining and retrieval in design.

[46] James Wayne Hunt and M Douglas MacIlroy. 1976. *An algorithm for differential file comparison.* Bell Laboratories Murray Hill, USA.

[47] Olga Iarygina, Kasper Hornbæk, and Aske Mottelson. 2025. Demand characteristics in human–computer experiments. *International Journal of Human-Computer Studies* 193 (2025), 103379. doi:10.1016/j.ijhcs.2024.103379

[48] Instructor Python API. 2025. Instructor Repository: .cursor/rules Directory. https://github.com/instructor-ai/instructor/tree/main/.cursor/rules Accessed: 2025-03-29.

[49] Nan-Jiang Jiang and Marie-Catherine de Marneffe. 2022. Investigating reasons for disagreement in natural language inference. *Transactions of the Association for Computational Linguistics* 10 (2022), 1357–1374.

[50] Per Jönsson and Mikael Lindvall. 2005. *Impact Analysis.* Springer Berlin Heidelberg, Berlin, Heidelberg, 117–142. doi:10.1007/3-540-28244-0_6

[51] Haruhiko Kaiya and Motoshi Saeki. 2006. Using Domain Ontology as Domain Knowledge for Requirements Elicitation. In *14th IEEE International Requirements Engineering Conference (RE'06)*. IEEE, St. Paul, MN, USA, 189–198. doi:10.1109/RE.2006.72

[52] Majeed Kazemitabaar, Oliver Huang, Sangho Suh, Austin Z Henley, and Tovi Grossman. 2025. Exploring the Design Space of Cognitive Engagement Techniques with AI-Generated Code for Enhanced Learning. In *Proceedings of the 30th International Conference on Intelligent User Interfaces (IUI '25)*. Association for Computing Machinery, New York, NY, USA, 695–714. doi:10.1145/3708359.3712104

[53] Majeed Kazemitabaar, Jack Williams, Ian Drosos, Tovi Grossman, Austin Zachary Henley, Carina Negreanu, and Advait Sarkar. 2024. Improving Steering and Verification in AI-Assisted Data Analysis with Interactive Task Decomposition. In *Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology* (Pittsburgh, PA, USA) *(UIST '24)*. Association for Computing Machinery, New York, NY, USA, Article 92, 19 pages. doi:10.1145/3654777.3676345

[54] Tae Kim. 2018. Warren Buffett says bitcoin is 'probably rat poison squared'. https://www.cnbc.com/2018/05/05/warren-buffett-says-bitcoin-is-probably-rat-poison-squared.html Updated: 2018-05-06.

[55] Clemens Nylandsted Klokmose, James R. Eagan, and Peter van Hardenberg. 2024. MyWebstrates: Webstrates as Local-first Software. In *Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology* (Pittsburgh,

PA, USA) *(UIST '24)*. Association for Computing Machinery, New York, NY, USA, Article 42, 12 pages. doi:10.1145/3654777.3676445

[56] Philippe Laban, Jesse Vig, Marti Hearst, Caiming Xiong, and Chien-Sheng Wu. 2024. Beyond the Chat: Executable and Verifiable Text-Editing with LLMs. In *Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology* (Pittsburgh, PA, USA) *(UIST '24)*. Association for Computing Machinery, New York, NY, USA, Article 20, 23 pages. doi:10.1145/3654777.3676419

[57] Nancy G. Leveson. 2000. System Safety in Computer-Controlled Automotive Systems. *SAE Transactions* 109 (2000), 287–294. http://www.jstor.org/stable/44699139

[58] Jingyi Li, Eric Rawn, Jacob Ritchie, Jasper Tran O'Leary, and Sean Follmer. 2023. Beyond the Artifact: Power as a Lens for Creativity Support Tools. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology* (San Francisco, CA, USA) *(UIST '23)*. Association for Computing Machinery, New York, NY, USA, Article 47, 15 pages. doi:10.1145/3586183.3606831

[59] Brian Y. Lim, Anind K. Dey, and Daniel Avrahami. 2009. Why and why not explanations improve the intelligibility of context-aware intelligent systems. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Boston, MA, USA) *(CHI '09)*. Association for Computing Machinery, New York, NY, USA, 2119–2128. doi:10.1145/1518701.1519023

[60] Susan Lin, Jeremy Warner, J.D. Zamfirescu-Pereira, Matthew G Lee, Sauhard Jain, Shanqing Cai, Piyawat Lertvittayakumjorn, Michael Xuelin Huang, Shumin Zhai, Bjoern Hartmann, and Can Liu. 2024. Rambler: Supporting Writing With Speech via LLM-Assisted Gist Manipulation. 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 1043, 19 pages. doi:10.1145/3613904.3642217

[61] Geoffrey Litt, Sarah Lim, Martin Kleppmann, and Peter Van Hardenberg. 2022. Peritext: A crdt for collaborative rich text editing. *Proceedings of the ACM on Human-Computer Interaction* 6, CSCW2 (2022), 1–36.

[62] Michael Xieyang Liu, Tongshuang Wu, Tianying Chen, Franklin Mingzhe Li, Aniket Kittur, and Brad A Myers. 2024. Selenite: Scaffolding Online Sensemaking with Comprehensive Overviews Elicited from 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 837, 26 pages. doi:10.1145/3613904.3642149

[63] Diana Loeffler, Anne Hess, Andreas Maier, Joern Hurtienne, and Hartmut Schmitt. 2013. Developing intuitive user interfaces by integrating users' mental models into requirements engineering. In *Proceedings of the 27th International BCS Human Computer Interaction Conference* (London, UK) *(BCS-HCI '13)*. BCS Learning & Development Ltd., Swindon, GBR, Article 15, 10 pages.

[64] Robert K Logan. 2015. Feedforward, IA Richards, Cybernetics and Marshall McLuhan. *Systema: Connecting Catter, Life, Culture and Technology* 3, 1 (2015), 177–185.

[65] Sebastian Lubos, Alexander Felfernig, {Thi Ngoc Trang} Tran, Damian Garber, Merfat {El Mansi}, {Seda Polat} Erdeniz, and {Viet Man} Le. 2024. Leveraging LLMs for the Quality Assurance of Software Requirements. In *Proceedings - 32nd IEEE International Requirements Engineering Conference, RE 2024*, Grischa Liebel, Irit Hadar, and Paola Spoletini (Eds.). IEEE Computer Society, United States, 389–397. doi:10.1109/RE59067.2024.00046 Publisher Copyright: © 2024 IEEE.; 32nd IEEE International Requirements Engineering Conference : RE 2024 ; Conference date: 24-06-2024 Through 28-06-2024.

[66] Qianou Ma, Weirui Peng, Chenyang Yang, Hua Shen, Kenneth Koedinger, and Tongshuang Wu. 2024. What Should We Engineer in Prompts? Training Humans in Requirement-Driven LLM Use. arXiv:2409.08775 [cs.HC] https://arxiv.org/abs/2409.08775

[67] Bryan Min and Haijun Xia. 2025. Feedforward in Generative AI: Opportunities for a Design Space. arXiv:arXiv:2502.14229

[68] 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 *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems* (Hamburg, Germany) *(CHI '23)*. Association for Computing Machinery, New York, NY, USA, Article 355, 34 pages. doi:10.1145/3544548.3581225

[69] Robert B Musburger. 2017. *Animation production: documentation and organization*. CRC Press, Boca Raton, FL.

[70] Nicholas Nelson, Caius Brindescu, Shane McKee, Anita Sarma, and Danny Dig. 2019. The life-cycle of merge conflicts: processes, barriers, and strategies. *Empirical Software Engineering* 24 (2019), 2863–2906.

[71] Jakob Nielsen. 1994. Enhancing the explanatory power of usability heuristics. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Boston, Massachusetts, USA) *(CHI '94)*. Association for Computing Machinery, New York, NY, USA, 152–158. doi:10.1145/191666.191729

[72] Scott Norton. 2009. *Developmental editing: a handbook for freelancers, authors, and publishers*. University of Chicago Press, Chicago, USA.

[73] OpenAI. 2024. Memory and new controls for ChatGPT. https://openai.com/index/memory-and-new-controls-for-chatgpt/ Accessed: 2025-04-07.

[74] 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 *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology* (San Francisco, CA, USA) *(UIST '23)*. Association for Computing Machinery, New York, NY, USA, Article 2, 22 pages. doi:10.1145/3586183.3606763

[75] S.H. Phatak and B.R. Badrinath. 1999. Conflict resolution and reconciliation in disconnected databases. In *Proceedings. Tenth International Workshop on Database and Expert Systems Applications. DEXA 99*. IEEE, Florence, Italy, 76–81. doi:10.1109/DEXA.1999.795148

[76] Git Project. 2025. Git: Fast Version Control System. https://git-scm.com/ Accessed: 24 Jan. 2025.

[77] Kevin Pu, Daniel Lazaro, Ian Arawjo, Haijun Xia, Ziang Xiao, Tovi Grossman, and Yan Chen. 2025. Assistance or Disruption? Exploring and Evaluating the Design and Trade-offs of Proactive AI Programming Support. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems (CHI '25)*. Association for Computing Machinery, New York, NY, USA, Article 152, 21 pages. doi:10.1145/3706598.3713357

[78] Reddit. 2024. The hidden Claude system prompt (on the Artefacts system, new response styles, thinking tags, and more...). https://www.reddit.com/r/ClaudeAI/comments/1hb3evv/the_hidden_claude_system_prompt_on_the_artefacts/ Accessed: 2025-04-02.

[79] Ivor A Richards. 1968. The secret of feedforward. *Saturday Review* 3 (1968), 14–17.

[80] Diana Robinson, Christian Cabrera, Andrew D. Gordon, Neil D. Lawrence, and Lars Mennen. 2024. Requirements are All You Need: The Final Frontier for End-User Software Engineering. arXiv:arXiv:2405.13708

[81] Advait Sarkar. 2024. Intention Is All You Need. arXiv:arXiv:2410.18851

[82] Arvind Satyanarayan and Graham M. Jones. 2024. Intelligence as Agency: Evaluating the Capacity of Generative AI to Empower or Constrain Human Action. https://mit-genai.pubpub.org/pub/94y6e0f8.

[83] Omar Shaikh, Kristina Gligoric, Ashna Khetan, Matthias Gerstgrasser, Diyi Yang, and Dan Jurafsky. 2024. Grounding Gaps in Language Model Generations. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, Kevin Duh, Helena Gomez, and Steven Bethard (Eds.). Association for Computational Linguistics, Mexico City, Mexico, 6279–6296. doi:10.18653/v1/2024.naacl-long.348

[84] Omar Shaikh, Michelle Lam, Joey Hejna, Yijia Shao, Michael Bernstein, and Diyi Yang. 2024. Show, Don't Tell: Aligning Language Models with Demonstrated Feedback. arXiv:2406.00888 [cs.CL]

[85] Omar Shaikh, Hussein Mozannar, Gagan Bansal, Adam Fourney, and Eric Horvitz. 2025. Navigating Rifts in Human-LLM Grounding: Study and Benchmark. arXiv:2503.13975 [cs.CL] https://arxiv.org/abs/2503.13975

[86] Omar Shaikh, Shardul Sapkota, Shan Rizvi, Eric Horvitz, Joon Sung Park, Diyi Yang, and Michael S. Bernstein. 2025. Creating General User Models from Computer Use. arXiv:arXiv:2505.10831

[87] Shreya Shankar, J.D. Zamfirescu-Pereira, Bjoern Hartmann, Aditya Parameswaran, and Ian Arawjo. 2024. Who Validates the Validators? Aligning LLM-Assisted Evaluation of LLM Outputs with Human Preferences. In *Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology* (Pittsburgh, PA, USA) *(UIST '24)*. Association for Computing Machinery, New York, NY, USA, Article 131, 14 pages. doi:10.1145/3654777.3676450

[88] Yijia Shao, Vinay Samuel, Yucheng Jiang, John Yang, and Diyi Yang. 2024. Collaborative Gym: A Framework for Enabling and Evaluating Human-Agent Collaboration.

[89] Lev Tankelevitch, Viktor Kewenig, Auste Simkute, Ava Elizabeth Scott, Advait Sarkar, Abigail Sellen, and Sean Rintel. 2024. The Metacognitive Demands and Opportunities of Generative AI. 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 680, 24 pages. doi:10.1145/3613904.3642902

[90] Valerio Terragni, Annie Vella, Partha Roop, and Kelly Blincoe. 2025. The Future of AI-Driven Software Engineering. *ACM Trans. Softw. Eng. Methodol.* 34, 5, Article 120 (May 2025), 20 pages. doi:10.1145/3715003

[91] Yu-Min Tseng, Yu-Chao Huang, Teng-Yun Hsiao, Wei-Lin Chen, Chao-Wei Huang, Yu Meng, and Yun-Nung Chen. 2024. Two Tales of Persona in LLMs: A Survey of Role-Playing and Personalization. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (Eds.). Association for Computational Linguistics, Miami, Florida, USA, 16612–16631. doi:10.18653/v1/2024.findings-emnlp.969

[92] Priyan Vaithilingam, Ian Arawjo, and Elena L. Glassman. 2024. Imagining a Future of Designing with AI: Dynamic Grounding, Constructive Negotiation, and Sustainable Motivation. In *Proceedings of the 2024 ACM Designing Interactive Systems Conference* (Copenhagen, Denmark) *(DIS '24)*. Association for Computing Machinery, New York, NY, USA, 289–300. doi:10.1145/3643834.3661525

[93] Jo Vermeulen, Kris Luyten, Elise van den Hoven, and Karin Coninx. 2013. Crossing the bridge over Norman's Gulf of Execution: revealing feedforward's true identity. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Paris, France) *(CHI '13)*. Association for Computing Machinery, New York, NY, USA, 1931–1940. doi:10.1145/2470654.2466255

[94] Thiemo Wambsganss, Christina Niklaus, Matthias Cetto, Matthias Söllner, Siegfried Handschuh, and Jan Marco Leimeister. 2020. AL: An Adaptive Learning Support System for Argumentation Skills. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) *(CHI '20)*. Association for Computing Machinery, New York, NY, USA, 1–14. doi:10.1145/3313831.3376732

[95] Xinru Wang, Hannah Kim, Sajjadur Rahman, Kushan Mitra, and Zhengjie Miao. 2024. Human-LLM Collaborative Annotation Through Effective Verification of LLM Labels. 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 303, 21 pages. doi:10.1145/3613904.3641960

[96] Cat Wu. 2025. Press # to instruct Claude Code to add a memory. Then, type your memory and hit Enter to add it to the CLAUDE.md file. https://x.com/_catwu/status/1904941906867413054 Tweet by the Product Designer of Claude Code..

[97] Ryan Yen and Jian Zhao. 2024. Memolet: Reifying the Reuse of User-AI Conversational Memories. In *Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology* (Pittsburgh, PA, USA) *(UIST '24)*. Association for Computing Machinery, New York, NY, USA, Article 58, 22 pages. doi:10.1145/3654777.3676388

[98] Young Seok Yoon and Brad A. Myers. 2015. Supporting selective undo in a code editor. In *Proceedings of the 37th International Conference on Software Engineering - Volume 1 (ICSE '15)*. IEEE Press, Florence, Italy, 223–233.

[99] Ann Yuan, Andy Coenen, Emily Reif, and Daphne Ippolito. 2022. Wordcraft: Story Writing With Large Language Models. In *Proceedings of the 27th International Conference on Intelligent User Interfaces* (Helsinki, Finland) *(IUI '22)*. Association for Computing Machinery, New York, NY, USA, 841–852. doi:10.1145/3490099.3511105

[100] J.D. Zamfirescu-Pereira, Eunice Jun, Michael Terry, Qian Yang, and Bjoern Hartmann. 2025. Beyond Code Generation: LLM-supported Exploration of the Program Design Space. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems (CHI '25)*. Association for Computing Machinery, New York, NY, USA, Article 153, 17 pages. doi:10.1145/3706598.3714154

[101] 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* (San Francisco, CA, USA) *(UIST '23)*. Association for Computing Machinery, New York, NY, USA, Article 5, 30 pages. doi:10.1145/3586183.3606800

[102] Zixin Zhao, Damien Masson, Young-Ho Kim, Gerald Penn, and Fanny Chevalier. 2025. Making the Write Connections: Linking Writing Support Tools with Writer Needs. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems (CHI '25)*. Association for Computing Machinery, New York, NY, USA, Article 1216, 21 pages. doi:10.1145/3706598.3713161