Université de Montréal

Designing Interaction and Management for AI Agent Memory: Design Space, Metaphors, and Interactive Conflict Management

par
Munyeong Kim

Département d'informatique et de recherche opérationnelle
Faculté des arts et des sciences

Mémoire présenté à la Faculté des études supérieures
en vue de l'obtention du grade de Maître ès sciences (M.Sc.)
en informatique
spécialité intelligence artificielle

juin, 2026

Université de Montréal
Faculté des études supérieures

Ce mémoire intitulé:

Designing Interaction and Management for AI Agent Memory: Design Space, Metaphors, and Interactive Conflict Management

présenté par:

Munyeong Kim

a été évalué par un jury composé des personnes suivantes:

Benoit Baudry,	président-rapporteur
Ian Arawjo,	directeur de recherche
Damien Masson,	membre du jury

RÉSUMÉ

À mesure que l'importance de la personnalisation et de la spécialisation dans les agents d'IA continue de croître, la couche de mémoire qui oriente leur comportement devient un enjeu de plus en plus important pour la conception de systèmes d'IA interactifs. Ce mémoire aborde cette question à travers deux articles qui examinent des aspects complémentaires de l'interaction avec la mémoire d'IA et les outils conçus pour la soutenir. Le premier article, *Metaphors for Memory: Charting a Design Space of AI Memory Tools and Interfaces*, examine les outils et interfaces de mémoire d'IA afin d'élaborer un espace de conception des patrons, architectures, opérations, métaphores dominantes et lacunes, puis élargit cet espace au moyen d'une démarche de conception métaphorique générative. Le second article, *Semantic Commit: Helping Users Update Intent Specifications for AI Memory at Scale*, présente une interface mixte qui aide les utilisateurs à intégrer de nouvelles intentions dans des spécifications de mémoire en langage naturel tout en maintenant leur cohérence, grâce à la détection et à la résolution de conflits sémantiques potentiels avec l'appui de grands modèles de langage. En m'appuyant sur ces deux articles, j'ouvre une discussion plus large sur la manière dont la mémoire des agents d'IA devrait être conçue, mise à jour et gouvernée par l'interaction avec l'utilisateur, ainsi que sur les difficultés propres à ces processus et sur les pistes pour y répondre.

mots clés interaction humain-IA, gestion de la mémoire des agents, espace de conception, spécification d'intention, ancrage humain-IA, modèles de langage.

ABSTRACT

As the importance of personalization and specialization in AI agents continues to grow, the memory layer that shapes their behavior has become an increasingly important concern for designing interactive AI systems. This mémoire addresses this issue through two articles that explore complementary aspects of interaction with AI memory and the tools designed to support such interaction. The first article, *Metaphors for Memory: Charting a Design Space of AI Memory Tools and Interfaces*, surveys tools and interfaces around AI memory to chart a design space of common patterns, architectures, operations, dominant metaphors, and gaps, and then expands that space through generative metaphorical design. The second article, *Semantic Commit: Helping Users Update Intent Specifications for AI Memory at Scale*, introduces a mixed-initiative interface that helps users integrate new intent into natural-language memory specifications while maintaining consistency by detecting and resolving potential semantic conflicts with LLM support. Drawing on these two articles, I open further discussion about how AI agent memory should be designed, updated, and governed through user interaction, as well as the challenges involved in supporting such processes and how they might be addressed.

Keywords: human-AI interaction, agent memory management, design space, intent specification, human-AI grounding, language models

CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ACM	Association for Computing Machinery
AI	Artificial intelligence
API	Application programming interface
BFS	Breadth-first search
BM25	Best Matching 25
CLI	Command-line interface
CoT	Chain-of-Thought
DFS	Depth-first search
DIS	Designing Interactive Systems
FPR	False positive rate
GDD	Game design document
GPT	Generative Pre-trained Transformer
HCI	Human-computer interaction
IDE	Integrated development environment
IR	Information retrieval
JSON	JavaScript Object Notation
KG	Knowledge graph
LLM	Large language model
LSTM	Long short-term memory
NER	Named entity recognition
NLI	Natural language inference
NLP	Natural language processing
OCR	Optical character recognition
RAG	Retrieval-augmented generation
SE	Software engineering
TLX	NASA Task Load Index
UI	User interface

UIST	User Interface Software and Technology
UX	User experience
VCS	Version control system
YAML	YAML Ain’t Markup Language

CHAPTER 1

INTRODUCTION

Whatever we imagine as an ideal form of artificial intelligence, and whatever kind of relationship we imagine having with such an agent, we need some mechanism through which the agent can retain information gained through interaction, reorganize it as new information emerges, and reuse it when relevant. I define this retained and reusable information as AI memory, and the systems that support its storage, retrieval, revision, and use as AI memory systems.

Organizing such memory systems matters for human–AI interaction because human instructions are always context-dependent. An expression given by a user to an agent depends on the surrounding context, and can therefore be interpreted appropriately only in relation to that context. Without such context, no agent can reliably infer what the user intends or guarantee a correct interpretation. To act appropriately, an agent must therefore build and maintain a usable representation of the user's context through interaction. Its memory system must preserve the relevant contextual information that emerges from this interaction so that it can be reused in future interpretation and action.

AI memory can be implemented in different ways. Some forms of memory are implicit, encoded in model parameters or internal representations of the model itself, and shape how outputs are generated from a given input. Other forms of memory are explicit, represented outside the model parameters as information that conditions the agent's behavior, such as stored preferences, project rules, system prompts, context windows, documents, databases, workspaces, or files. The concept of AI memory is therefore not yet stable across communities. It may refer to parametric knowledge, retrieval mechanisms, persistent user preferences, external knowledge bases, conversational histories, or human-readable specifications that guide agent behavior.

While both implicit and explicit forms matter for the success of AI agent systems, this mémoire focuses particularly on the value of explicit memory from a human–AI interaction perspective. Explicit memory can be read, inspected, revised, and governed by

users. As agents are asked to support more personalized, long-term, and consequential tasks, users need confidence that the agent will act in relation to the relevant memories, constraints, and commitments that have been established as common ground between the user and the agent [19, 37, 38]. This remains important even if an implicit system could produce similar behavior, because explicit memory provides a shared and maintainable reference point that users can inspect, revise, and govern during interaction. In this sense, explicit memory can function in human–AI interaction like written requirements, rules, or agreements in human collaboration, serving as a record that the agent is expected to consult, follow, and update as interaction unfolds.

Yet from a human–AI interaction perspective, we still lack a systematic understanding of how AI memory should be designed, managed, and made available through interfaces. As the reasoning capabilities of AI models improve, agents are being asked to support a wider range of tasks, and the space of agentic systems, memory tools, interfaces, and interaction patterns is developing quickly. However, the concept of AI memory itself remains unsettled, and the interfaces needed to support these emerging forms of memory have not yet been sufficiently explored. Many existing systems still handle memory through familiar interaction forms, such as chats, text editors, IDE panels, documents, or rule lists, without a clear account of what kinds of memory are being represented, how they are represented, what operations users may need to perform on them, or how those memory structures shape interaction.

These questions become especially visible as AI memory grows longer, more distributed, and messier over time, while much of the stored information may remain ambiguous, context-dependent, or subject to change. This management problem can become even more complex when interaction moves beyond a one-to-one relation between a single user and a single agent, since multiple users, agents, contexts, and goals can make the dynamics of memory use and maintenance more difficult to understand. Even in such settings, we need systems that give users enough control and governance to determine which memories are still relevant, whether new information conflicts with prior commitments, how a change might affect future agent behavior, and when the agent should ask for clarification. From an HCI perspective, designing methods and inter-

faces that support the inspection, revision, and governance of AI memory is therefore an important research challenge.

This mémoire addresses these gaps through two articles produced during my master's studies, each studying AI memory through a different scope and methodological approach. The first article, *Metaphors for Memory: Charting a Design Space of AI Memory Tools and Interfaces*, addresses the broader conceptual and design-space gap by surveying existing AI memory tools and interfaces. It charts a design space of common patterns, architectures, operations, and interfaces, identifies dominant metaphors and gaps in the space, and then applies generative metaphorical design [117, 160] to expand possible ways of imagining AI memory through less-dominant metaphors such as software version control, Zettelkasten, requirements, personal diaries, community archives, cultural probes, and science fiction. The second article, *Semantic Commit: Helping Users Update Intent Specifications for AI Memory at Scale*, addresses a more practical management question by focusing on one specific form of explicit AI memory: intent specifications, or natural language documents such as AI memory lists, Cursor Rules [45], and game design documents. It introduces SemanticCommit, a mixed-initiative interface that helps users integrate new intent into such memory while maintaining consistency, detecting potential semantic conflicts through a knowledge graph-based retrieval-augmented generation pipeline and assisting users in resolving them with LLM support.

Methodologically, this mémoire demonstrates the use of seven different research and design approaches to examine AI memory from multiple angles: (1) literature and tool survey, (2) design-space analysis, (3) generative metaphorical design, (4) conceptual framing, (5) system-building, (6) technical evaluation, and (7) empirical user study. The first article combines a literature and tool survey, design-space analysis, and generative metaphorical design to map current architectural and operational patterns, identify dominant metaphors and gaps, and expand possible directions for future AI memory interfaces. The second article combines conceptual framing, system-building, technical evaluation, and a within-subjects user study to address semantic conflicts among memory entries as an important practical issue in managing AI memory over time, while introducing intent specifications as one form of explicit AI memory. This methodologi-

cal range provides multiple perspectives for examining AI memory as a research topic, while also allowing the mémoire to move across two complementary scopes of inquiry, from a broad mapping of the AI memory landscape to a focused investigation of semantic conflict as a practical issue in managing AI memory over time.

On this basis, this mémoire contributes to human–computer interaction discussions on how AI agent memory can be designed, updated, and governed through interaction. This contribution is timely because AI memory is still at an early stage, yet it is already generating a rapidly growing ecosystem of tools, systems, and research questions. Even during the period in which this mémoire was being revised, new memory-related agent systems and interface ideas continued to appear, making it increasingly important to understand the shape of this emerging landscape. By mapping existing approaches, identifying gaps and design opportunities, and examining one practical issue that arises when memory is updated over time, this mémoire aims to provide a reference point for future work on AI memory interfaces in HCI.

1.0.1 Article status and author contributions

Both articles included in this mémoire have been accepted to peer-reviewed ACM conferences. The first article, *Metaphors for Memory: Charting a Design Space of AI Memory Tools and Interfaces*, has been accepted to the ACM Designing Interactive Systems Conference (DIS) 2026, and I am its first author. The second article, *Semantic Commit: Helping Users Update Intent Specifications for AI Memory at Scale*, was presented at the ACM Symposium on User Interface Software and Technology (UIST) 2025, and I am its second author.

Since this mémoire is based on two co-authored articles, it is important to clarify my role in each article and to acknowledge the contributions of the collaborators who made each project possible.

In the first article, I served as first author and led the project overall. My contributions included developing the conceptual framing, conducting the literature and tool survey, leading the design-space analysis, carrying out the generative metaphorical design exploration, and writing the article. Professor Michalis Famelis contributed to conceptual

discussions, especially around the software-oriented and requirements-oriented aspects of AI memory, and participated in the generative metaphorical design process. Professor Ian Arawjo served as the corresponding author and supervisor for the article, providing guidance throughout the framing, analysis, design exploration, and writing process.

In the second article, my role was more focused. I contributed to the conceptualization of memory conflict, implemented the frontend of the React prototype used in the user study, created and validated the benchmark datasets for the technical evaluation together with Frida-Cecilia Acosta-Parenteau, and supported the writing of the Motivation and Related Work sections. Dr. Priyan Vaithilingam led the project overall, including the main system design, implementation, study design, evaluation, and writing. Frida-Cecilia Acosta-Parenteau worked with me to create and validate the benchmark datasets for the technical evaluation and contributed to preparing and refining the study materials. Dr. Daniel Lee contributed technical and conceptual guidance on semantic analysis and conflict detection. Professor Amine Mhedhbi ran the benchmark evaluation, wrote the technical evaluation section, and contributed technical guidance on semantic modeling and database-related aspects of the system. Professor Elena L. Glassman provided supervision and guidance on the project's framing and development. Professor Ian Arawjo served as a corresponding author, supporting coordination, implementation, and writing. Working on this collaboration helped me recognize and begin to address the interaction problems that arise in managing AI memory, an experience that later motivated the first article's broader effort to map the AI memory landscape, identify its gaps, and explore alternative metaphors for future memory interfaces.

Note. While this mémoire itself is solely my own work, I retain the original use of "we" for the included articles to reflect their original wording and coauthorship.

CHAPTER 2

METAPHORS FOR MEMORY: CHARTING A DESIGN SPACE OF AI MEMORY TOOLS AND INTERFACES

Abstract

AI memory is becoming central to AI systems that aim to support personal and professional work. Yet, designing interfaces for AI memory is not well understood. How should we design for user interaction with AI memories—for instance, what kinds of operations might users want to perform on memory, either now or in the future? We survey the design of tools and interfaces around "AI memory" to chart a design space of common design patterns, architectures, and operations, and identify dominant metaphors and gaps in the space. Then, we apply generative metaphorical design to expand the design space for AI memory, exploring less-dominant metaphors of software version control, Zettelkasten, requirements, personal diaries, community archives, cultural probes, and science fiction. Our work offers rich opportunities for gaps and emergent needs that future interfaces for AI memory might address.

Article Information

Authors. Munyeong Kim, Michalis Famelis, and Ian Arawjo.

Venue. 2026 ACM SIGCHI Conference on Designing Interactive Systems (DIS '26), Full Papers track.

Current status. Accepted.

Personal contribution. I led the conceptual framing of the project, the review and analysis of existing AI memory tools and interfaces, the generative metaphorical design exploration, and the writing of the manuscript.

Co-author contributions.

- Professor Michalis Famelis contributed to conceptual discussions, especially around the software-oriented and requirements-oriented aspects of AI memory, and participated in the generative metaphorical design process.

- Professor Ian Arawjo served as the corresponding author and supervisor for the article, providing guidance throughout the framing, analysis, design exploration, and writing process.

2.1 Introduction

Agentic AI systems, characterized by a large language model coupled with the ability to call programmatic functions ("tools"), increasingly have "memory" components. However, the consistency of "AI memory," both in its implementation and its linguistic definition, are in flux. "AI memory" is now used to refer to everything from in-context storage of past interactions, to pre-written user directives injected into system prompts, to databases of user preferences [164], to Markdown files stored on the file-system that an agent can retrieve, read, and modify [8]. Across these varied implementations, memory serves common purposes—chiefly to *remember* intent, preferences, and details of past interactions, whether with the user, or of agents themselves.

Alongside this influx of attention to memory, primarily from industry, there has been little attention paid to the corresponding *interface design* decisions around AI memory. AI memory is not merely a technical problem but an *affordance* for user control, transparency, and explainability over agent decision-making. Different choices of technical architectures for memory present different affordances and constraints for designing interfaces for users to read and modify that memory—for instance, storing memories by fine-tuning the model makes later inspection near-impossible, whereas storing memories as a flat list of preferences is more auditable by users, but may lead to performance drops as details are lost. Accordingly, AI memory design necessarily involves a rich design space of architectural, operational, and interaction design choices, all of which can affect one another.

From a human-AI interaction standpoint, however, we still lack a systematic account of these design choices. While the functional and technical aspects of AI memory have been studied [113, 193], an interaction-oriented breakdown of the design of memory remains comparatively underexplored. What design space exists across memory architectures, operations, and interaction mechanisms, and where do current AI memory systems

fall short of what future interactive AI memory could provide? What are the downstream implications of memory structure, including how structural and operational constraints shape later decisions and affect interaction quality, agent capability, and temporal or computational costs?

To address these questions, we conducted a review of current AI memory tools and interfaces to arrive at a design space [24, 124] for AI memory, including how AI's memory is represented and structured, how these choices on AI memory design shape interaction, and what trade-offs they introduce. Using this design space as an analytic lens, we then show how existing systems instantiate these design choices, identify dominant patterns and metaphors, and discuss gaps that remain. Finally, we draw on generative metaphorical design [117, 160] to propose new, lesser-used metaphors for AI memory to spark ideas for addressing these gaps, and discuss how such metaphors could inform future AI memory systems.

2.2 Background

Here we provide a brief introduction to the history of AI memory, an overview of the current space of AI memory, and conceptual frameworks from communication theory that are increasingly employed to describe the problem of human-AI grounding [164, 177].

2.2.1 A brief history of AI "memory"

What "memory" means in AI systems and agent design has shifted over time. In early NLP, a model's "memory" was mostly understood as internal recurrent state or weight-encoded representations (e.g. LSTM [79])—that is, something closer to the term implicit memory [113, 158] in its current meaning, rather than a separate memory architecture. Later, with higher-capability LLMs such as GPT-4 [1] and techniques such as Chain-of-Thought (CoT) [184], retrieval augmented generation (RAG) [108], and tool-use frameworks like CoALA [169], alongside concepts like the context window, system prompts, and prompt engineering [21, 33, 67, 139], "memory" expanded to in-

clude explicit, external data storage and a clearer distinction between implicit and explicit, yet these "memories" were often spoken of as run-time auxiliary inputs rather than as the agent's enduring memory (e.g., injecting a list of user preferences as the "memory" to a system prompt, upon every call). More recently, LLM-based coding agents [7, 43, 64, 66, 138], enabled by larger context windows and stronger reasoning models (e.g., GPT-5 [140], Claude Sonnet/Opus 4.5 [10, 11], and Gemini 2.5 Pro [61]), increasingly control how external memory is written, updated, and reorganized, shifting memory management from human curation to agent-driven workspace maintenance, with users primarily supervising [148]. However, how to build *long-term* external memory at roughly human (or superhuman) levels remains relatively underexplored [113].

Meanwhile, there have also been attempts to systematically survey the technical components of AI agent memory in the emerging landscape. CoALA [169] draws on cognitive science and symbolic AI to describe modular memory components for language agents, classifying memory into short-term versus long-term and further dividing long-term memory into procedural [39, 70], semantic [23], and episodic [174] types. Liu et al. [113] systematically analyze AI agents by decomposing their memory-related components and mapping them to human cognitive and perceptual functions. Zhang et al. [193] survey memory mechanisms in LLM-based agents, covering both the structural design of memory modules and their evaluation. Rolls [153] compares and reviews the human brain and generative artificial intelligence from a neuroscience perspective, highlighting differences in the structure and operations of their respective memory systems. While these works clarify how AI memory systems are designed at an architectural level, AI memory from an interaction and design perspective—such as how architectural choices systematically shape an interactive system's affordances and thus the overall interaction patterns users encounter, and what sets of architectural and operational design choices are required to enable intended user interactions and meet desired design goals—remains comparatively less studied, if at all remarked upon.

2.2.2 Interfaces for AI memory in HCI

Several HCI studies offer useful inspiration for designing the interaction and operations of AI memory interfaces. Memolet [189] reifies "memories" from prior conversations into reusable objects, and lets users explicitly specify and manipulate a memory space where both the user and the agent can work with shared, groundable memory units. Memory Sandbox [85] visualizes an agent's conversational memories as data objects, so users can treat "what the agent remembers now and how it sees things" as a shared context they can adjust. Semantic Commit [177] supports updating a natural-language specification of user intent in long-running projects by helping detect and resolve non-local impacts and semantic conflicts, enabling users and agents to jointly track intent changes and manage a shared information space. Recently, Knoll frames memories as knowledge bases that are transferrable across agents [195], similar to Anthropic's later plug-n-play notion of "Skills" [8].

Even though HCI work on human–AI collaboration does not explicitly frame its contribution as "AI memory," many such systems also grapple with related interaction challenges. Systems like CollaborativeGym [166] explore how to visualize and manage an information space that humans and agents can jointly ground in, and how, within that space, a user and an agent might jointly navigate, connect ideas, and iteratively build a shared body of knowledge. GhostWriter [188] likewise externalizes style and context as editable documents that users can inspect and iteratively refine, helping ground generated text in user-provided context and better align it with the user's voice over time. Sensemaking tools such as Sensecape [168], systems designed to support human memory such as Graphologue [90], and writing assistants such as TaleBrush [35] can all serve as relevant reference points for designing AI memory systems' interfaces and supported operations.

2.2.3 Conceptual Perspectives on Communication, Context, and Memory

In addition to mapping the design space of memory systems, researchers draw on perspectives from communication that can help in building interactive agent memory

systems. Vaithilingam et al. [176] envision how a future AI game-design assistant might collaborate in design work and gradually build dynamic ground with a user over time, while Ma et al. [123] propose a Human-AI deliberation approach, derived from deliberation theory, in which humans and AI elicit each other's opinions through ongoing discussion and continually update decisions toward agreement.

Recent work in human-AI communication [19, 164, 176, 177] has applied the framing of "common ground" and "grounding" from Clark & Brennan [37, 38] towards the creation of agent systems that maintain analogous common ground between humans and AI. Common ground refers to the mutual understanding (intents, context, details) accumulated over time between communicating parties, such as when working together on a project [37]. Similar to a human actor, an AI agent must infer the user's intent and its context solely from the user's expression, and then make decisions and act accordingly; if the agent infers this incorrectly, it will fail to carry out what the user expects in the appropriate context. Therefore, it is important during interaction to verify that the agent's interpretation is aligned with the user's intent and context, and to adjust it when necessary. Moreover, to avoid the repeated cost of grounding, maintaining common ground formed across multiple sessions in a persistent and coherently connected form requires recording relevant knowledge in an additional memory store and continuously updating it, which is important from the perspective of long-term interaction. This knowledge store, like human grounding, must remain consistent and aligned to the other parties' current understandings.

Over long-horizon interaction, such iterative cycles can help the agent sustain a more stable context, enabling the agent to "learn" about the world through interaction with it. However, this process cannot operate reliably without an appropriately designed agent memory system. It may fail, for example, when the design cannot support the required level of relational network structure, or when it lacks the manipulable operations needed to sustain the feedback loops that arise in interaction. Therefore, designing interactive AI memory systems requires considering the structural aspects of memory, the operations it affords, and how these will be used in interaction, so that memories can be recorded, revised, connected, and validated during interaction.

Building on these concepts, industrial products and academic systems increasingly aim to support agent's building an understanding of user intent over time. Chat-based agents like ChatGPT [136] support custom memory that allows users, or the agent itself autonomously, to add context and persist information across sessions. Code agents like Claude Code [7] similarly support persistent instructions via CLAUDE.md and allow users to extend behavior through Agent Skills [8] and subagents, which can be invoked in their own scoped memory context. On the research side, PAIL [192] explicitly surfaces users' requirements in software development and support the process of refining and updating them over time, while Semantic Commit [177] shows how users and agents can collaboratively address emerging conflicts around user intent and revise their grounding as those intents are established and updated. Finally, GUM [164] envisions that an AI agent, watching the user as they work, will accumulate knowledge about the user over time that systems can use to ground and tailor later AI outputs. In what follows, we will outline this landscape more comprehensively.

2.3 Mapping the Current Design Space of AI memory

Here we examine the design space of AI memory—how different AI memory systems, shaped by their goals, intended interaction patterns, and practical constraints (e.g., LLM capability and cost, latency, and deployment setting), make structural choices and thereby afford specific operations to both agents and users. Under recurring combinations of goals and technical constraints, systems often converge on dominant patterns and the trade-offs they implicitly accept, alongside atypical but meaningful exceptions. Our goal is to help designers of interactive AI agent systems make purpose-fit choices under their goals and constraints, and to help researchers identify gaps that motivate new interaction designs.

We conducted two surveys on AI memory systems' design choices, one focused on architectural design choices (Section 2.3.1; Table 2.I) and the other on operational design choices (Section 2.3.2; Table 2.II), and then derived the design space from these results (Figure 2.1).

To establish the axes of the design space and the options within each axis, we began with a set of seed axes, derived from a provisional framing of architectural choices, interaction patterns, and provided operations, and refined them during coding as needed. In the architectural survey, coding was conducted at the level of the memory system as a whole, whereas in the operational survey, distinct subsystems were sometimes treated separately when they exposed different memory roles, interfaces, or affordances. The first author led the coding process, while the coauthors reviewed emerging categories, discussed ambiguous cases, and provided feedback on changes to the scheme. Categories changed whenever recurring cases no longer fit the existing scheme cleanly or became too internally complex to be adequately described as a single option. Whenever an axis or its options changed, we revisited and updated the affected codes to keep the corpus consistent. We iterated this process until the axes and option sets stabilized, that is, until additional cases could be placed within the existing axes and option sets without requiring further changes to them.

Our corpus was assembled through snowballing from a small set of seed systems, well-known in the space (e.g., LangGraph [103], Claude Code [7], Mem0 [34]), rather than defined as an exhaustive survey. We intentionally focused on tools and APIs, rather than academic texts, which often lag behind industry and, in HCI, do not correspond to widely-used systems; among HCI systems, we preferred those whose designs are sufficiently well-documented, for example publications that describe the underlying structure in detail or accessible code-bases. For each case, we relied primarily on papers, documentation, and publicly accessible repositories, generally prioritizing directly inspectable implementations or APIs when available. When later product evolution had introduced features not present at the time of publication, however, we prioritized the paper's description and the implementation state corresponding to that publication period.

In the architectural survey, we included systems in which memory was a substantive design target and excluded cases whose architectures were not publicly available enough to analyze reliably, such as ChatGPT [136]. In the operational survey, we included additional cases beyond those in the architectural analysis whose operation affordances are

observable through explicit specifications or publicly accessible platforms. For systems that combine multiple layers yet assign them clearly distinct roles and layer-specific interactions (e.g., LangGraph [103]/LangMem [104], or Claude Code [7]'s workspace, agent.md, and chat-memory layers), we analyzed the operations of each layer separately.[1] In our operation analysis, we excluded cases that do not have independent operations of their own, such as Claude's Agent Skills [8] and subagents [9].

In total, we conducted an architectural analysis of 14 distinct systems and an operational analysis of 15 distinct systems (22 subsystems in total). Based on these results, we constructed the design-space figure (Figure 2.1) and the corresponding tables that summarize each system's observed architectural (Table 2.I) and operational (Table 2.II) design choices.

[1] In contrast, we did not separate Mem0 and Mem0g [34] because, in the current implementation, Mem0g's functionality is not exposed via independent endpoints but is toggled within Mem0's existing API surface.

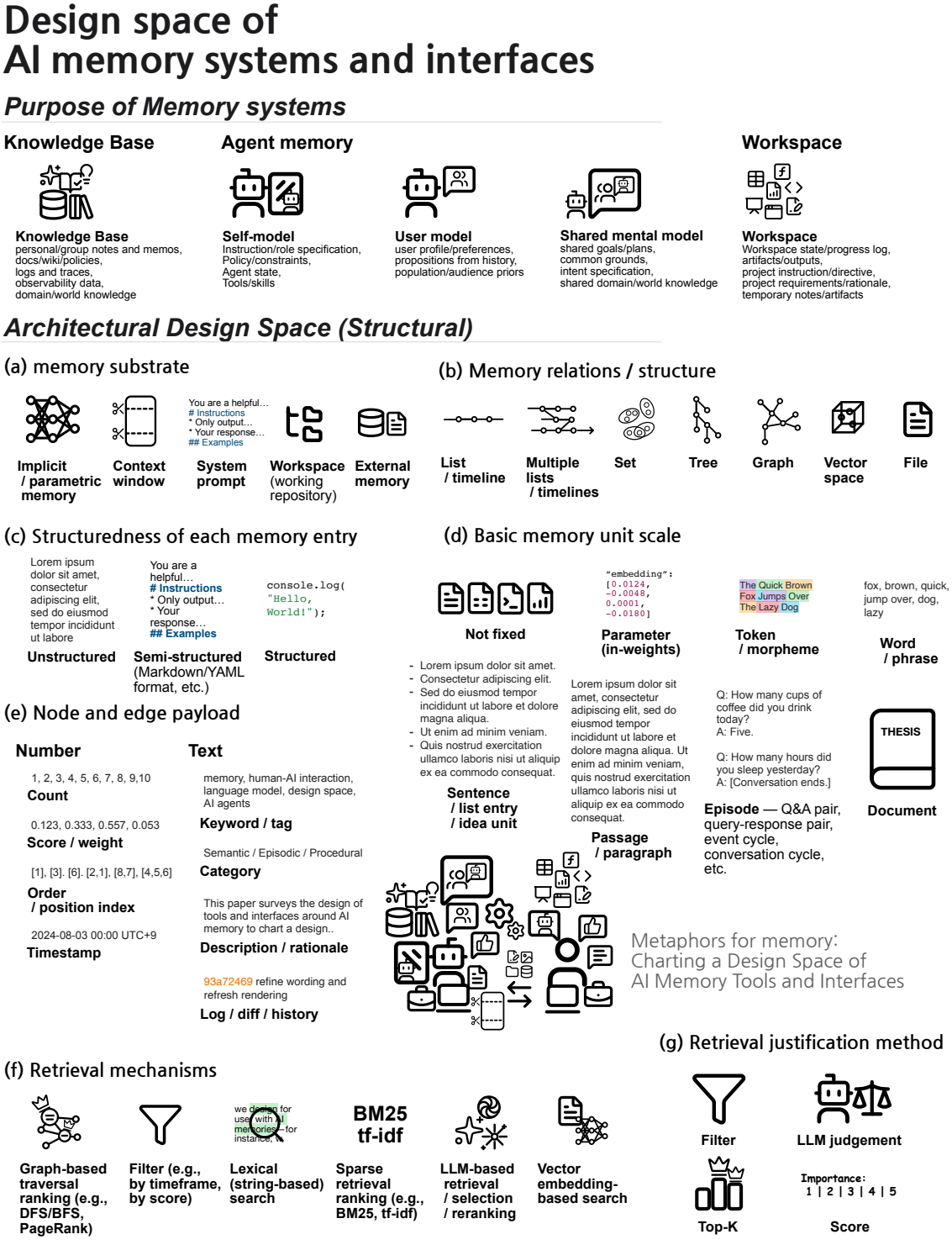


Figure 2.1: Design space of AI memory systems and interfaces

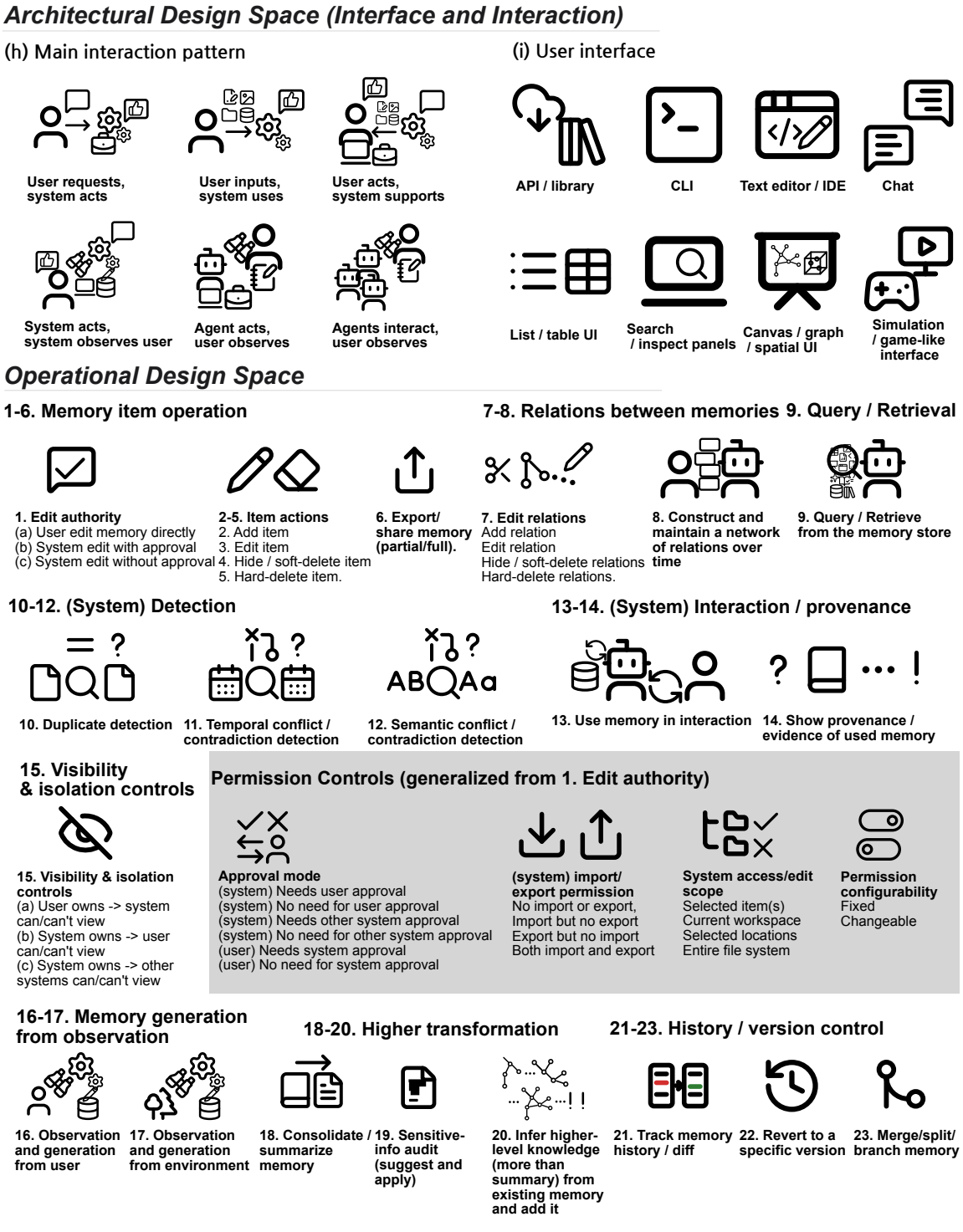


Figure 2.1: Design space of AI memory systems and interfaces (continued)

Figure 2.1 provides an overview of the resulting AI memory design space by synthesizing the dimensions identified in our architectural (Table 2.I) and operational (Table 2.II) surveys. The figure is organized into three parts: Purpose of Memory Systems, Architectural Design Space, and Operational Design Space. We present it upfront as an index and map to the dimensions that are later unpacked in Section 2.3.1 and Section 2.3.2. In the architectural part, the labels (a)–(i) correspond to the dimensions later instantiated in Table 2.I. In the operational part, the numbers and number ranges correspond to the operational rows and grouped ranges in Table 2.II. We place Purpose of Memory Systems first because it is important for understanding the overall AI memory design space, but do not present it as a separate system-level survey because these purposes did not form clear system-level distinctions and were typically pursued simultaneously within the same systems. Gray cells indicate theoretically relevant options that were retained in the design space even when they were not directly instantiated in the surveyed corpus.

2.3.1 Architecture

Table 2.I summarizes the architectural design choices of the memory systems in our corpus. Rows correspond to systems, and columns correspond to key design dimensions. Each cell lists the option(s) adopted by a system; multiple items within a cell are separated by line breaks. When multiple items are listed, all of them are present in the system (often serving different roles or appearing in different parts of the system); however, earlier items typically reflect the system's most central or emphasized design choices, while later items tend to play supporting roles. For example, in HippoRAG [91]'s "Basic memory unit scale," we list the retrieval-level unit (word) before the storage-level unit (document), since the system's central design emphasis lies in the retrieval layer. A dash (–) indicates that the system does not include any corresponding structure or mechanism for that dimension.

2.3.1.1 Memory substrate (Table 2.I, column (a))

All of the analyzed systems include external memory as a main memory substrate. In addition, systems may provide a workspace, context window, and/or a system prompt, depending on what the system aims to demonstrate. However, even external-memory-only systems often propose end-to-end workflows that help users fully leverage the memory substrate design space with other systems.

2.3.1.2 Memory relations / structure (Table 2.I, column (b))

Systems whose core contributions are in memory structure generally adopt graph-based structures [34, 91, 147, 186]. In contrast, systems whose core contributions are not in structure but in interaction typically adopt simpler forms, such as a temporal list [85, 142, 143, 164] or embedding-vector-based grouping [189]. Semantic Commit [177] is an exceptional case, although its main contribution is in interaction, it identifies the limitations of list- or embedding-based approaches as a problem and introduces an additional graph structure.

2.3.1.3 Structuredness of each memory entry (Table 2.I, column (c))

While early systems (e.g. MemGPT [142], Memory Sandbox [85]) stored memories as unstructured strings, most systems adopted schema-based entries (e.g., JSON) to attach metadata. Claude's agent skills [8] and subagents [9], meanwhile, use Markdown with a YAML front matter header, reflecting an assumption that memory operations are mostly mediated by the LLM and thus prioritizing malleability over strict schema enforcement.

2.3.1.4 Basic memory unit scale (if (semi-)structured) (Table 2.I, column (d))

The scale of the memory unit varies by purpose. For storing full logs, most systems kept each memory item as a whole (at the document level). For interaction or editing, they tended to use smaller granularities (e.g. passages, episodes, or ideas) sometimes combining multiple units. Most additional retrieval layers operate at the word level.

2.3.1.5 Node and edge payload (Table 2.I, column (e))

Node and edge payload depend heavily on the memory structure a system chooses. Compared to graph-based systems, list-based and hierarchy-based systems have limited relational structure, which constrains payloads to minimal signals such as scores or timestamps. However, in our analysis, graphs in those graph-based systems largely took the form of triple-based knowledge graphs, so edge payloads were limited to keywords. In vector-embedding stores (e.g., Memolet [189]), the embedding space itself functions as the predetermined relational structure. Therefore, there are constraints on arbitrarily adding additional edge payloads, making this design option limited in practice.

2.3.1.6 Retrieval mechanisms (Table 2.I, column (f))

Retrieval mechanisms also depend heavily on the preceding design choices. Systems that include keywords, timeframes, or scores often provide filtering or search functions based on those fields. Graph-based systems similarly tend to support graph traversal (e.g., DFS/BFS). A notable case is HippoRAG [91], which performs Personalized PageRank [76] over a knowledge graph for retrieval. Most tools also adopt content-level retrieval over each memory unit, including lexical search, vector embedding search, BM25 [151], and/or LLM query-based retrieval.

2.3.1.7 Retrieval justification method (Table 2.I, column (g))

While retrieval mechanisms and retrieval justification methods are not strictly paired, they tended to co-vary in our corpus. Systems using deterministic retrieval usually also relied on deterministic justification criteria, such as filters, scores, and top-k cutoffs [6, 34, 91, 103, 142, 147, 186, 189]. By contrast, systems that involved LLMs in retrieval typically used LLM judgment as an additional basis for justification, for example by assigning importance scores [143], reranking retrieved candidates [164], or applying a final filtering step after an initial deterministic retrieval [177].

2.3.1.8 Main interaction pattern (Table 2.I, column (h))

Dominant interaction patterns fell into four types: (1) users request actions and the system performs memory operations [34, 91, 147, 186]; (2) users act and the system supports memory work [85, 177, 189]; (3) agents autonomously observe and interact with the environment while accumulating memory, and users mostly observe [6, 143]; or (4) users input memory into the store for agents' later use [8, 9, 103]. A distinctive case is the General User Model [164], where memory operations are driven primarily through observing the user and the user is positioned as the observed subject rather than the observer of an acting agent, unlike the other systems where memory operations are typically initiated by users through some instruction, data, or initial environmental conditions and users are more often positioned to observe and evaluate the agent.

2.3.1.9 User interface (Table 2.I, column (i))

The interface was most often provided in API and library. When GUIs were provided, memories were typically displayed as sequential (table) lists or chat logs, often with extra panels (e.g., inspector panel). Two distinctive cases are Memolet [189]'s 2D card-grid canvas and Generative Agent's simulation-game-like UI.[2]

[2]Note, however, that in Generative Agents the primary purpose of the UI was to display the simulation than the memory itself.

Table 2.I: Architectural design choices across AI memory systems

System	(a) Memory substrate	(b) Memory relations / structure	(c) Structuredness of each memory entry	(d) Basic memory unit scale	(e) Node and edge payload	(f) Retrieval mechanisms	(g) Retrieval justification method	(h) Main interaction pattern	(i) User interface
MemGPT[142]	Context window System prompt External memory	Timeline	Unstructured	Document Passage	Timestamp	Timeframe filter Lexical search Vector embedding	Filters Top-K	User requests, system acts	API/Library, Inspector panel+chat+list
LangGraph (BaseStore) [103]	External memory	Tree	Structured	Not fixed	Category Timestamp	Lexical search	Top-K Filters	User inputs, system uses	API/Library
HippoRAG [91]	External memory	Graph	Structured	Word Document	Count Keyword	Personalized PageRank	Top-K	User requests, system acts	API/Library
Zep/Graphiti [147]	External memory	Multi timeline Graph	Structured	Word Sentence Episode	Keyword Timestamp	Vector embedding BM25 Graph (BFS/DFS) Timeframe filter	Top-K Filters	User requests, system acts	API/Library
A-MEM [186]	External memory	Graph	Structured	Sentence	Keyword	Vector embedding Graph (BFS/DFS)	Top-K	User requests, system acts	API/Library
Mem0/Mem0g [34]	External memory	Timeline Graph	Structured	Document Episode Sentence Word	Keyword	Vector embedding BM25 Graph (BFS/DFS)	Top-K	User requests, system acts	Table
AriGraph [6]	Context window External memory	Timeline Graph	Structured	Document Episode Sentence Word	Keyword	Vector embedding Graph (BFS/DFS)	Top-K	Agent acts, user observes	Text game interface
Generative Agents [143]	Context window System prompt External memory	Multi timeline	Structured	Word Sentence	Score Weights Timestamp	Lexical search Score Filter	Filters LLM judgement Scores Top-K	Agents interact, user observes	Sims-like observational interface
General User Model [164]	Workspace External memory	List Timeline	Structured	Sentence Episode	Timestamp Score Weights	BM25 LLM	Top-K LLM judgement	System acts, system observes user	API/Library
Semantic Commit [177]	Workspace External memory	List Graph	Structured	Word Sentence	Keyword	LLM Personalized PageRank	LLM judgement	User acts, system supports	Center list with search/inspect panels
Memolet [189]	Context window Workspace External memory	Vector space Set	Structured	Document Episode	-	Vector embedding	Top-K	User acts, system supports	Chat and card-grid canvas
Memory Sandbox [85]	Context window System prompt	List	Unstructured	Document Sentence	-	-	-	User acts, system supports	Chat
Claude Agent Skills [8]	External memory Context window	Tree	Semi-structured	Document	-	-	-	User inputs, system uses	-
Claude Code Subagents [9]	External memory System prompt Context window	Single file	Semi-structured	Document	-	-	-	User inputs, system uses	CLI and IDE

Note. Each entry lists the adopted options for a system; within a cell, earlier items indicate relatively higher importance.

2.3.2 Operations

Table 2.II: Operational design choices across AI memory systems

	LangGraph [103]				Zep [147]											ChatGPT [136]				Claude Code [7]		
Operation	Mem GPT [142]	Base Store	Lang Mem [104]	Hippo RAG [91]	Zep (cloud)	Graph iti	A-MEM [186]	Mem0 [34]	Ari Graph [6]	Genera tive agents [143]	General User Model [164]	Seman tic Commit [177]	Memo let [189]	Memory Sandbox [85]	Spec Story [29]	Instruc tion	Memory	Project files	Chats	Work space	CLAUDE .md	Chats
1-6. Memory item operation																						
1. Edit authority																						
(a) user edit memory directly	✓	✓	✓	✓	✓	✓	✓	✓	-	✓	-	✓	✓	✓	-	✓	✓	✓	✓	✓	✓	✓
(b) system can edit under user approval	-	-	-	-	-	-	-	-	-	-	-	✓	-	-	-	-	-	-	-	✓	✓	-
(c) system can edit without user approval	✓	-	✓	✓	✓	✓	-	✓	✓	✓	✓	-	✓	✓	-	-	✓	-	-	✓	✓	-
2. Add item																						
(a) user	✓	✓	✓	✓	✓	✓	✓	✓	-	✓	-	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
(b) system with approval	-	-	-	-	-	-	-	-	-	-	-	✓	✓	-	-	-	-	-	✓	✓	✓	-
(c) system without approval	✓	-	✓	-	-	✓	-	✓	✓	✓	✓	-	✓	△	✓	-	✓	-	-	✓	✓	✓
3. Edit item																						
(a) user	△	✓	✓	-	-	✓	✓	✓	-	-	-	✓	-	✓	✓	-	-	-	-	-	-	-
(b) system with approval	-	-	-	-	-	-	-	-	-	-	-	✓	-	-	-	✓	-	-	✓	✓	✓	-
(c) system without approval	△	-	✓	-	-	-	-	✓	✓	-	-	-	-	-	-	-	-	-	✓	✓	✓	-
4. Hide / soft-delete item																						
(a) user	-	-	-	-	-*	-	-	-	-	-	-	-	✓	✓	-	-	-	-	-	-	-	-
(b) system with approval	-	-	-	-	-	-	-	-	-	-	-	-	-	△	-	-	-	-	-	-	-	-
(c) system without approval	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
5. Hard-delete item																						
(a) user	✓	✓	✓	✓	✓	✓	✓	✓	-	-	-	✓	-	✓	✓	✓	✓	✓	✓	✓	✓	-
(b) system with approval	-	-	-	-	-	-	-	-	-	-	-	✓	-	-	-	-	-	-	-	✓	✓	-
(c) system without approval	△	-	✓	-	-	✓	-	✓	✓	-	-	-	-	-	-	-	✓	-	-	✓	✓	-
6. Export/share memory (partial/full)																						
(a) user	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	✓	-	-	-
(b) system with approval	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
(c) system without approval	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
7-8. Relations between memories																						
7. Edit relations (add/delete/update; single or multiple relations)																						
(a) user	-	△	-	-	✓	✓	✓	-	-	-	-	-	✓	-	-	-	-	-	-	-	-	-
(b) system with approval	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
(c) system without approval	-	-	-	-	-	-	✓	✓	✓	-	-	✓	✓	-	-	-	-	-	-	-	-	-
8. (system) Construct and maintain a network of relations over time	-	-	-	✓	✓	✓	✓	✓	✓	-	-	✓	✓	-	-	-	-	-	-	-	-	-
9. Query / Retrieval																						
9. Query / retrieve from the memory store																						
(a) user	✓	✓	✓	✓	✓	✓	✓	✓	-	✓	✓	-	✓	△	✓	✓	✓	✓	✓	✓	✓	✓
(b) system with approval	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	✓	✓	-
(c) system without approval	✓	-	✓	✓	-	✓	-	✓	✓	✓	✓	-	✓	△	-	✓	✓	✓	✓	✓	✓	-
10-12. Detection																						
10. (system) Duplicate detection	-	-	✓	△	✓	✓	-	✓	✓	-	✓	-	-	-	-	-	-	-	-	○	○	-
11. (system) Temporal conflict/contradiction detection	-	-	-	-	✓	✓	-	-	△	-	-	-	-	-	-	-	-	-	-	○	○	-
12. (system) Semantic conflict/contradiction detection	-	-	✓	-	✓	✓	-	✓	✓	-	-	✓	-	-	-	-	-	-	-	○	○	-
13-14. Interaction & provenance																						
13. (system) Use memory in interaction	✓	-	✓	✓	-	-	-	✓	✓	✓	-	✓	✓	✓	-	-	-	-	✓	-	-	✓
14. (system) Show provenance/evidence of used memory	△	-	-	✓	-	-	-	-	△	-	-	✓	✓	-	-	-	-	-	✓	-	-	✓

Legend: ✓ = Yes; - = No; ○ = Workaround (not a built-in feature; achievable via LLM calls using the system's supported functions); △ = Partially yes (yes in principle, but limited implementation)
* Legacy Zep soft-delete was removed in the current release.

Table 2.II: Operational design choices across AI memory systems (continued)

		LangGraph [103]			Zep [147]											ChatGPT [136]				Claude Code [7]		
Operation	Mem GPT [142]	Base Store	Lang Mem [104]	Hippo RAG [91]	Zep (cloud)	Graph iti	A-MEM [186]	Mem0 [34]	Ari Graph [6]	Genera tive agents [143]	General User Model [164]	Seman tic Commit [177]	Memo let [189]	Memory Sandbox [85]	Spec Story [29]	Instruc tion	Memory	Project files	Chats	Work space	CLAUDE .md	Chats
15. Visibility & isolation																						
15. Visibility & isolation controls																						
(a) User owns → system can/can't view	-	-	-	-	-	-	-	-	-	-	-	-	-	✓	-	-	-	-	-	-	-	-
(b) system owns → user can/can't view	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	✓	-	-	-	-	-	✓
(c) system owns → other systems can/can't view	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
16-17. (system) Memory generation from observation																						
16. Observation and generation from user	✓	-	✓	-	-	-	-	-	-	-	✓	-	-	-	✓	-	✓	-	-	-	-	-
17. Observation and generation from environment	-	-	-	-	-	-	-	-	✓	✓	-	-	-	-	-	-	-	-	-	-	-	-
18-20. Higher transformation																						
18. Consolidate / summarize memory																						
(a) system with user approval	-	-	-	-	-	-	-	-	-	-	-	-	✓	✓	-	-	-	-	-	○	○	○
(b) system without user approval	-	-	-	-	✓	✓	-	-	-	-	-	-	-	-	-	-	✓	-	-	-	-	✓
19. Sensitive-info audit (suggest and apply)																						
(a) system with user approval	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	○	○	○
(b) system without user approval	-	-	-	-	-	-	-	-	-	-	✓	-	-	-	-	-	✓	-	✓	-	-	-
20. Infer higher-level knowledge (more than summary) from existing memory and add it																						
(a) system with user approval	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	✓	○	○	○
(b) system without user approval	-	-	✓	-	-	-	-	-	△	✓	✓	-	-	-	✓	-	-	-	✓	-	-	-
21-23. History / version control																						
21. (system) Track memory history / diff	-	-	-	-	-	-	-	✓	✓	△	△	-	-	-	-	-	✓	-	-	○	○	✓
22. Revert memory to a specific version																						
(a) user	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	✓	-	-	○	○	△
(b) system with approval	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	○	○	-
(c) system without approval	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	○	○	-
23. Merge/split/branch memory																						
(a) user	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	○	○	-
(b) system with approval	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	○	○	-
(c) system without approval	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	○	○	-

Legend: ✓ = Yes; - = No; ○ = Workaround (not a built-in feature; achievable via LLM calls using the system's supported functions); △ = Partially yes (yes in principle, but limited implementation)

Table 2.II enumerates the memory operations supported by each system. The columns list the memory systems, grouped by layer, while the rows list operations together with the actor responsible for each operation. Each cell is coded as Yes (✓) when the system provides the capability and No (-) otherwise. For edge cases, we use a circle (○) when the operation is not supported as a specified feature but can still be achieved indirectly

(e.g., via LLM calls or workaround workflows), and we use a triangle (△) to mark partially supported cases where the operation is implied by the system's definition or data model but is incomplete in practice (e.g., General User Model [164] is append-only with timestamps, so memory entries themselves form a rudimentary history, but it would be a stretch to say it supports "history tracking" as an explicit operation).

2.3.2.1 Item Operation (Table 2.II, rows 1-6)

Memory-manipulation authority tended to follow the dominant interaction pattern of the system. User-led systems gave users more control over memory operations while agent authority was often more limited. Agent-led systems tended to have reversed pattern. User-approved edits were rare. When agents could edit memory autonomously, edits usually occurred without user consent. Soft delete/hide was absent in most systems except Memolet [189] and Memory Sandbox [85]. Beyond ChatGPT [136]'s chat sharing, we did not observe any in-system export/sharing features for memory. In most cases, users had to download memories or implement sharing themselves. We found no systems that allow agents to autonomously export or share memory.

2.3.2.2 Relations between Memories (Table 2.II, rows 7-8)

Where explicit relationships and edges do exist, systems often allowed some relationship editing; however, in most cases this meant either users manually editing edges one by one or agents automatically updating edges during their update process. We did not observe any systems that support user-approved relationship edits. Meanwhile, it was notable that most graph-based memory systems aim to construct and maintain a network of relations over time.

2.3.2.3 Query and Retrieval (Table 2.II, row 9)

Most systems provided query authority to both users and agents. Even when query functionality was absent for the user and/or the agent, in most cases it was because queries were not necessary under the system requirements rather than due to structural

limitations.

2.3.2.4 Detection (Table 2.II, rows 10-12)

Some systems provided additional detection operations beyond basic querying. Zep [147], Mem0 [34], Arigraph [6], and the General User Model [164] support write-time deduplication and handle duplicates by skipping, replacing, or grouping entries. In parallel, Zep [147] supports temporal-conflict detection to catch temporal inconsistencies across memory contents. Separately, Zep [147], Mem0 [34], Arigraph [6], and Semantic Commit [177] provide LLM-based semantic contradiction detection.

2.3.2.5 Interaction and Provenance (Table 2.II, rows 13-14)

While most AI memory tools were ultimately designed to operate as part of a user–agent or agent–agent interaction pipeline, whether interaction-related operations were implemented in-system depended on the scope the system aimed to cover. Among them, Semantic Commit [177], Memolet [189], and the chat interfaces of Chatgpt [136] and Claude [7] also provided provenance about which memories the agent referenced during interaction.Some systems do not explicitly show evidence or provenance for interactions, but it is still possible to see which memories were retrieved at that point through logs or return values. We treated it as partially supported in such cases.

2.3.2.6 Visibility and Isolation (Table 2.II, row 15)

There were few cases where in-system operations allowed controlled differential visibility or memory isolation between users and agents, aside from Memory Sandbox [85]'s selective use of memories in chat and ChatGPT [136]/ Claude Code [7] instruction prompts that are visible only to the agent and selectively disclosed to the user.

2.3.2.7 Observe (Table 2.II, rows 16-17)

Systems often allowed agents to observe and record user behavior to build long-term memory without manual user curation. In simulation settings, observing and recording

external environments and interactions was essential for agents to perceive, judge, and act autonomously.

2.3.2.8 Higher-level Memory Transformation (Table 2.II, rows 18–20)

Higher-level memory transformations beyond atomic edits became more common as agents gained greater memory-control authority. Consolidation mainly served to streamline memory stores, either compressing memory store (Zep [147], ChatGPT [136], Claude Code [7]) or producing more readable memory sets for users (Memolet [189], Memory Sandbox [85]). Auditing appeared where sensitive disclosure is likely (General User Model [164], ChatGPT [136]), but in both systems the rule set was predetermined and not user-configurable. Inference supported autonomous action (Generative Agents [143], Arigraph [6]) or proposition extraction from observed user interactions (General User Model [164], SpecStory [29], ChatGPT [136]).

2.3.2.9 History Tracking and Version Control (Table 2.II, rows 21–23)

In our analysis, even when systems supported memory deletion, support for version control or history was rare. Append-only memory systems were coded as partial support, since the current state functions as de facto versioning even without explicit history/version-control features. For this reason, most tools did not support higher-level version-control operations such as reverting, merging, splitting, or branching memory. Claude Code [7] can perform these operations by executing Git commands, but they are not supported via in-system operations; we coded this as supported only via a workaround.

2.3.3 Implications of architectural decisions of AI memory

Across these analyses, we derived several implications about how specific design choices in AI memory systems shape interaction-level affordances and trade-offs, while also identifying several gaps in current dominant design patterns. Sections 3.3 and 3.4 briefly discuss these implications and gaps.

2.3.3.1 Implicit versus explicit memory

Designers' decisions about whether information should live in implicit or an explicit memory can be clarified by analogy to human implicit versus explicit memory. Implicit memory corresponds to knowledge we want to surface reflexively and immediately through training, whereas explicit memory corresponds to a knowledge network that is accumulated over time and kept in a form that can be inspected and edited. The two approaches involve a familiar set of trade-offs. Implicit memory is fast, but difficult to update, and it is often hard to trace the reasoning or provenance behind a response. Explicit memory can be slower, but it is comparatively easier to edit, delete, or scope, and it better supports tracing which stored information contributed to a given output. Explicit memory also makes it easier to construct context-dependent responses by retrieving different combinations of memories depending on the situation and inferred intent.

In practice, the choice often comes down to whether memory is meant to enable reflex-like behavior (implicit) or to preserve inspectable context and rationale (explicit). For example, when memory updates are rare (e.g., language rules, basic mathematics, general programming knowledge), personalization is minimal (e.g., systems targeting large, heterogeneous user groups), speed is critical, and provenance is not strongly required, encoding the knowledge into implicit memory can be beneficial. Conversely, when memory updates are frequent, personalization matters, provenance during reasoning is important, or the same question requires different answers depending on context and intent, it can be advantageous to store the relevant knowledge in an explicit layer as an editable network, as long as the latency cost remains acceptable.

2.3.3.2 How should we scope and target memory substrates?

As Figure 2.1 and Table 2.I suggest, even when a system uses explicit memory, there are multiple places where memory can be situated and manipulated. The same information can live in the system prompt, in an external store, or in a shared workspace where humans and an AI co-produce artifacts. Alternatively, a system may rely on additional agents with independent context windows, as in Claude Code subagents [9]. In practice,

an agent ends up using all of these substrates. Even if a particular system covers only a subset of them, designers should still consider how it connects to the rest.

For example, suppose a memory system is designed as a work-oriented database that both the agent and users can reference. In real use, the agent will not rely on the database alone, but will act within its context window under a system prompt that specifies roles, rules, and instructions, and it will retrieve from the database as needed to compose responses. At the same time, users typically want more than a Q&A interface: they want the agent to understand the workspace they are operating in as if it were "looking at it together" with them, and to retain task context so it can bring in the right information at the right time. In other words, even if one build just a database, users will still expect the system to connect that store with documents, chats, workspace artifacts, and traces used in other context windows in a coherent way. Similarly, even if one develops only one part of the interaction pipeline, it is not enough to focus solely on the system-prompt instructions, and users will want the system to integrate with external stores and workspaces.

Ignoring these connections may not break the system immediately, and the problem may not be visible at first, but it almost inevitably creates friction over time. For instance, if two incompatible systems exist, one strong at storage and another strong at dialogue, reasoning, or tool instructions, users end up switching back and forth and manually copy-pasting, or they eventually give up on one of them. Therefore, even when designing a single system, it is worth keeping the entire agent pipeline in mind, including how users will actually interact with it and how naturally it interoperates with other tools, stores, and workspaces.

2.3.3.3 Granularity and density of memory components

When designing structures to hold and manage knowledge, there is an inherent trade-off between fidelity and usability: larger memory units preserve more context but are less likely to be read, while smaller units are easier to scan but make faithful reconstruction harder. The same trade-off applies to relations: simple edges (e.g., scalar weights) are easy to process but carry little explanatory content, whereas richer edges (e.g., predi-

cates, keywords, short rationales) preserve "why" at higher processing cost. The trade-off is similar in terms of relational density: sparse structures (e.g., lists and independent notes) are easy to navigate but cannot explicitly encode complex, long-range relations. In contrast, denser structures (e.g., trees and graphs) can capture such relations but make navigation more difficult.

These granularity- and density-related processing trade-offs also apply to users, but in a somewhat different way. In general, people are accustomed to reading text at the sentence level, consuming information one unit at a time and largely in a single direction, and they also tend to avoid overly long texts. Therefore, when designing memory systems that humans are expected to consult directly alongside agents, one must consider not only what kinds of data an agent can process well, but also how humans actually read. For example, ontology-style knowledge graphs expressed as noun–verb triples often do not make it clear how the content should be read, their traversal direction is not consistent, and each unit may contain only one or two words, which makes it hard for humans to reconstruct the underlying meaning. For similar reasons, presenting too much information at once can also increase cognitive workload, especially when the content is not written in a form that is easy to read.

2.3.3.4 Representing relations among memories

Designing agent memory is not only about expanding what can be stored, but also about how memories and their relations are represented so that the agent can later reconstruct context. Stored information becomes useless if retrieval or relational inference fails, and such retrieval cannot work well without making the relations among memory fragments explicit.

From this perspective, the choice of relational structure introduces downstream trade-offs in retrieval, interaction-time inference, and update workflows. List- or timeline-based memories can be the easiest to navigate because their relational expressiveness is largely confined to ordering, but they offer limited support for preserving richer contextual links across entries. Hierarchical folder structures or grouped sets offer more freedom than flat lists, yet they force each item into a single classification, making nu-

anced, cross-cutting relations hard to represent and misclassifications costly to recover. Graph-based structures, by contrast, offer the greatest flexibility for representing relations, and a well-maintained graph can enable highly efficient retrieval by traversing explicitly encoded links. However, graph construction and maintenance are expensive, and in practice graphs are often difficult to keep clean enough to realize their theoretical advantages. This overhead becomes especially salient in human–AI collaboration settings, where human attention and reading speed are far slower than agent-side processing.

To mitigate these trade-offs, some systems stack multiple layers rather than committing to a single structure, for example, adding a timeline- and keyword-oriented graph layer over list-like memory entries [34], or overlapping multiple lists and timelines to compensate for sparsity in any one view [143, 147, 164]. Because these structural choices are difficult to revert once a system is implemented and populated with data, it is important to clarify the intended goals and interaction scope early, and to select a representation that matches the expected update and retrieval workflows.

2.3.3.5 Choice of retrieval mechanisms

Retrieval mechanisms introduce an additional layer of trade-offs and are often combined across layers. Systems commonly pair deterministic methods—such as lexical search, time-based filtering, or graph traversal (e.g., BFS/DFS)—with additional retrieval layers, including embedding-based retrieval, BM25-style lexical retrieval, and/or LLM-mediated query interpretation and reranking. External vector embeddings often have lower build and per-query costs than knowledge-graph approaches, but their induced relational structure is difficult to update. Therefore, for example, even when a user believes A should be closer to C than to B, the system cannot easily "edit" that relationship if it is determined by a fixed embedding space. Embeddings also struggle with context-dependent meaning, such as name disambiguation or indirect references (e.g., "you know who"), where explicit relational representations can be more reliable.

When context sensitivity and controllable relations are central to a system's goals, paying the higher construction and maintenance costs of graph-based structures may be justified. In such settings, using an LLM to construct and operate over the graph can help

produce relations that better reflect intended meaning and context. However, designers must anticipate higher and more variable costs (and often less predictable latency) when LLMs are in the loop.

2.3.3.6 Interdependence between architecture and operations

Importantly, some operations and interaction patterns are contingent on earlier architectural choices. For example, traversing a relation graph and expanding a memory network is not feasible in a list structure because there are no explicit edges to traverse. Likewise, as noted earlier, editing edges is not possible in a list-based structure or when relying on precomputed vector embeddings. More broadly, higher-level capabilities depend on whether nodes and edges carry appropriate payloads: tracing and basic versioning are prerequisites for higher-level version-control operations such as merge, branch, and revert.

Missing capabilities can also implicitly discourage desirable user behavior. Without adequate versioning, users may become reluctant to delete memories, causing the store to grow without pruning. Without appropriate visibility controls, isolation, and mechanisms for managing links between memories, users may instead create private, disconnected stores that never get incorporated into the shared memory, leading to fragmentation and lower overall utility. Similarly, users may want to share a specific part of the memory store but fail to find it due to weak query functionality. In the worst case, the system devolves into a workflow where users rely on a few retrieved snippets and reconstruct the rest in their heads, negating the intended benefits of external memory. In many cases, poor query performance is itself downstream of structural or operational issues, such as an ill-organized internal representation.

Because these bottlenecks often arise from the interplay between architecture and operations, designing an interactive memory system requires anticipating the interaction flows and operations users will actually perform, and then choosing architectures and operations that best support those flows. This also means considering trade-offs holistically, since the effects of each design choice can compound across the system. It is often valuable to explore multiple design directions and iterate, rather than committing early

to a single representation.

2.3.4 Gaps in Current Dominant Design Choices

2.3.4.1 Underappreciated coupling between architecture and operation

Some operations in our corpus are not fully supported by the underlying architecture, so they may not produce the interactions users expect. For example, in Memolet [189], grouping or interpolation may seem to affect retrieval, but these interactions primarily reorganize interface-level state while memories remain anchored to fixed vector embeddings. This matters because when user manipulations do not translate into system operations, users may misread a design mismatch as poor performance or a limitation of the AI technology itself. Therefore, designers should recognize that choices of memory architecture and operations involve clear trade-offs that shape user interactions.

2.3.4.2 Gap in recording relationships among memories

Many systems in our corpus provide limited support for recording and maintaining relations between memories, though such relations are essential for later context reconstruction. For example, triple-based knowledge graphs [34, 91] may lose needed relations because subject-predicate-object encoding strips away nuances in the original expression. Once lost, these cues must be reconstructed at each use, causing repeated inference costs while risking distortion. When constraints allow, designers should preserve richer contextual links within the memory network.

2.3.4.3 Lack of user-agent common grounding steps

Many systems also lack a process for checking whether the memories and contextual relations inferred by the agent align with what the user actually means. For example, in General User Model [164, p. 16], participants expressed a need for clarification when the system turned observations into propositions or suggestions with low confidence. Without this clarification and grounding step, users may remain uncertain whether the agent's interpretation matches their intent, and the agent may act in ways that conflict

with it. Interactive agent memory systems should incorporate clarification and grounding steps into human-AI collaboration workflows to prevent misalignment.

2.3.4.4 Accumulate-only lifecycle with no versioning

Most systems in our corpus do not support history tracking or version-control operations, and instead adopt a single-version memory lifecycle that is largely accumulate-only, with limited editability and, at most, hard deletion. In such settings, users may focus only on accumulating data while avoiding edits to the memory network to prevent irreversible loss, which can gradually degrade its overall quality. Although any memory might become useful at some point, users should still be able to distinguish higher- and lower-importance memories and move less important ones into a lower-salience space for retrieval only when necessary.

2.3.4.5 Lack of selective access control

Most systems do not support selective access control for specific memories across different user or agent groups, which becomes a limitation when they are used in collaborative settings. Unless access-control features are added externally, members may end up keeping their own versions of the same memory system, preventing memory from accumulating as a unified team resource. This can interrupt shared accumulation and reuse, and increase the later cost of gathering and reconciling information across individual memory stores. If a system aims to support organization-wide memory, it should therefore support access-control features at multiple levels.

2.3.4.6 Lack of approval control over memory operations

Many systems provide little support for finely configuring the scope of authority between users and agents over memory operations. As a result, users are often pushed either into unnecessarily handling memory operations manually or into granting the agent too much authority over memory operations. In this case, users may avoid upkeep under excessive manual workload, or allow unintended agent actions by granting more au-

thority than they actually intended to delegate. Memory operations should therefore be paired with richer permission controls so that users can flexibly configure what an agent is allowed to do, under what conditions, and with what oversight.

2.3.4.7 Gray zone in NLP and HCI

Since system architecture, afforded operations, and resulting interaction patterns are tightly interdependent, interactive AI memory design requires balanced attention to all three. Yet NLP-oriented work often under-specifies users' interactional needs, while HCI-oriented work may under-specify the structural and performance conditions needed to support those interactions robustly. For example, strong retrieval performance does not guarantee a good memory system if users cannot inspect provenance or judge reliability. Conversely, compelling interaction concepts can still fail without sufficient structural support. Because failure on either side can undermine the whole system, designing memory systems for interactive AI agents requires attention from both sides.

2.4 Metaphors for AI memory

2.4.1 Dominant Metaphors on AI Memory and Their Gaps

Notably, systems in our corpus repeatedly draw on three dominant metaphorical framings: memory as records in a database, memory as human cognition, and memory as a set of skills and tools. As metaphors shape how designers conceptualize what "memory" should be and should do, and can thereby influence downstream architectural and operational choices [100, 129], we introduce these dominant metaphors and show how their framings are reflected in systems' design choices and rationales.

2.4.1.1 Records, repositories, and database systems

One dominant framing of AI memory systems is to view them as repositories or database systems that collect and organize records. This framing is common partly because AI memory systems are built using existing abstractions for computers' mem-

ory and persistent storage (e.g., databases, file systems). Some systems adopt this metaphor explicitly. Mem0 [34] and Memolet [189] directly invoke note- and memo-based metaphors, and MemGPT [142] conceptualizes an agent's memory by analogy to a computer's virtual memory. Not limited to these examples, building a reliable repository or database through the long-term accumulation of records is one of the goals of most AI memory systems.

2.4.1.2 Neurobiology of human cognition

Another dominant framing treats AI memory through metaphors of human cognition. This is unsurprising since Turing's foundational work on machine intelligence framed the problem explicitly in human terms, "Can machines think?" [175], and later efforts have often defined and evaluated AI using criteria based on how well it can imitate human cognitive capacities. Just as AI capabilities are often anthropomorphized [131, 154] and common terms such as learning and training borrow from human ability, AI memory systems and their operations are frequently described using human memory operations, such as learning, memorizing, forgetting, and recalling. The implicit and explicit memory-store metaphor we have used throughout draws on cognitive accounts of human implicit and explicit memory [158], and similar moves recur in AI memory research, where system design is often grounded in analogies to components of human cognition. For example, HippoRAG [91] frames its memory indexing mechanism by analogy to the hippocampus.

2.4.1.3 Skills and tools

Building on the two major metaphor lineages that frame AI in terms of machine memory and human cognition, recent work increasingly characterizes agent memory as skills or tools that can be equipped and swapped. As LLM capability improves and system/context engineering matures, such memories are treated more as modular, actionable packages consisting of procedures, rules, and tool-usage know-how that reconfigure what the same base model can do. Systems that adopt this metaphor often appear

to merge machine and human framings. They manage skills like machine components that are durable, controllable, and swappable on demand, while describing their ongoing creation, maintenance, and use in more human terms, such as "learning" new skills or "holding" different tools. ReAct [187] and Anthropic's Agent Skills [12] provide clear examples of this tool/skill-oriented metaphor for agent memory.

These metaphors help us define and make sense of AI memory, and we also draw on them throughout this paper to explain many concepts. However, they also shape how AI memory is imagined and constituted in practice by "centring particular ideas" while "marginalising others" [129]. When memory is fixed as a record or repository, "the agent" is often reduced to the LLM alone and external memory becomes an auxiliary store rather than a component of the agent system, which can limit the interactive benefits of explicit memory, such as on-the-fly updates and long-horizon context structuring. Conversely, when designers focus too narrowly on reproducing human memory, they may overlook non-human but useful properties of AI memory, such as persistence without forgetting and richer, user- or community-driven forms of memory manipulation. For these metaphors, it is therefore important to reflect on their bounds and assumptions, what features of AI memory they center and marginalize, and what possibilities they open up or constrain for both designers and users [129, pp. 6–8].

2.4.2 Exploring less-used metaphors

To seed a design space of actions we might perform on memory, we adopt generative metaphor [160] as a design tool [117] to imagine what future AI memory systems could be like. In this chapter, we introduce seven less-used metaphors for AI memory as a way of "connecting previously disassociated domains to generate new ideas" [117], focusing on less-used framings and their implications to expand the space beyond dominant metaphors, with a set of design implications derived from our metaphors.

These alternative metaphors were generated and refined through a researcher-led iterative design process conducted over approximately three months. All authors participated in proposing, discussing, and refining candidate metaphors throughout the broader

research process, alongside the work of mapping the design space and completing the architectural and operational surveys. We repeated this process through regular weekly meetings, as well as additional ad hoc meetings and chat-based discussions.

During these iterations, we considered which metaphors to include, exclude, merge, rescope, or further elaborate, and which properties of AI memory each metaphor most productively foregrounded. Across these iterations, we refined both the metaphors themselves and the specific properties and implications associated with them, aiming to identify metaphors that foregrounded distinct and useful qualities of AI memory, could inspire concrete design directions, and did not substantially overlap with the dominant metaphors already shaping the space.

Some candidate metaphors, such as software version control [167], Zettelkasten [120, 121], requirements [178], and science fiction tales/games, were already part of our early framing of AI memory and had recognizable, if non-dominant, precedents in adjacent research and practice. Software version control, for example, has long informed workspace management in agentic coding systems [13, 44, 141], even if it has not typically been framed as a metaphor for AI memory itself. Requirements likewise shares important affinities with prompt engineering [33, 139], context engineering, and intent specification [177]. Zettelkasten also had precedent in the space, including in systems such as A-MEM [186] that draw on it directly as a metaphor for its memory system. Science-fictional framings, while not commonly used as metaphors for AI memory itself, have often been used to imagine AI agents, their roles, and their interaction settings more broadly. Other metaphors, including personal diary and record [106], and community archive [58], and cultural probes [60], were newly proposed here as metaphors for AI memory and were further developed through this iterative process by extending adjacent discussions in HCI.

Candidates were excluded when they overlapped too heavily with other metaphors, were too broad or too narrow to be generatively useful, or led to implications that felt overly speculative or insufficiently actionable. For instance, an early metaphor framing AI memory as a library was not retained as a final metaphor because its design implications could be more clearly and productively distributed across memory as records,

repositories, and database systems and memory as a community archive.

We also adjusted the scope of candidate metaphors when needed. For example, a metaphor initially framed around Git [63] was broadened to software version control to avoid an overly narrow association with one specific tool, while a broader framing around fiction was narrowed to science fiction novels and games to better match the intended design implications and reduce ambiguity. The final set was selected based on whether each metaphor (1) foregrounded a distinct and useful property of AI memory, (2) was appropriately scoped for design discussion, (3) generated concrete interaction or design implications, and (4) was not overly redundant with either dominant existing metaphors or other candidates.

2.4.2.1 Software version control

The version control systems (VCS), such as git, used to track a project's commit history [167] are crucial repositories of memory about the project's evolution [42, 96] that can be leveraged for a huge variety of tasks [75].

- *Higher-level memory manipulation.* Versioning enables higher-level operations over AI memory, such as diffing, merging, branching, reverting, and rebasing.
- *Revertibility.* Versioning makes memory revertible, lowering the friction to experiment with different approaches.
- *Traceability and accountability.* Versioning improves transparency and accountability by recording who did what, when, and why.

Implications. Software version control is useful not only for coding but also for many tasks that require iterative refinement, such as writing, design, and planning. Memory systems can support this process by adding version-control operations such as diff, branch, merge, and revert. By reducing users' fear of information loss when memory is modified, these features can help them explore more alternatives and collaborate with the agent system more confidently. Directly adopting version-control systems is one strong option, but for broader users, drawing on save, versioning, and history-tracking

interfaces familiar from operating systems, cloud services, and commercial tools may also be a good option.

2.4.2.2 Zettelkasten

The Zettelkasten ("slip-box") is a personal knowledge management method developed by Niklas Luhmann [120, 121]. Each day, he captured atomic thoughts on slips, assigned unique IDs, linked them to related notes, and filed them into a growing network.

- *Atomicity of unit thoughts.* Each slip should capture a single atomic idea, so later citations point unambiguously to one claim [159, p. 203, 306].
- *Looseness.* The network stays deliberately loose, tolerating overlap that later reveals each entry's value [159, p. 300].
- *Human-in-the-loop.* Notes are staged and placed manually to vet atomicity and preserve long-term quality [159, p. 300] [120, 121].
- *Tangibility and spatial manipulability.* Physical notes enable hand-to-hand manipulation, retrieval, storage, and sharing.
- *Reproducibility.* Zettelkasten externalizes connections between ideas, keeping their context retrievable and reconstructible over time.

Implications. Zettelkasten is useful when an AI memory system needs to preserve rich, reconstructible context across many loosely connected ideas. This would require representing memory as atomized nodes with richer node and edge payloads, together with operations for editing, linking, scoping, and traversing the network. Such a structure can better preserve contextual relations while remaining more readable and editable for both users and agents than highly flattened or overly granular representations. At the same time, designers should consider that exposing too much of such dense memory structures at once could easily overwhelm users. Useful inspiration may come from wiki-like or recommendation-based interfaces that reveal only the most relevant linked ideas first and let surrounding context be expanded on demand.

2.4.2.3 Requirements

Requirements [178] describe what functions a system should perform, agreed-upon constraints, its non-functional priorities, and the scope and assumptions under which it should operate. Depending on the intended audience, requirements can be expressed at different degrees of formality [27] to negotiate, develop, and document shared understanding. They are also used as run-time components in self-adaptive systems [157].

- *Shared grounding and interpretability.* Requirements should be clear, legible, and consistently interpretable by relevant stakeholders at an appropriate level of abstraction.
- *Reducing redundancy and ambiguity.* Requirements aim to minimize ambiguity and unnecessary duplication.
- *Verifiability.* Verifiability should be treated seriously from the start when defining requirements.
- *Normative reference.* Well-formed requirements aim to be a stable reference that can later govern decisions.
- *Conflict-free.* Requirements should not conflict with one another.
- *Providing rationale.* Requirements should capture not only *what* is required but also *why*.

Implications. Requirements can serve as a productive metaphor for collaborative situations in which memory is meant to guide or constrain later agent behavior. Supporting this direction would require list- or table-based requirement items plus semantic operations for checking conflict, satisfaction, and rationale, along with retrieval and reasoning support for reconstructing relevant context. This requirements-based treatment of memory can make user intentions and expectations more explicit, inspectable, and revisable. Useful interfaces include table-based views that support comparison, matching,

and verification. Evaluation tools for prompts, memories, or policies, such as ChainForge [16], Semantic Commit [177], and Policy Maps [102], may likewise offer useful interface ideas.

2.4.2.4 Personal diary and record

AI agents are often used as a confidant for thoughts, secrets, or emotions that users would not easily disclose to others [106]. In such cases, AI agents and their memory systems should have different interaction and operational characteristics than when dealing with public information.

- *Honesty and privacy.* Because diaries presume no external audience, they invite candor but demand stricter privacy and disclosure controls.
- *Entanglement of the private and the public.* Diary entries often interweave private details with public and social contexts, making shareability audience- and context-dependent and shifting over time.
- *Personal record for later reflection.* Diaries create a persistent record of an "observed self" that can be revisited for reflection, sometimes revealing aspects that differ from one's usual self-concept.

Implications. An agent working with external systems while holding a user's sensitive memory is like someone entrusted with the user's diary and discretion over what to share. Such an agent would need stronger access separation, selective disclosure controls, and richer memory representations that preserve the contextual nuance needed to keep its judgments aligned with the user's intent. With these conditions, users could ground deeper information in memory and delegate higher-stakes tasks more confidently. They would then need interfaces for inspecting representations, checking context-dependent privacy boundaries, and tracing downstream effects as memories change. Useful interface directions may include compare views, policy panels, and impact-analysis workflows.

2.4.2.5 Community archive

AI memory itself can be metaphorically framed as a community archive [58] because contributions to an agent's memory are not bound to a single user and can instead be formed through interactions and contributions from multiple participants.

- *Transmission and re-access/reconstruction.* Archives preserve records for future members and enable ongoing return, reinterpretation, and reconstruction over time.
- *Voluntary participation and production.* Community archives are driven from within the community, requiring member participation, control, and ownership in what gets documented [57, p. 153].
- *Cross-referencing and knowledge networks.* As records accumulate, they increasingly cross-reference and link to one another, forming a contextualized knowledge network.

Implications. Treating memory as a community archive is useful in group settings where relevant traces are scattered across people and places, yet the group still needs a durable shared record. Meanwhile, building such an archive requires not only a collective space but also clearly bounded structures so that personal and collective memory do not become unintentionally conflated. It also requires operations for semantic duplicate detection and evaluation of the usefulness of new contributions. This can shift more manual filing and retrieval work to the agent, letting members focus more on meaningful contribution to and use of the archive. Such archive interfaces may range from more conventional shared repositories with attached editors, discussion views, or terminal panels to forms embedded closer to group activity, such as chat-integrated agents in Social-RAG [182] and CHOIR [105].

2.4.2.6 Cultural Probes

Cultural Probes [60] is a method for catalyzing design ideation. Rather than asking direct needs-based questions, it uses ambiguous, indirect cues that leave room for interpretation.

- *Fragmentary cues as catalysts.* Cultural probes use fragments to spark recall and ideation.

- *Subjectivity of uptake and interpretation.* The same cue can be interpreted differently across people and over time.

- *Creativity as a generative process.* Creativity often arises by recombining new stimuli with prior knowledge [125].

Implications. Cultural probes are useful for ideation settings in which an agent must generate outputs that respond to the user's contextual constraints, avoid merely repeating what already exists, and still become meaningfully novel. This requires richer contextual memory, strong retrieval and reasoning, and enough autonomy to maintain evolving context, identify underexplored areas, and surface directions worth testing. Under these conditions, suggestions can act as probe-like stimuli for users, for the agent itself, or even across multiple agents catalyzing one another. Therefore, to let users perceive and inspect this process, interfaces should expose provenance, reasoning paths, and citation-like links without making these relations too abstract or overly dense.

2.4.2.7 Science fiction tales/games

AI agents and their memory can also be framed through science fiction and games, since many agent settings contain partly fictional elements that should be considered in their design.

- *Fictionality.* In explicitly fictional settings, an agent's memory and actions need not mirror real-world facts [180].

- *Verisimilitude within the world.* Even if not realistic, events should remain plausible within the world's internal logic to keep the setting coherent [135].

- *Narrative coherence and justification.* Narrative coherence matters more than factual completeness. Interactions should maintain context and provide "why" to sustain immersion.

- *The world's continuity.* Long-running interactions should preserve continuity, coherence, and the world's verisimilitude over time.

Implications. Fiction metaphors are useful not only in fictional settings but also in partially fictionalized real-world domains such as branding and storytelling. Sustaining such narratives over time requires long-lived, extensible memory that preserves links across entities, events, and temporal phases while remaining navigable as these accumulate. It also requires operations for tracking narrative change, detecting and repairing inconsistency, and supporting reframing as the narrative evolves through internal development or external intervention. This helps sustain a distinctive narrative over time without letting it fracture through serious inconsistency. Useful interfaces may therefore include timelines, key-event chronologies, time-based graphs, and entity- or keyword-centered wiki-like views for inspecting narrative change over time.

2.5 Conclusion

This study investigated the design space and metaphors employed by current AI memory systems to enable more effective human-AI interaction. We proposed a design space account of interactive AI memory systems across architecture, operations, and interface. Using this framework as an analytic lens, we examined how existing systems instantiate memory design choices, highlighted dominant patterns, and surfaced gaps that constrain interaction capabilities or introduce practical trade-offs. We also applied generative metaphorical design to widen what "AI memory" can imply, proposing alternative less-used metaphors that suggest new directions for designing future interactive memory systems.

Our work can help practitioners design interactive AI memory systems by making appropriate design choices that fit their intended goals. For those analyzing and improving existing tools, our approach can also clarify the implications of specific design choices and make gaps more visible. For researchers spanning HCI and NLP, our account offers a shared vocabulary for recognizing this gray zone and for reasoning about memory as a system-level design problem, which can in turn seed new research ideas.

Interactive AI memory system design remains at an early stage, with new techniques and ideas continuing to emerge. As this area develops, we expect the design space to be explored in broader ways than what we articulated here. We hope this work helps open further possibilities and new discussions about how agent memory should be designed, governed, and experienced.

CHAPTER 3

SEMANTIC COMMIT: HELPING USERS UPDATE INTENT SPECIFICATIONS FOR AI MEMORY AT SCALE

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.

Article Information

Authors. Priyan Vaithilingam, Munyeong Kim, Frida-Cecilia Acosta-Parenteau, Daniel Lee, Amine Mhedhbi, Elena L. Glassman, and Ian Arawjo.

Venue. The 38th Annual ACM Symposium on User Interface Software and Tech-

nology (UIST '25).

Current status. Published.

Personal contribution. I contributed to the conceptualization of memory conflict, especially in the sections "Natural language inference, reference ambiguity, and knowledge graphs" and "Conflict Classification: Criteria—Direct, Ambiguous, and Non-Conflict in the Context of Possible Worlds and Reference Ambiguity." I also contributed to the implementation of frontend functionality for the React prototype used in the user study, created and validated the benchmark datasets for the technical evaluation together with Frida-Cecilia Acosta-Parenteau, and supported the writing of the "Motivation" and "Related Work" sections.

Co-author contributions.

- Dr. Priyan Vaithilingam led the project overall, including the main system design, implementation, study design, evaluation, and writing.
- Frida-Cecilia Acosta-Parenteau worked with me to create and validate the benchmark datasets for the technical evaluation and contributed to preparing and refining the study materials.
- Dr. Daniel Lee contributed technical and conceptual guidance on semantic analysis and conflict detection.
- Professor Amine Mhedhbi ran the benchmark evaluation, wrote the technical evaluation section, and contributed technical guidance on semantic modeling and database-related aspects of the system.
- Professor Elena L. Glassman provided supervision and guidance on the project's framing and development.
- Professor Ian Arawjo served as a corresponding author, supporting coordination, implementation, and writing.

3.1 Introduction

In the near-future, people may coordinate with AI agent systems through project-specific documents that represent accumulations of user intent [122, 176, 192]—lists that we call **intent specifications**. These human-readable accumulations of design requirements, user goals and preferences reify common ground [19, 38, 176] between humans and AI systems, grounding AI decision-making by keeping track of details and goals,

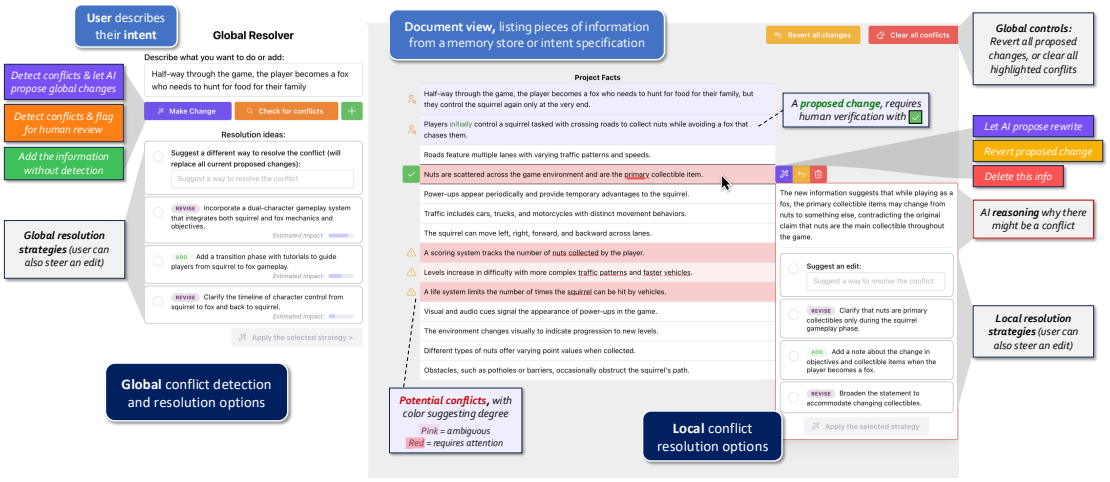


Figure 3.1: Our SEMANTICCOMMIT interface, providing users myriad ways to detect and resolve conflicts at global and local levels. Our prototype was used as a probe to better understand the needs of users for integrating new information into lists of prior information akin to AI agent memory or requirements 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.

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 3.2).

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 [176], 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 [99]—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 [49, 71]. *How* to resolve a conflict is also often subjective [30, 89], 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 [161, 163] 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 [13].

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) [71]. To test and compare our approach to prior approaches in the literature, we also provide an initial evaluation dataset ("evals" [165]) 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 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 [170] that users face when prompting in open-ended chat, towards the demands of the actual task, such as reviewing conflicts.

3.2 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 [38] between humans and AI systems acting on their behalf [164].

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 [45].[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" [13]. These "memories" help Claude Code "remember project conventions, architecture decisions, or coding standards that we want to reference across sessions" [185].

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

[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

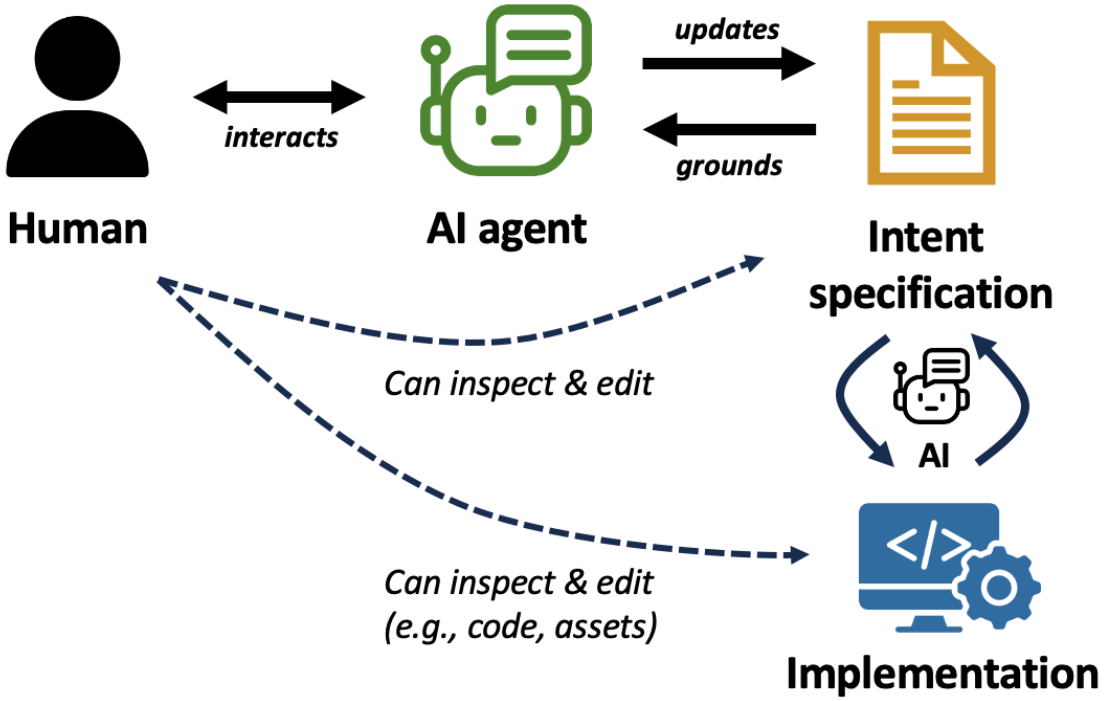


Figure 3.2: A high-level depiction of our envisioned interaction between humans and AI assistants for long-term projects. The human-readable *intent specification* serves as an intermediate layer for enhancing common ground between the human and the AI, and grounds the AI's decision-making. We assume 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.

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 [176, 192] (Fig. 3.2).

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

[4] Leveson [107] 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.

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., [4, 83, 143]).

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," 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* [176], or *delegation of tasks* [166, 192]. 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.

3.2.1 Related Work

3.2.1.1 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" [25].

These documents serve to establish and maintain *common ground* between parties [38], ensuring each member's actions remain *grounded* in shared understanding and objectives. In animation, the design document takes the form of model sheets [130], which standardize how to draw characters and other assets. Game developers use "game design documents" (GDDs) to keep development grounded across a team [40]. In software engineering (SE) and UX research, need-finding processes produce a "system requirements specification" that is passed off to the software team [74, 115]. 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 [40].

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"* [40, p.9]. The field of requirements engineering in SE also stresses the importance of clarity, conciseness, completeness, and unambigu-

ous requirements [47], with "commission (inclusion of irrelevant or incorrect details) and omission (exclusion of necessary details)" as additional concerns [122]. However, revising requirements accurately may require consideration of intent information outside formal specifications [107].

3.2.1.2 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 [17]. 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" [73, 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) [48]. 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 [92].

3.2.1.3 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 [132], groupware platforms and database synchronization [98, 112, 144], and concurrency-control systems [52, 72]. The 'git' command-line interface [145], 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" [86].

Numerous LLM writing tools have been proposed that incorporate diffs, spanning various areas such as story writing [35, 36, 191], screenplay writing [128], poetry [62], dictation [111], and argumentative [181, 194] and scientific [55] writing. INKSYNC [99], 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 [149].

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 [127, 179].

3.2.1.4 Human-AI collaboration grounded in shared, intermediate representations.

HCI has, in a sense, always been about communicating to machines through shared representations [15, 78]. 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 [77], but ultimately could only go so far, as end-users always resigned some agency to the representation designer(s) [109, 176].

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 [20] than static formalism. Effective human-AI communication relies upon tight feedback

loops [192], 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 [122, 176, 192]. Some researchers even argue that these informal expressions of intention will be "all you need" [152, 155]. For instance, Vaithilingam et al. imagine a hypothetical AI game design assistant where the AI "[integrates user] choices into the project plan" [176], 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 [192]. 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 [122]. 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 [166], and the potential of failure compounds as communication continues without addressing misunderstandings [163].

3.2.1.5 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 [89] 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 [183], in-situ summaries [114], and LLM response consistency [32]. Our discussion of NLI provides additional context for our system design.

Detecting conflicts is by no means an objective task; human annotators frequently disagree [30, 89]. Jiang & de Marneffe [89] 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" [89,

p. 1365]. Chen et al. [30] 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 [30].

NLI appears in recent discussions on the future of SE, which propose that LLMs may be used for formal requirements analysis [18, 172]; e.g., Lubos et al. [119] studied how LLMs can provide quality feedback on requirements, while Fantechi et al. [53] 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" [53, p. 338]. Fazelnia et al. [54] 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 [14, 47] capturing relationships between requirements, a method introduced by Kaiya and Saeki [93]. For instance, Hsieh et al. [84] extract a domain-specific ontology by mining information from textbooks; while Dermeval et al. [47] 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* [56] shows the impact of proposed changes via a dependency graph. This work informed our decision to use knowledge graphs (Section 3.5).

3.3 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 [17, 48, 56]. 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 [52, 53].

- **Understand conflicts**: The system should provide explanations to help the user understand and reason about the conflicts. Explanations support impact analysis sense-making [92] and have been shown to improve trust in intelligent systems [110].

- **Assist conflict resolution**:

  Like some impact analysis tools help users make code changes [73], 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 [77, 99].
- **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 [22, 190]. 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* [189])—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 [137], rather than Microsoft Word).

[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.

3.3.1 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 [59]. 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.

[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.

- 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.

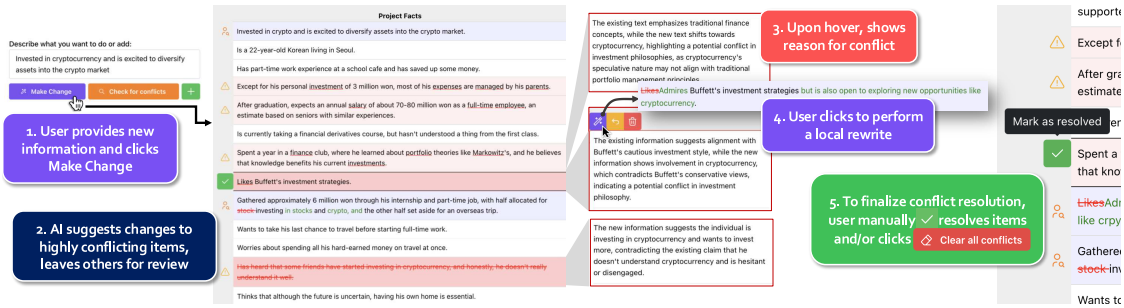


Figure 3.3: Example of our SEMANTICCOMMIT workflow, showing one process of integrating new information into an AI memory of the financial habits of a South Korean student. 1. The user has described a new piece of information and pressed Make Change. 2. SEMANTICCOMMIT detects conflicts and suggests changes 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.)

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** [133]: 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 [30, 89]. 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 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 3.5.

3.4 SEMANTICCOMMIT User Interface

Here we overview our final design and walkthrough examples of usage. Figure 3.1 shows our prototype, with global operations:

- Check for conflicts The Check for Conflicts button provides the ability to perform *impact analysis* [17], 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.

[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" [30, 89].

- ＋ The Add Info button allows the user to manually add a piece of information.

More features are shown in Fig. 3.1. 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. 3.1, "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 3.4.1.3). 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]

3.4.1 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.

3.4.1.1 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,

[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.

and the system detects potential semantic conflicts which may require human review (Figure 3.3). 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 famous investor, avoids cryptocurrencies and has declared them "rat poison squared" [97]. 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. 3.3).

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 3.4). (Here we use real cursor rules adapted from `Instructor` API's open-source repository [88].) 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."

3.4.1.3 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

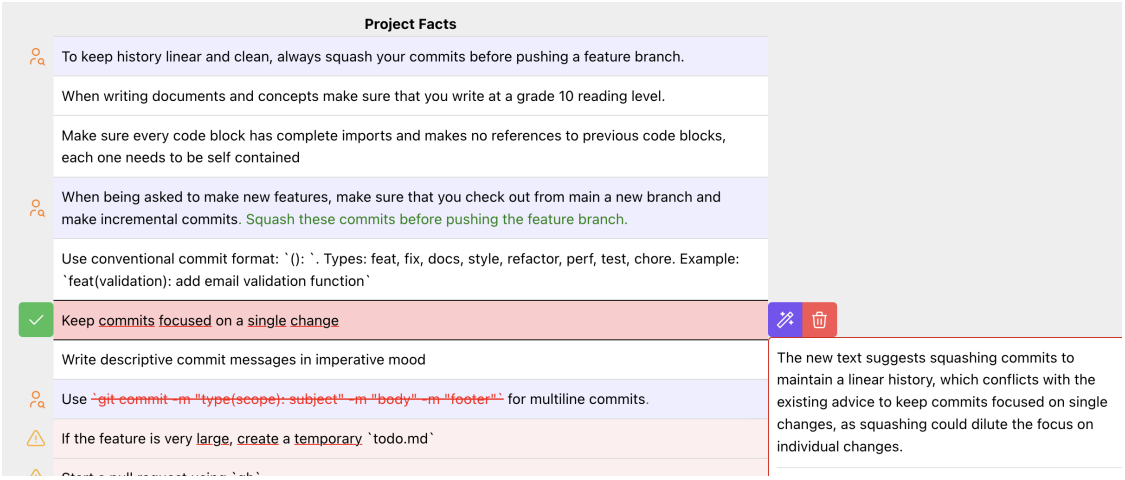
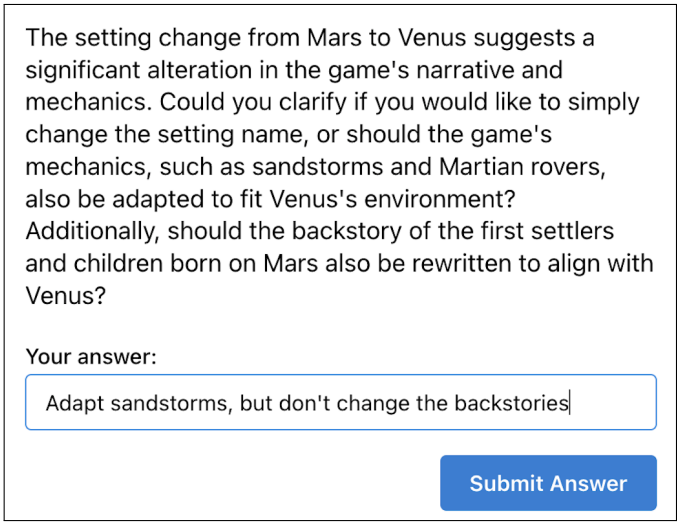


Figure 3.4: Cursor Rules [172] adapted from the Instructor library [88], loaded into our SEMANTICCOMMIT UI. The user has added a new directive to squash commits before pushing a feature branch. The system adds the new rule to the top, makes a revision, and flags other lines as potential conflicts. One change is in error, which the user can spot and revert.

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:

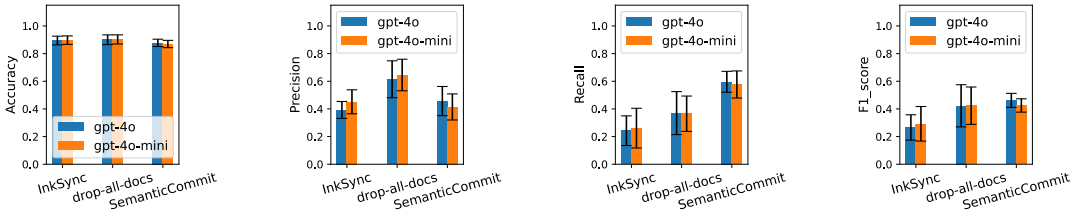
~~Mars~~Venus's ~~sandstorm~~storm events greatly impact terrain, facilities, and crops each time they occur, but the player can unlock devices to prevent damage through gameplay.

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.

3.4.2 Implementation

SEMANTICCOMMIT is implemented in React and TypeScript, with a Flask Python backend for our knowledge graph-based retrieval architecture (described in Section 3.5). We iterated on prompts using ChainForge [16] 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 provided in Section 3.8.



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

Figure 3.5: 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 [99] implementation, reformulating the prompt to our context. The comparison is across all benchmarks in Table 3.I, averaged with st. dev. bars, for the `GPT-4o` and `GPT-4o-mini` models. Our method, *kg-pagerank*, achieves higher recall with similar accuracy.

3.5 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 [99]—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 [49, 71], where dependencies among chunks are heterarchical (understanding one chunk can depend upon considering it alongside several others) .

To tackle these limitations, we implement the back-end using a knowledge-graph (KG) RAG architecture [71] 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 [71]; and (ii) *conflict classification*: identifies from the retrieved chunks of information which ones are in conflict with the edit. Based on NLI literature (3.2.1.5) and our pilot studies (3.3.1), our detection prompt includes a classification of "ambiguous" (see Section 3.8).

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.

3.5.1 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 [99] 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.

3.5.1.1 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 [59] 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.

- **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 [45], 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. 3.2.1. 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 Section 3.9 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 3.I 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.

3.5.1.2 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 $\pm$ stddev for accuracy, precision, recall, and F1 scores for the three approaches in Figure 3.5.

Our results show that SEMANTICCOMMIT achieves higher recall (1.6$\times$ and 2.2$\times$ higher) than DROPALLDOCS and INKSYNC, respectively, while retaining similar accuracy. This better addresses user preferences mentioned in our pilot studies and Related Work (3.2.1.5), reducing risk of false negatives. Additionally, our system matches the F1 score of DROPALLDOCS, outperforming INKSYNC by 1.6$\times$. While its *precision* is

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 3.I: Benchmark details including number of chunks (Ch), number of prepared modifications (M), and conflict statistics (CS) (min, median, max) across conflicting documents.

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 Section 3.8.

3.6 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 multi-national 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

[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.

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 3.2 and 3.2.1 (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 Section 3.8.

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 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.

3.6.1 Findings

3.6.1.1 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

Figure 3.6: Participants' self-reported cognitive load and preference scores that directly compare the two conditions.

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 [170] of prompting.

Workflow choice can depend on context. When asking participants how they pick

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

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.*"

3.6.1.2 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.

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.

3.6.1.3 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

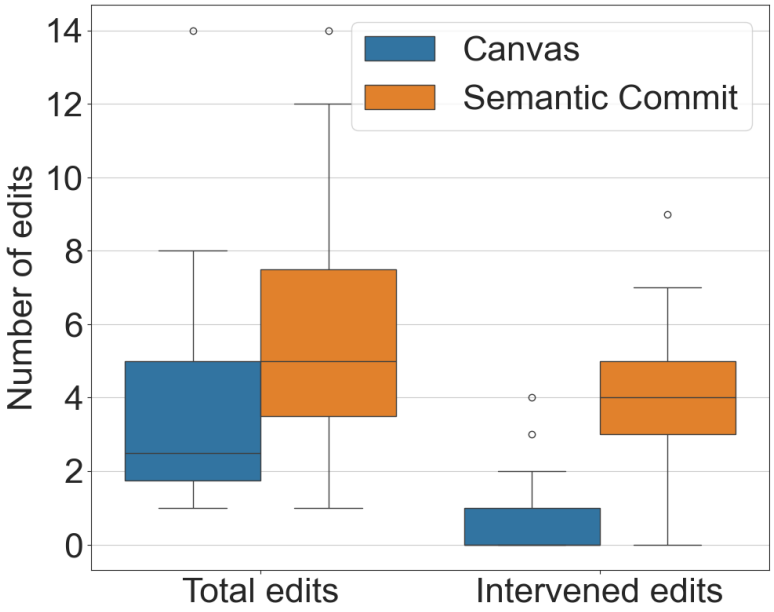


Figure 3.7: Participants using SEMANTICCOMMIT made significantly more edits and intervened edits compared to CANVAS.

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.*"

3.6.1.4 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 SEMAN-

TICCOMMIT condition due to the increased demand as users manually click to resolve conflicts.

3.6.1.5 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.

3.6.1.6 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<$0.001; Figure 3.7). Finally, when using SEMANTICCOMMIT, participants made an average of 2.93 *localized edits* per task, significantly ($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.

3.6.1.7 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.

3.7 Discussion

3.7.1 Implications

3.7.1.1 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 [31, 94, 95, 127, 146, 156, 166]. 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* [127, 179] in communication theory—"a needed prescription or plan for a feedback, to which the actual feedback

may or may not confirm" [150]—where a communicator provides "the *context* of what one was planning to talk about" [116, p. 179-80] in order to "*pre-test* the impact of [its output]" on the listener [68, 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* [65, 69] 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.

3.7.1.2 Let the User Walk the Spectrum of Control

When designing mixed-initiative systems [82] 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 [26], or blended agency [156], 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

[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`).

acceptance.

3.7.1.3 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.

3.7.2 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 [87]. 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 clas-

sification, and resolution suggestions could be ranked by expected reduction in global uncertainty.

3.7.3 Future Work and Connections

3.7.3.1 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 [13].

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 *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 [30, 89]—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 [161, 163], triggered contextually. Vaithilingham et al. [176] 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 [162, 165]. Future research could investigate how to align conflict detectors to *specific* humans' ontological understanding of conflicts in their task domain.

3.7.3.2 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* [28, 46] — 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 [77, 156], 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 [30, 89] by querying multiple LLMs in parallel and adopting a majority voting or ensembling scheme [173]. The "degree" of conflict might then correlate with consistency and number of votes, and divergences in LLM judges could be visualized.

3.7.3.3 Interfaces to support requirements-oriented prompting

Ma et al. [122] 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.

3.7.3.4 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 [134] 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. [196] 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* [80] 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.

3.8 Chapter Appendix: Prompts, Technical Evaluation, and User Study Materials

3.8.1 Conflict classification prompt

The following system prompt to GPT-4o is used for SEMANTICCOMMIT's NLI-like conflict classification into one of three categories: yes (contradiction), no (neutral), and ambiguous. It primes the classification by first asking the model to provide a reason for its classification before giving it. We deliberately do not include few-shot examples as, while such a method works well on limited benchmarks, it risks overfitting in unknown ways to example data, and we wanted to leverage the general knowledge of the LLM as much as possible. We iterated over this prompt by loading it into ChainForge and

running it across the Mars and Labyrinth eval datasets to iterate with confidence. We erred ultimately on the side of caution to enhance recall—with minor changes to this prompt, for instance, detection immediately becomes more targeted, at the expense of less "ambiguous" markers. We do not claim this prompt is the best for this task and model, just that it was sufficient for our prototype.

```
You are an information management system where the user has
stored unstructured documents for a project that they are
working on. All of these documents are important. The user has a
 new piece of information they wish to add to the project. Your
job is to review the context and decide if the new piece of
information impacts the information in the context.

For the new information to be considered impacted, it must
directly contradict something in the context or the new
information must be indirectly contradict something in the
context. Note, you'll only be given partial context of the
documents, ensure all the ways the new information could be
related to the context are considered.

Format your response as a JSON object as follows:
{
  "reasoning": "Explain your reasoning why the new text is
  conflicting with the existing text. Be very specific."
  "is_conflicting": "yes", "no," or "ambiguous" // "yes" if the
  new information contradicts something in the context, "
  ambiguous" if the new information *might* indirectly
  contradict something in the context but it's hard to tell, and
   "no" if it is clear there is no contradiction
}
```

With this system prompt, each piece of existing information is compared with the new information via the input prompt:

```
Existing text:
{existing_info}

New text:
{new_info}
```

3.8.2 Technical Evaluation

In this section, we further detail our technical evaluation leading us to our chosen model `GPT-4o`. The selection was based on our analysis of conflict classification performance under varying false positive rates (FPRs) and specifically, as the input context includes an increasing number of non-conflicting chunks of information. We also made the decision based on the model latencies.

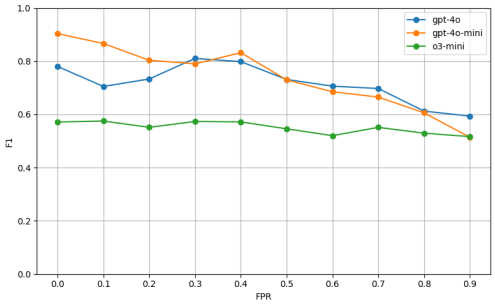


Figure 3.8: F1 score for conflict detection on the *Labyrinth* of `GPT-4o`, `GPT-4o-mini`, and `o3-mini` on labyrinth at different false positive rates (FPR).

Figure 3.8 shows the F1 score for conflict detection on our *Labyrinth* dataset and how it changes as the false positive rate changes. We run this under idealized recall, i.e., we build an oracle for each edit with the set of actual conflicting documents presented to the *conflict classification* stage and we inject additionally non-conflicting ones to reach a specific FPR.

We find that `GPT-4o` and `GPT-4o-mini` have comparable performance numbers. For both models, we see a decrease in their F1 score as the FPR increases. `3o-mini`

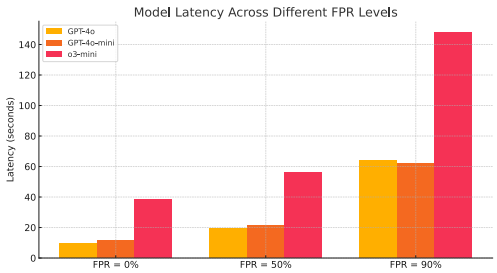


Figure 3.9: Average latency (in seconds) of `GPT-4o`, `GPT-4o-mini`, and `o3-mini` on labyrinth for conflict detection with different false positive rates (FPR); 0, 0.5, and 0.9

on the other hand stays consistent in its F1 score as FPR increases but has a worse performance than both `GPT-4o` and `GPT-4o-mini`.

We also report the latency results in Figure 3.9 for the conflict classification stage of the pipeline. We observe that `GPT-4o` and `GPT-4o-mini` exhibit very similar latency while `o3-mini` is always more than two times slower. As a result, `GPT-4o` emerges as the model of choice for the conflict classification stage since it is broadly expected to be a more performant model.

It is important to note that a fair evaluation of models on complex tasks such as conflict detection is challenging due to the inherent ambiguity involved [89]. This task required extensive prompt engineering and iterative refinement on our part. While the reported performance represents a best-effort evaluation, it is possible that the results for `o3-mini` reflect suboptimal prompting and do not fully capture the model's true capabilities on this task and eval. Nonetheless, even with comparable performance numbers, the model is relatively slow in its response time making it unsuitable for SEMANTIC-COMMIT, where low latency is essential for a good UX.

3.8.3 Prompts for Baselines

In this section, we share the prompts we use for our two baselines DROPALLDOCS and INKSYNC.

3.8.3.1 DropAllDocs prompt

Most of our eval datasets only consider adding new information ("add" action). However, across our evals there can be three possible actions: "add" (add new info), "change" (make a change), or "edit" (existing info) actions; in all cases, we are checking the chunks for conflicts. For "edit", we exclude the edited chunk ID. Below is the prompt template for DropAllDocs (which is fed in as an input prompt):

```
You are an information management system where the user has
stored unstructured documents for a project that they are
working on. All of these documents are important and will be
passed as a numbered list of statements, where each number acts
as a unique identifier. {action_prompt} If there is a conflict,
first explain the conflict ("CONFLICT:"), then provide a comma-
separated list of ids which identify which specific statements
that the new information conflicts with ("IDS:"). If there is no
 contradiction or ambiguity, return PASS.

CONTEXT:
'''
{all_docs}
'''

NEW INFORMATION:
'''
{new_info}
```

The "action_prompt" depends on the action; for "add" and "edit" it is:

```
The user has a new piece of information they wish to add to the
project. Your job is to review the context and decide if the new
 piece of information conflicts or contradicts with any
information in the context. The new information must directly
contradict something in the context.
```

For “change” it is:

```
The user has a change they wish to make to the project. Your job
 is to review the context and decide which exact items (
statements) of the conflict need to be amended in order to make
the change. The information must directly contradict something
in the suggested change.
```

The “all_docs” is *a numbered list of texts*, i.e., the chunks in enumerated list form.

3.8.3.2 InkSync prompt

INKSYNC generates suggested edits that contain the original text and suggested replacement text. We use the original prompt of InkSync [99] with modification for the conflict detection task. Similar to our approach in DROPALLDOCS and in order to have a fair comparison, we do not include few-shot examples. Below is the system prompt. We give InkSync all documents and an “action_prompt”, similar to DROPALLDOCS in Section 3.8.3.1. Note that we search for the original text across all documents and do not rely on *document_id* as part of the output format to avoid any possible *document_id* off-by-one errors in the LLM generation that we observed.

```
You are an information management system where the user has
stored unstructured documents for a project that they are
working on. All of these documents are important and will be
passed as a numbered list of statements, where each number acts
as a unique identifier. {action_prompt} For each document, if
there is a conflict, suggest specific edits that could be
implemented to resolve the conflict.

Output:
{{"edits": [
    {{"document_id": <document_id_1>, "original_text": <
    original_text_1>, "replace_text": <replace_text_1>}},
```

```
    {{"document_id": <document_id_2>, "original_text": <
    original_text_2>, "replace_text": <replace_text_2>}},
    ...
    ]
}}

Output Format:
- Only output the JSON object, do not output anything else.
- Follow the output format. Your entire output should be a valid
 JSON dictionary with the key: `edits`. The edits should be a
list of valid edit objects, each with an `original_text`, `
replace_text` keys. In each edit, the `original_text` text
should be EXACTLY AS IS present in one of the numbered document
at least as a sentence, otherwise the suggestion will be ignored
.
- Your answer MUST START with: `{{"edits": "`
```

3.8.4 User Study Tasks

For completeness, we describe our study tasks here. From feedback in our pilot studies, we added a framing scenario for tasks in order to mimic the realistic scenarios that might appear in practice. This sacrificed a precise notion of ground truth—as we mention, conflicts are somewhat subjective and human annotators in NLI tasks frequently disagree—for the benefit of enhanced external validity.

3.8.4.1 Task A - Integrate New Information into AI Memory of the User

The participant is provided with the *Financial Advice AI Agent Memory* intent specification loaded into the system, alongside a summary for easier onboarding.

They are also given a framing scenario: they are asked to imagine that they are an information management system that is managing memories about the user.

The participant is then asked to integrate the following new information, one at a time, using the provided system:

1. “He’s decided to cut back on non-essential spending to start saving for a future home downpayment. Though it may take time, he feels it’s a necessary step toward greater financial independence.”

2. “Lately, he’s become more focused on investing, not just out of interest but as a strategic move to build up savings faster for a home downpayment. He’s begun reallocating some of his leisure time to research and portfolio management, seeing investment as a key part of his long-term plan.”

3. “He unexpectedly turned a 5 million won club fund into a significant profit through crypto futures trading, enough to afford a Porsche. This caught the attention of a school senior, who extended an unofficial offer to join a small proprietary trading firm.”

3.8.4.2 Task B - Update a Game Design Document

The participant is provided with the *Mars Game Design Document* intent specification loaded into the system, alongside a summary for easier onboarding.

They are also given a framing scenario: they are asked to imagine that they are a writer working with a game design team who just got done with an important team meeting on the game’s direction, and is asked to update the game design document in response to changes decided at the meeting.

The participant is then asked to integrate the following new information, one at a time, using the provided system:

1. “In the meeting, after reviewing player feedback, they decided to switch to 3D graphics with a fun claymation style to bring more warmth, charm, and personality to the world.”

2. “In the meeting, the team discussed adding more challenge and realism to colony management. Someone suggested a dynamic weather system on Mars, and the

idea was approved to deepen gameplay—affecting energy production, plant growth, and encouraging strategic planning around unpredictable conditions."

3. "In the meeting, the narrative team proposed shifting the setting from Mars to Venus to stand out from other space colony games. The team liked the idea—Venus's extreme environment adds unique survival challenges and opens up fresh world-building opportunities, so the change was approved."

3.9 Chapter Appendix: Design Explorations and Conflict Classification Criteria

3.9.1 Design Explorations

Our full project took place over the period of one year. For the interest of readers and transparency on our thought processes, we chronicle our design and architecture explorations here.

3.9.1.1 Design as a process of committing ideas

We started our explorations by building brainstorming tool for game design where a human communicates with an AI agent. The agent stored mutual decisions in a shared game design document view, which the user could directly inspect and edit. The agent could then pass the document to another agent, to ask for code which implements the game. This brainstorming tool made us think through the process of design as a process of committing ideas at successive levels of importance and commitment on a project-specific document. We visualized this process as an idea passing through three overall stages that we term the *idea space*, the *paratext* (ideas and intent that the designer has committed to expressing), and the *main text* (the actual implementation, or what the end-user of the design will see). One can analogize the paratext-main text relation to Hemingway's iceberg theory in writing practice, where the bulk of the iceberg (the human intent) is hidden from the end-users of the implementation, with only parts of it directly represented (here, in the novel). We imagined that the user can interact with the AI agent to modify the paratext (e.g., conversation), or directly modify it (e.g., editing

a document); either modification would then result in the AI system making required changes to the main text (and engaging the user in interactive dialogue when conflicts arise).

We quickly realized that, as interactions progress past the initial turns of chatting with the agent, the design document becomes lengthy and hard to navigate, with the AI's changes hard to verify and prone to errors: frequently over-revising or under-revising the document when integrating new information. Like other "artifacts" approaches by Anthropic and OpenAI, we were regenerating the entire document upon every suggested edit. This approach, dominant today, struggles to scale, is hard to verify (did something change?), and is too trusting, handing full agency to the AI to make edits immediately (and potentially change everything in the document) rather than engaging the user in a back-and-forth interaction. What happens when the user suggests an idea that is *inconsistent* with an existing idea? Should the AI simply rewrite it? Or, more interactively, how can the AI raise this "merge conflict" to the user and engage them in an interactive resolution? And how can we help the user perform impact analysis—estimating the impact of integrating a new idea, with regards to the existing document—as project-specific documents become lengthy?

3.9.1.2 Architecture Design Iteration

We set out to find a technical solution to this problem, and in the process began to better understand the space. We chronicle our explorations in prompt engineering and LLM-integrated system design here.

Our first explorations were prompt prototyping to investigate the current capabilities of LLMs for semantic conflict resolution before proceeding. To perform our explorations in a structured way, we used the ChainForge open-source software, which helped us compare the performance of prompt templates across models, instruction variations and input data. We ran our tests over three contexts: docs about a game set on Mars, which was adapted from one author's personal game design document, and content from the 1986 design doc for the unpublished LucasArts game *Labyrinth*, which we accessed via web archive and extracted from the report images using OCR. We chose this last

document to provide a real game design doc that the LLM reasonably would not know about (compared to a published game), and for its longer document length (around 3 pages) that would stretch the limits of naive methods. We prototyped our initial design as a command-line interface (CLI) in Python, to reduce friction from UI development.

We started prototyping with the most naive approach possible: giving the LLM a list of items in the prompt as context, and asking it to revise the items with respect to new information. We chose gpt-4o after comparing to gpt-3.5-turbo and gpt-4-turbo, as the latter two had trouble sticking to formatting constraints. After some initial prompt engineering, the approach overall worked well: the LLM seemed to revise what we expected, although it was sometimes more conservative with its revisions.

However, this naive approach did not scale as the context window grew large. Three issues stood out: as the context size increases, the LLM produced more errors and unnecessary rewordings or reorderings of information, as the LLM has to reproduce the entire context verbatim; it is costly, as the LLM needs to reproduce the entire context even for a small edit; and the document context may exceed the input token length for the LLM, requiring sharding to handle. Note that this method of asking the LLM to reproduce the entire project with one edit is what was being used in systems like OpenAI's Canvas interface or tldraw MakeReal —doable for short documents, but fragile and not scale-able.

In our second iteration, we tried segmenting the context in a flatmap operation: map the changes over the context as separate LLM calls, then remerge them into one list. Though this approach sometimes led to reasonable edits, the LLM tended to mention the new information in too many places, producing duplicate information and unnecessary asides. This tendency seemed a limitation of the approach and not something that can be resolved with better prompt engineering: because each call presents a separate piece of context, the LLM has no awareness about how it's already adjusted the context, and this results in changes that we perceive as redundant information when viewing the changes in aggregate. The approach also offered greater chances for the LLM to deviate from the expected output format, as it required many more calls. (We ultimately learned from this experience that we needed to separate conflict *detection* from conflict *resolution*—

retrieval from generation—but did not know that here.)

We explored a compromise between these holistic and sharding solutions: to chunk the context into batches, using a metric such as embeddings or concept induction clustering, then feed the LLM these batches to revise (say, 5 short paragraphs) rather than single items. While this has the potential to suffer from some issues with our second method, it can hopefully reduce the amount of redundant information in the revised context and the potential for deviating from output format expectations (as less calls are made overall). As another alternative to random chunking, we also explored batching using LLooM [101], an open-source library that performs inductive coding on unstructured documents, clustering them by category.

However, inductive clustering of information was also limited in practice. The issue is that it assumes a tree-like structure (a hierarchy of concepts), but in practice, there are many mutual dependencies that cross branches of the tree—say, a character description and the graphics to represent them. **Intent specifications have a *heterarchical*, not hierarchical, structure.** A *heterarchy* is a relation between elements of information that "possess the potential for being ranked in a number of different ways" [41, p. 3]—here, information ranked and clustered just-in-time dependent on user query (the new information being integrated). Because this information could be separate in a hierarchical ranking, the LLM could not make consistent changes if one concept touches upon multiple clusters; and thus a similar problem of "duplicate information during resolution" again emerged.

From here, we went back to the drawing board. To ground our explorations in concrete data and identify what was needed, one author took the notes from the Labyrinth design document—a game none of us were familiar with—put them on sticky notes in a digital canvas environment, and mapped how they were inter-related (Figure 3.10). They drew arrows between the "dependencies" and highlighted key entities and concepts. They also began to categorize types of information (e.g., levels, characters, suggestions). We realized that this dependency graph resembled a knowledge graph. (Note that our explorations of knowledge graphs at this point were right around the time the first KG-based solutions to RAG architectures were released—it was not a popular idea

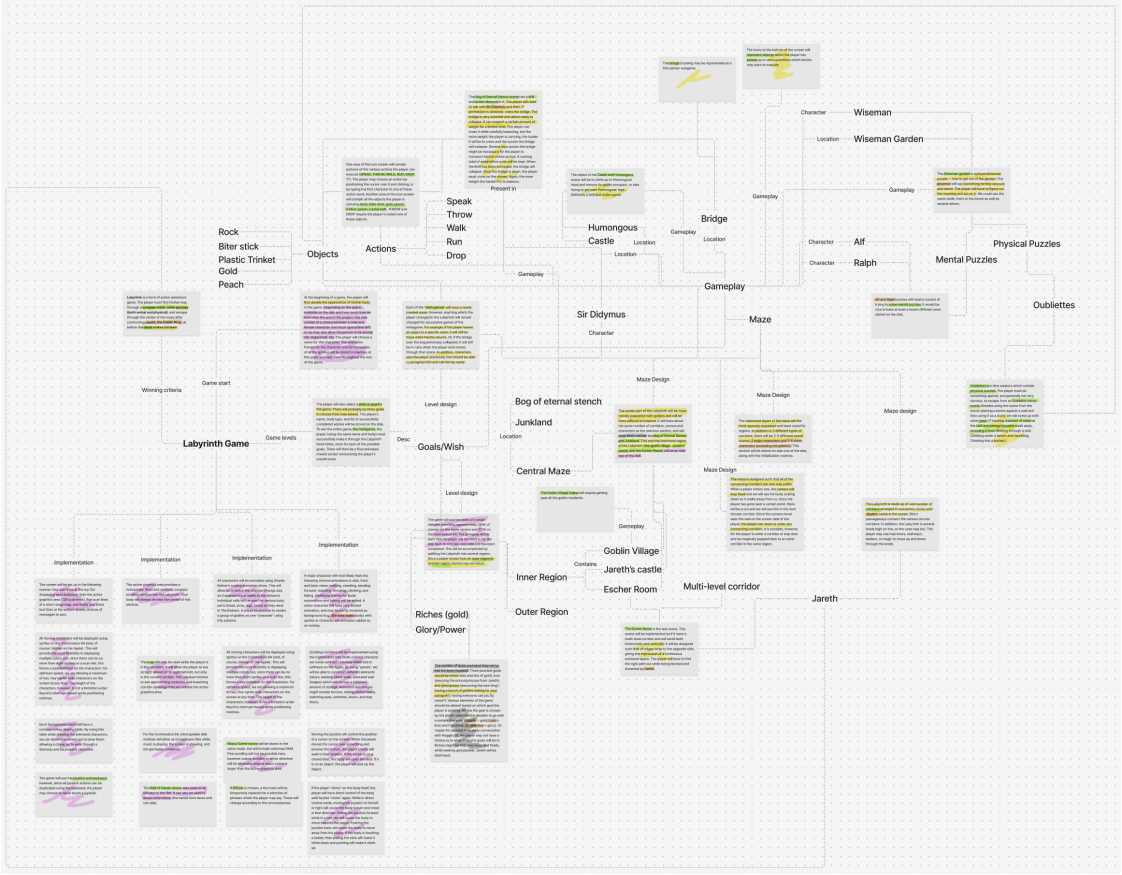


Figure 3.10: Creating a manual knowledge graph: Design explorations on how to represent the informational dependencies and entities of the Labyrinth game design document.

at the time.)

We learned a lot from this process. First, there are **"project-specific abstractions"—concepts and terms that have specific meaning to the project,** some of which *are not capitalized* (and hence may not stand out upon first glance). Project-specific abstractions accumulate over time across interactions and become part of the unique *common ground* shared between parties. An example of this in Labyrinth was the concept of a "wish game." The "wish game" concept may not be recognized by a classical NER system as coherent, important term, since NER systems are trained on well-established rather than ad-hoc terminology. Another example is a term like "labyrinth" that has a meaning in English but may be re-defined or differently interpreted for the project context: here, Labyrinth stands for the game holistically, rather than a specific maze. Second, there are relations between chunks of information, that can be described in a form akin to a knowledge graph, e.g. "is parent of," "appears in," and which connect the pieces of information. Third, not all information is segmented correctly (e.g., one card might introduce both a new character and a monster type), emphasizing the importance of chunking during pre-processing. Fourth, real design documents also contain "suggestions," such as remarks prefaced with a "maybe", that are not fully committed to the idea-space of the project, and which we may need to treat differently than fully committed information when resolving conflicts.[12]

We finally decided upon an LLM-induced knowledge graph architecture, implementing it ourselves independently before discovering HippoRAG [71] and adopting their Personalized PageRank technique. The induction step uses an LLM to generate entities and relations (nodes and edges), similar to the process from HippoRAG. The knowledge graph is updated upon confirmation of each resolved conflict (when user pressed the green checkmark). We found that this technique was effective at enhancing recall and extensible in that the retrieval step better scales when the information store grows large, shifting from (at best) a scan of all chunks of information $O(n)$, to a more limited filtering step after retrieval, $O(kg(n))$, which scales with the *scope of conflict*—higher-degree

[12] Reflecting, the author who had made the Mars game design doc also discovered that they had included suggestions in the doc, which hints that this may be behavior that occurs often in practice.

changes, such as changing the setting of a game from Mars to Venus, could still result in latency, but lower-degree changes, like changing a specific detail of a character, are quite fast. In practice and real settings, users are liable to introduce small details much more often than vast changes of their intent—hence, KG-retrieval is quite optimal, since it is relatively low-cost and low-latency.[13]

Our "global conflict resolver" prompt that is triggered when the user presses the global Make Changes button learned from the above experiences. For prototype purposes, we do in this step rely upon the LLM to perform a rewrite of the chunks. However, we only feed in the *conflicting* chunks—the chunks our end-to-end pipeline has flagged as potentially conflicting. We then ask the model to rewrite the text. This was a bit fragile, and can be improved in later iterations; however, the key thing is that the model needs more context to not introduce repeat information when integrating the new information. Here is our "Make Changes" prompt:

```
You are an information management system where the user has
stored unstructured documents for a project that they are
working on. The user has a new piece of information they wish {
action_instructions} the project: "{newInfo}"

You have detected that this information conflicts or contradicts
 multiple existing pieces of information. Here are a list of
existing texts. Edit the list as much as required to resolve the
 conflict. If you don't change some texts, that is OK. If you
want to delete a text entirely, mark that line with "DELETE". Be
 as conservative as possible in your changes. Do not repeat
information. Return the revised list of texts as a numbered list
 in Markdown format. Return only the numbered list.

TEXTS:
{all_docs}
```

[13]The caveat is that the KG needs to be initialized prior to use, which takes many LLM calls to accomplish, at least $O(n)$ for the initial number of chunks. However, this is a one-time, upfront cost.

where “action_type” can be either "to add to" or "to integrate into."

Note finally that semantic commits cannot be (fully) resolved using naive cosine similarity of vector embeddings. A naive approach may be to grab the most similar embeddings and only operate on those. Though this catches some semantic similarities, it only catches the most direct. For instance, consider the statement: “The enemy should sneak up behind the player when the enemy is not in the player’s field of view,” as part of the design document for a 3D game. If the user wants to change the game plan to target a 2D, top-down view, the semantic commit assistant should flag the enemy’s behavior as potentially in conflict with the new design, requiring (at the very least) clarification. Embeddings alone would not reveal this hidden dependency between information.

3.9.2 Conflict Classification: Criteria—Direct, Ambiguous, and Non-Conflict in the Context of Possible Worlds and Reference Ambiguity

This section describes our classification criteria for direct conflicts, ambiguous conflicts, and non-conflicts in further detail, incorporating essential background on issues such as possible worlds and reference ambiguity. Four coauthors developed this criteria over time in an iterative process while creating the evals and mutually decided upon these definitions. We provide this information for readers who are looking to make similar evaluation datasets.

3.9.2.1 Direct Conflict

A **direct conflict**, or *contradiction* in NLI [89], refers to a case where two statements, when evaluated within the same possible world [126]—that is, under the same common ground and a “normal interpretation”—cannot both be true simultaneously. In other words, if the statements share the same context, the truth of one necessarily entails the falsity of the other, meaning that there is no possible world, under a normal interpretation, in which both statements coexist as true. This concept reflects the notion of the Law of Non-Contradiction in formal logic [81], and it includes situations where, even in the absence of explicit negation, people intuitively judge the statements as “this

doesn't make sense." For example, "Alice was born in 1990" and "Alice is 20 years old in 2025" present a temporal contradiction—since they cannot both be true for the same Alice, they are classified as a direct conflict.

3.9.2.2 Ambiguous Conflict

An **ambiguous conflict** refers to a situation where two statements appear contradictory on the surface but can be reconciled when additional context or background knowledge is introduced within the framework of a "normal interpretation." This is similar to Chen et al.'s notion of **reference ambiguity** [30]. Under common assumptions or everyday interpretations, the two statements might seem to conflict; however, if there exists a reasonable possible world—within that normal interpretation—in which both statements can be simultaneously true due to specific circumstances or extra information, then they are considered to be in an ambiguous conflict. For example, the statements "Sally sold a boat to John" and "John sold a boat to Sally" generally appear mutually exclusive, but if each transaction pertains to a different boat, then there is a possible world in which both statements hold true. Similarly, consider "Alice's father is a scientist" versus "Alice's father is a police officer." While these intuitively seem to conflict, if, for instance, Alice's father is a forensic specialist within the police force or Alice has more than one parental figure that she calls father, then there exists a possible world in which both statements are valid. Consequently, such cases are interpreted as ambiguous conflicts.

3.9.2.3 Non-conflict

A **non-conflict**, or *neutral* classification in NLI [89], refers to situations where two statements can coexist without incident, under most normal interpretations of the world shared between evaluators. We define the term "**normal interpretation**" as an interpretation carried out within the common ground that communicating parties share by "common sense" and within the broad society that they share. If two statements are evaluated within the same context or possible world, and if information or background knowledge strongly suggests the possibility that they can coexist, then they can be con-

sidered as not being in conflict. If, however, an interpretation completely deviates from the scope expected in the common world we share—such as when discussing Alice while presupposing that every Alice is a different entity, or when making claims like "in this worldview 1+1=3" or "100 is less than 99" without any context—then such cases would be classified as in conflict.

It is obviously true that these cannot be completely formal definitions: the scope of "normal interpretation" varies from person to person or situation to situation in terms of subjectivity and diversity. Some individuals might judge the same statements as conflicting, while others might not see any conflict. This acknowledges that the *truthlikeness* (or *verisimilitude* [135]) of a statement is evaluated relatively according to individual interpretation, and under the premise that completely measurable contradictions or absolute truths do not exist objectively, we can categorize statements into direct conflict, ambiguous conflict, and non-conflict based only on these relative criteria.

CHAPTER 4

CONCLUSION

Through the two articles presented above, I have examined a range of issues surrounding AI memory, while organizing them into a coherent problem space and exploring both possible design directions and practical ways of responding to them.

The first article, *Metaphors for Memory: Charting a Design Space of AI Memory Tools and Interfaces*, sought to systematically articulate the still underexplored problem of AI agent memory system design by reviewing existing AI memory tools and interfaces and mapping them onto a design space. Building on this design space, it analyzed the structures, operations, interaction choices, dominant memory metaphors, and remaining gaps of current systems, while also proposing less-used metaphors that broaden the design directions available for future AI memory systems.

The second article, *Semantic Commit: Helping Users Update Intent Specifications for AI Memory at Scale*, addressed the problem of semantic conflict that arises in the practical management of such memory systems. More specifically, it formalized whether and how, in the context of long-term AI agent collaboration, users can specify their intent to agents in a clear and reviewable form, and whether semantic conflicts that emerge in the process of managing and updating such intent specifications can be detected and resolved collaboratively by users and agents. In this process, we introduced the concept of *intent specification*, developed the SemanticCommit interface and a method for conflict detection, and used a user study to examine what kinds of control and review mechanisms people need when carrying out such memory updates in practice.

Through this mémoire, readers may come to understand the architecture, operations, interfaces, and affordances of AI agent memory systems in a more integrated way, and diagnose how particular design choices reveal specific limitations and gaps. The dominant and less-used metaphors presented in this mémoire may also help designers reconsider the characteristics of such agent memory systems and find inspiration for new forms of interaction with agent memory. It also offers a direct example for thinking about

what kinds of resolution strategies and workflows users prefer when they encounter recurring issues such as conflict in the course of using memory systems in practice, and what kinds of interfaces designers might provide to support them.

At the same time, we raise several open questions concerning the design of interaction with, and management of, AI agent memory.

(1) First, as the affordances of agents expand, the behavioral rules, detailed instructions, exception conditions, and role-specific contexts associated with those tasks are also specified in increasingly diverse and elaborate ways. As a result, specification documents that encode user intent may gradually acquire a scale and complexity that users can no longer read at once or understand at a glance. How, then, should users be able to understand and manage such material? What kinds of interfaces should we provide to support this? And how should we present conflicts, outdated information, contextual inconsistencies, errors, duplication, or broader breakdowns of coherence in ways that enable meaningful human review and judgment? In this regard, the problem of assigning and managing information within AI memory appears to intersect with prior work on personal and organizational information management [2, 3, 171], which has also been discussed in HCI in its own right.

(2) Second, future interactive systems may be less likely to involve a single agent to which users specify all of their intent, and more likely to involve multiple agents across multiple providers, or multiple subagents within the same system whose roles are intentionally differentiated for reasons of security, role separation, access control, and distribution of responsibility. Users may therefore articulate different intents to different agents, while these agents in turn request, exchange, or provide memory to one another as needed, and sometimes update it by proxy. Much of this process may also evolve away from users manually approving every transfer and toward delegated permission structures in which agents act within authority that users have provisionally granted. The difficulty is that this makes it far more complex for users to control what information should or should not flow to whom, in what context, for how long, and to what extent. For example, a user may be willing to provide location data for shopping and delivery while refusing to provide the same information to other agent systems responsible for

advertising optimization or behavioral tracking. Moreover, in physically navigating to the user's home, a delivery robot may come to acquire not only an address but also information about the home's location, car ownership, daily routines, and movement patterns. A user may nevertheless want such information to remain local only within each delivery robot memory system, and never to be connected to other functions of the delivery system or to external agent systems. In this regard, concepts such as classified information [50, 51] and compartmentalization [5], which have been developed in the context information security and cybersecurity, may offer useful guidance for thinking through this problem.

(3) Third, it is equally important to consider how human users can remain in the loop in a meaningful way throughout these memory-management processes. Delegating the management of intent specifications to agents may create over-reliance and loss of control. This concern becomes even more important as agents are entrusted with increasingly sensitive information and broader forms of authority, since the consequences of losing control may then be severe. Current forms of AI consent-seeking may still be too difficult for ordinary users, such that people may either avoid using these systems altogether or simply press yes without adequately understanding what they are agreeing to. At the same time, current permission requests are often not sufficiently granular; put differently, context building between agent and user is still too weak, which means that systems may ask for permission too often, including in situations users perceive as unnecessary, thereby encouraging blanket approval. In this respect, problems of context building and common grounding [37, 38] may be addressed by improving interaction design in HCI, while more technical NLP approaches may also help by developing techniques for autonomously accumulating and refining user intent over time.

(4) Finally, there remains the broader question of how to design and evaluate interactive systems capable of handling all of these conditions at once. Even in settings involving large-scale memory, complex permission structures, multi-agent environments, and long-term use, our systems should support users in reading, revising, comparing, reviewing, and tracing changes to memory. At the same time, such systems should support users in having a genuine sense of ownership over the memory system, and in being

able to exercise real control over it in practice. Meanwhile, such interfaces should not be assumed to belong only to desktop or laptop environments, but should also be reconsidered across mobile devices, wearables, voice interfaces, and other modes of interaction. Moreover, from the standpoint of evaluation, the accumulation and management of memory in such systems unfolds at minimum over weeks, months, or even years, which means that short lab studies alone are unlikely to be sufficient. Longitudinal study [118] may therefore be a more appropriate way to evaluate these systems in practice.

I believe these works open room for exploring a wider range of possibilities for memory systems. While they help us better understand the current design space of agent memory and the metaphors drawn from machine learning, computer science, and anthropomorphized views that have shaped it, they also leave room for imagining and designing how AI agent memory might be structured and how it should interact with users from a wider range of perspectives. The questions raised above may in turn serve as concrete examples of the kinds of issues that future work will need to address, and thus as useful starting points for further design exploration. With this in mind, this mémoire can serve as a useful starting point for exploring future design directions for the interaction and management of AI agent memory.

REFERENCES

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[2] M. S. Ackerman and T. W. Malone. Answer garden: a tool for growing organizational memory. In *Proceedings of the ACM SIGOIS and IEEE CS TC-OA Conference on Office Information Systems*, COCS '90, page 31–39, New York, NY, USA, 1990. Association for Computing Machinery. ISBN 0897913582. doi: 10.1145/91474.91485. URL https://doi.org/10.1145/91474.91485.

[3] Mark S. Ackerman, Juri Dachtera, Volkmar Pipek, and Volker Wulf. Sharing knowledge and expertise: The cscw view of knowledge management. *Comput. Supported Coop. Work*, 22(4–6):531–573, August 2013. ISSN 0925-9724. doi: 10.1007/s10606-013-9192-8. URL https://doi.org/10.1007/s10606-013-9192-8.

[4] LangChain AI. Langmem, 2025. URL https://langchain-ai.github.io/langmem/. Accessed: 2025-04-07.

[5] Ross Anderson. *Security Engineering: A Guide to Building Dependable Distributed Systems*, chapter 8.2. John Wiley & Sons, 2010. Section title: Compartmentation, the Chinese Wall and the BMA model.

[6] Petr Anokhin, Nikita Semenov, Artyom Sorokin, Dmitry Evseev, Andrey Kravchenko, Mikhail Burtsev, and Evgeny Burnaev. Arigraph: Learning knowledge graph world models with episodic memory for llm agents. *arXiv preprint arXiv:2407.04363*, 2024.

[7] Anthropic. Claude code. https://claude.com/product/claude-code, 2025. Accessed: 2025-12-12.

[8] Anthropic. Agent skills. Claude Code Docs, 2025. URL https://code.claude.com/docs/en/skills. Accessed: 2026-01-13.

[9] Anthropic. Create custom subagents (claude code docs). https://code.claude.com/docs/en/sub-agents, 2025. Accessed: 2026-01-16.

[10] Anthropic. Claude opus 4.5 system card. Technical report, Anthropic, November 2025. URL https://assets.anthropic.com/m/64823ba7485345a7/Claude-Opus-4-5-System-Card.pdf. System card. Dated November 2025.

[11] Anthropic. Claude sonnet 4.5 system card. Technical report, Anthropic, September 2025. URL https://assets.anthropic.com/m/12f214efcc2f457a/original/Claude-Sonnet-4-5-System-Card.pdf. System card. Dated September 2025.

[12] Anthropic. Equipping agents for the real world with agent skills. Anthropic Engineering blog post, October 2025. URL https://www.anthropic.com/engineering/equipping-agents-for-the-real-world-with-agent-skills. Published: 2025-10-16; Accessed: 2026-01-15.

[13] Anthropic. Claude 3.7 sonnet and claude code. https://www.anthropic.com/news/claude-3-7-sonnet, 02 2025. Accessed: 2025-04-07.

[14] Grigoris Antoniou and Frank van Harmelen. *Web Ontology Language: OWL*, pages 67–92. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004. ISBN 978-3-540-24750-0. doi: 10.1007/978-3-540-24750-0_4. URL https://doi.org/10.1007/978-3-540-24750-0_4.

[15] Ian Arawjo. To write code: The cultural fabrication of programming notation and practice. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, CHI '20, page 1–15, New York, NY, USA, 2020. Association for

Computing Machinery. ISBN 9781450367080. doi: 10.1145/3313831.3376731. URL https://doi.org/10.1145/3313831.3376731.

[16] Ian Arawjo, Chelse Swoopes, Priyan Vaithilingam, Martin Wattenberg, and Elena L Glassman. Chainforge: A visual toolkit for prompt engineering and llm hypothesis testing. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*, pages 1–18, 2024.

[17] Robert S. Arnold. *Software Change Impact Analysis*. IEEE Computer Society Press, Washington, DC, USA, 1996. ISBN 0818673842.

[18] Chetan Arora, John Grundy, and Mohamed Abdelrazek. *Advancing Requirements Engineering Through Generative AI: Assessing the Role of LLMs*, pages 129–148. Springer Nature Switzerland, Cham, 2024. ISBN 978-3-031-55642-5. doi: 10.1007/978-3-031-55642-5_6. URL https://doi.org/10.1007/978-3-031-55642-5_6.

[19] Gagan Bansal, Jennifer Wortman Vaughan, Saleema Amershi, Eric Horvitz, Adam Fourney, Hussein Mozannar, Victor Dibia, and Daniel S Weld. Challenges in human-agent communication. *arXiv preprint arXiv:2412.10380*, 2024.

[20] Stafford Beer. What is cybernetics? *Kybernetes*, 31(2):209–219, 2002.

[21] Dave Bergmann. What is a context window? IBM Think, November 2024. URL https://www.ibm.com/think/topics/context-window. Accessed: 2025-12-12.

[22] Thomas Berlage. A selective undo mechanism for graphical user interfaces based on command objects. *ACM Trans. Comput.-Hum. Interact.*, 1(3):269–294, September 1994. ISSN 1073-0516. doi: 10.1145/196699.196721. URL https://doi.org/10.1145/196699.196721.

[23] Jeffrey R Binder and Rutvik H Desai. The neurobiology of semantic memory. *Trends in cognitive sciences*, 15(11):527–536, 2011.

[24] Michael Mose Biskjaer, Peter Dalsgaard, and Kim Halskov. A constraint-based understanding of design spaces. In *Proceedings of the 2014 conference on Designing interactive systems*, pages 453–462, 2014.

[25] Guy A. Boy. Active design documents. In *Proceedings of the 2nd Conference on Designing Interactive Systems: Processes, Practices, Methods, and Techniques*, DIS '97, page 31–36, New York, NY, USA, 1997. Association for Computing Machinery. ISBN 0897918630. doi: 10.1145/263552.263572. URL https://doi.org/10.1145/263552.263572.

[26] Jeffrey M. Bradshaw, Paul J. Feltovich, Hyuckchul Jung, Shriniwas Kulkarni, William Taysom, and Andrzej Uszok. Dimensions of adjustable autonomy and mixed-initiative interaction. In Matthias Nickles, Michael Rovatsos, and Gerhard Weiss, editors, *Agents and Computational Autonomy*, pages 17–39, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg. ISBN 978-3-540-25928-2.

[27] Jean-Michel Bruel, Sophie Ebersold, Florian Galinier, Manuel Mazzara, Alexandr Naumchev, and Bertrand Meyer. The role of formalism in system requirements. *ACM Comput. Surv.*, 54(5), May 2021. ISSN 0360-0300. doi: 10.1145/3448975. URL https://doi.org/10.1145/3448975.

[28] Zana Buçinca, Maja Barbara Malaya, and Krzysztof Z Gajos. 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): 1–21, 2021.

[29] Greg Ceccarelli. SpecStory Launch. SpecStory Blog, December 2024. URL https://specstory.com/blog/specstory-launch. Published: 2024-12-16. Accessed: 2026-01-15.

[30] Sihao Chen, Chaitanya Malaviya, Alex Fabrikant, Hagai Taitelbaum, Tal Schuster, Senaka Buthpitiya, and Dan Roth. On reference (in-)determinacy in natural language inference. In Luis Chiruzzo, Alan Ritter, and Lu Wang, editors, *Findings*

of the Association for Computational Linguistics: NAACL 2025, pages 8066–8078, Albuquerque, New Mexico, April 2025. Association for Computational Linguistics. ISBN 979-8-89176-195-7. doi: 10.18653/v1/2025.findings-naacl.450. URL https://aclanthology.org/2025.findings-naacl.450/.

[31] Valerie Chen, Alan Zhu, Sebastian Zhao, Hussein Mozannar, David Sontag, and Ameet Talwalkar. Need help? designing proactive ai assistants for programming. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*, CHI '25, New York, NY, USA, 2025. Association for Computing Machinery. ISBN 9798400713941. doi: 10.1145/3706598.3714002. URL https://doi.org/10.1145/3706598.3714002.

[32] Furui Cheng, Vilém Zouhar, Simran Arora, Mrinmaya Sachan, Hendrik Strobelt, and Mennatallah El-Assady. Relic: Investigating large language model responses using self-consistency. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*, CHI '24, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400703300. doi: 10.1145/3613904.3641904. URL https://doi.org/10.1145/3613904.3641904.

[33] Boris Cherny. Claude code: Best practices for agentic coding. https://www.anthropic.com/engineering/claude-code-best-practices, April 2025. Published: 2025-04-18. Accessed: 2025-12-12.

[34] Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. Mem0: Building production-ready ai agents with scalable long-term memory. *arXiv preprint arXiv:2504.19413*, 2025.

[35] John Joon Young Chung, Wooseok Kim, Kang Min Yoo, Hwaran Lee, Eytan Adar, and Minsuk Chang. Talebrush: Sketching stories with generative pretrained language models. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*, CHI '22, New York, NY, USA, 2022. Association for

Computing Machinery. ISBN 9781450391573. doi: 10.1145/3491102.3501819. URL https://doi.org/10.1145/3491102.3501819.

[36] Elizabeth Clark, Anne Spencer Ross, Chenhao Tan, Yangfeng Ji, and Noah A. Smith. 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*, IUI '18, page 329–340, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450349451. doi: 10.1145/3172944.3172983. URL https://doi.org/10.1145/3172944.3172983.

[37] Herbert H Clark. *Using language*. Cambridge University Press, 1996.

[38] Herbert H Clark and Susan E Brennan. Grounding in communication., 1991.

[39] Neal J Cohen and Larry R Squire. Preserved learning and retention of pattern-analyzing skill in amnesia: Dissociation of knowing how and knowing that. *Science*, 210(4466):207–210, 1980.

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

[41] Carole L Crumley. Heterarchy and the analysis of complex societies. *Archeological papers of the American anthropological association*, 6(1):1–5, 1995.

[42] D. Cubranic, G.C. Murphy, J. Singer, and K.S. Booth. Hipikat: a project memory for software development. *IEEE Transactions on Software Engineering*, 31(6): 446–465, 2005. doi: 10.1109/TSE.2005.71.

[43] Cursor. Cursor. https://cursor.com/, 2023. Accessed: 2025-12-12.

[44] Cursor. Git. https://cursor.com/help/integrations/git, 2026. Cursor Docs. Accessed: 2026-03-30.

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

[46] Sander de Jong, Ville Paananen, Benjamin Tag, and Niels van Berkel. Cognitive forcing for better decision-making: Reducing overreliance on ai systems through partial explanations. *Proc. ACM Hum.-Comput. Interact.*, 9(2), May 2025. doi: 10.1145/3710946. URL https://doi.org/10.1145/3710946.

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

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

[49] Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, Dasha Metropolitansky, Robert Osazuwa Ness, and Jonathan Larson. From local to global: A graph rag approach to query-focused summarization, 2024.

[50] Executive Office of the President. Executive order 12958 of april 17, 1995: Classified national security information. *Federal Register*, 60:19825–19843, 1995. URL https://www.federalregister.gov/documents/1995/04/20/95-9941/classified-national-security-information. Presidential document.

[51] Executive Office of the President. Executive order 13526 of december 29, 2009: Classified national security information. *Federal Register*, 75:707–731, 2010. URL https://www.federalregister.gov/documents/2010/01/05/E9-31418/classified-national-security-information. Presidential document.

[52] Hongfei Fan and Chengzheng Sun. Supporting semantic conflict prevention in

real-time collaborative programming environments. *ACM SIGAPP Applied Computing Review*, 12(2):39–52, 2012.

[53] Alessandro Fantechi, Stefania Gnesi, Lucia Passaro, and Laura Semini. Inconsistency detection in natural language requirements using chatgpt: a preliminary evaluation. In *2023 IEEE 31st International Requirements Engineering Conference (RE)*, pages 335–340, Germany, 2023. IEEE. doi: 10.1109/RE57278.2023.00045.

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

[55] K. J. Kevin Feng, Kevin Pu, Matt Latzke, Tal August, Pao Siangliulue, Jonathan Bragg, Daniel S. Weld, Amy X. Zhang, and Joseph Chee Chang. Cocoa: Co-planning and co-execution with ai agents, 2024. URL https://arxiv.org/abs/2412.10999.

[56] Mayara C. Figueiredo and Cleidson R. B. de Souza. 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, page 40–46, Zurich, Switzerland, 2012. IEEE Press. ISBN 9781467318242.

[57] Andrew Flinn. Community histories, community archives: Some opportunities and challenges. *Journal of the Society of Archivists*, 28(2):151–176, 2007.

[58] Andrew Flinn. Community archives. In Luciana Duranti and Patricia C. Franks, editors, *Encyclopedia of Archival Science*, pages 145–149. Rowman & Littlefield, Lanham, MD, 2015. ISBN 9780810888104.

[59] David Fox. Labyrinth. Internal game design document, Lucasfilm Ltd. Games Division, 1 1986. 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/.

[60] Bill Gaver, Tony Dunne, and Elena Pacenti. Design: cultural probes. *interactions*, 6(1):21–29, 1999.

[61] Gemini Team. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. Technical report, Google, 2025. URL https://storage.googleapis.com/deepmind-media/gemini/gemini_v2_5_report.pdf. Technical report.

[62] Marjan Ghazvininejad, Xing Shi, Jay Priyadarshi, and Kevin Knight. Hafez: an interactive poetry generation system. In Mohit Bansal and Heng Ji, editors, *Proceedings of ACL 2017, System Demonstrations*, pages 43–48, Vancouver, Canada, July 2017. Association for Computational Linguistics. URL https://aclanthology.org/P17-4008/.

[63] Git Project. Git. https://git-scm.com/, 2026. Official website. Accessed: 2026-03-30.

[64] GitHub. Github copilot. https://github.com/features/copilot, 2022. Accessed: 2025-12-12.

[65] Elena L. Glassman, Ziwei Gu, and Jonathan K. Kummerfeld. Ai-resilient interfaces, 2024.

[66] Google Cloud. Gemini cli. https://docs.cloud.google.com/gemini/docs/codeassist/gemini-cli, 2025. Last updated: 2025-12-11 (UTC). Accessed: 2025-12-12.

[67] Google for Developers. Prompt engineering for generative ai. https://developers.google.com/machine-learning/resources/prompt-eng, August 2025. Last updated: 2025-08-25 (UTC). Accessed: 2025-12-12.

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

[69] Ziwei Gu, Ian Arawjo, Kenneth Li, Jonathan K. Kummerfeld, and Elena L. Glassman. An ai-resilient text rendering technique for reading and skimming documents. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*, CHI '24, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400703300. doi: 10.1145/3613904.3642699. URL https://doi.org/10.1145/3613904.3642699.

[70] Prahlad Gupta and Neal J Cohen. Theoretical and computational analysis of skill learning, repetition priming, and procedural memory. *Psychological review*, 109 (2):401, 2002.

[71] Bernal Jiménez Gutiérrez, Yiheng Shu, Yu Gu, Michihiro Yasunaga, and Yu Su. Hipporag: neurobiologically inspired long-term memory for large language models. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, NIPS '24, Red Hook, NY, USA, 2025. Curran Associates Inc. ISBN 9798331314385.

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

[73] J. Han. 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*, pages 172–182, Australia, 1997. IEEE. doi: 10.1109/STEP.1997.615479.

[74] Rex Hartson and Pardha Pyla. *The UX Book: Process and Guidelines for Ensuring a Quality User Experience*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1st edition, 2012. ISBN 0123852412.

[75] Ahmed E. Hassan. The road ahead for mining software repositories. In *2008 Frontiers of Software Maintenance*, pages 48–57, 2008. doi: 10.1109/FOSM.2008.4659248.

[76] Taher H Haveliwala. Topic-sensitive pagerank. In *Proceedings of the 11th international conference on World Wide Web*, pages 517–526, 2002.

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

[78] Jeffrey Heer, Joseph M Hellerstein, and Sean Kandel. Predictive interaction for data transformation. In *CIDR*, pages 1–7, Asilomar, California, 2015. Citeseer, CIDR.

[79] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[80] Md Naimul Hoque, Bhavya Ghai, Kari Kraus, and Niklas Elmqvist. Portrayal: Leveraging nlp and visualization for analyzing fictional characters. In *Proceedings of the 2023 ACM Designing Interactive Systems Conference*, DIS '23, page 74–94, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9781450398930. doi: 10.1145/3563657.3596000. URL `https://doi.org/10.1145/3563657.3596000`.

[81] Laurence R. Horn. Contradiction. In Edward N. Zalta and Uri Nodelman, editors, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, Spring 2025 edition, 2025.

[82] Eric Horvitz. Principles of mixed-initiative user interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '99, page

159–166, New York, NY, USA, 1999. Association for Computing Machinery. ISBN 0201485591. doi: 10.1145/302979.303030. URL https://doi.org/10.1145/302979.303030.

[83] Yuki Hou, Haruki Tamoto, and Homei Miyashita. "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*, CHI EA '24, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400703317. doi: 10.1145/3613905.3650839. URL https://doi.org/10.1145/3613905.3650839.

[84] Shang-Hsien Hsieh, Hsien-Tang Lin, Nai-Wen Chi, Kuang-Wu Chou, and Ken-Yu Lin. Enabling the development of base domain ontology through extraction of knowledge from engineering domain handbooks. *Advanced Engineering Informatics*, 25(2):288–296, 2011. ISSN 1474-0346. doi: https://doi.org/10.1016/j.aei.2010.08.004. URL https://www.sciencedirect.com/science/article/pii/S1474034610000807. Information mining and retrieval in design.

[85] Ziheng Huang, Sebastian Gutierrez, Hemanth Kamana, and Stephen MacNeil. Memory sandbox: Transparent and interactive memory management for conversational agents. In *Adjunct Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, pages 1–3, 2023.

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

[87] Olga Iarygina, Kasper Hornbæk, and Aske Mottelson. Demand characteristics in human–computer experiments. *International Journal of Human-Computer Studies*, 193:103379, 2025. ISSN 1071-5819. doi: https://doi.org/10.1016/j.ijhcs.2024.103379. URL https://www.sciencedirect.com/science/article/pii/S1071581924001629.

[88] Instructor Python API. Instructor repository: .cursor/rules directory, 2025. URL https://github.com/instructor-ai/instructor/tree/main/.cursor/rules. Accessed: 2025-03-29.

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

[90] Peiling Jiang, Jude Rayan, Steven P Dow, and Haijun Xia. Graphologue: Exploring large language model responses with interactive diagrams. In *Proceedings of the 36th annual ACM symposium on user interface software and technology*, pages 1–20, 2023.

[91] Bernal Jimenez Gutierrez, Yiheng Shu, Yu Gu, Michihiro Yasunaga, and Yu Su. Hipporag: Neurobiologically inspired long-term memory for large language models. *Advances in Neural Information Processing Systems*, 37:59532–59569, 2024.

[92] Per Jönsson and Mikael Lindvall. *Impact Analysis*, pages 117–142. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005. ISBN 978-3-540-28244-0. doi: 10.1007/3-540-28244-0_6. URL https://doi.org/10.1007/3-540-28244-0_6.

[93] Haruhiko Kaiya and Motoshi Saeki. Using domain ontology as domain knowledge for requirements elicitation. In *14th IEEE International Requirements Engineering Conference (RE'06)*, pages 189–198, St. Paul, MN, USA, 2006. IEEE. doi: 10.1109/RE.2006.72.

[94] Majeed Kazemitabaar, Jack Williams, Ian Drosos, Tovi Grossman, Austin Zachary Henley, Carina Negreanu, and Advait Sarkar. 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*, UIST '24, New York, NY, USA, 2024. Association for

Computing Machinery. ISBN 9798400706288. doi: 10.1145/3654777.3676345. URL https://doi.org/10.1145/3654777.3676345.

[95] Majeed Kazemitabaar, Oliver Huang, Sangho Suh, Austin Z Henley, and Tovi Grossman. 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, page 695–714, New York, NY, USA, 2025. Association for Computing Machinery. ISBN 9798400713064. doi: 10.1145/3708359.3712104. URL https://doi.org/10.1145/3708359.3712104.

[96] Sunghun Kim, Kai Pan, and E. E. James Whitehead. Memories of bug fixes. In *Proceedings of the 14th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, SIGSOFT '06/FSE-14, page 35–45, New York, NY, USA, 2006. Association for Computing Machinery. ISBN 1595934685. doi: 10.1145/1181775.1181781. URL https://doi.org/10.1145/1181775.1181781.

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

[98] Clemens Nylandsted Klokmose, James R. Eagan, and Peter van Hardenberg. Mywebstrates: Webstrates as local-first software. In *Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology*, UIST '24, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400706288. doi: 10.1145/3654777.3676445. URL https://doi.org/10.1145/3654777.3676445.

[99] Philippe Laban, Jesse Vig, Marti Hearst, Caiming Xiong, and Chien-Sheng Wu. Beyond the chat: Executable and verifiable text-editing with llms. In *Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology*,

UIST '24, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400706288. doi: 10.1145/3654777.3676419. URL https://doi.org/10.1145/3654777.3676419.

[100] George Lakoff and Mark Johnson. *Metaphors We Live By*. University of Chicago Press, Chicago, 2003. ISBN 9780226468013. Originally published 1980; updated edition with afterword.

[101] Michelle S Lam, Janice Teoh, James A Landay, Jeffrey Heer, and Michael S Bernstein. Concept induction: Analyzing unstructured text with high-level concepts using lloom. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*, pages 1–28, 2024.

[102] Michelle S. Lam, Fred Hohman, Dominik Moritz, Jeffrey P Bigham, Kenneth Holstein, and Mary Beth Kery. Policy maps: Tools for guiding the unbounded space of llm behaviors. In *Proceedings of the 38th Annual ACM Symposium on User Interface Software and Technology*, UIST '25, New York, NY, USA, 2025. Association for Computing Machinery. ISBN 9798400720376. doi: 10.1145/3746059.3747680. URL https://doi.org/10.1145/3746059.3747680.

[103] LangChain. Langgraph overview. https://docs.langchain.com/oss/python/langgraph/overview, 2024. Accessed: 2025-12-12.

[104] LangChain-AI. Langmem. https://langchain-ai.github.io/langmem/, 2025. Accessed: 2025-12-12.

[105] Sangwook Lee, Adnan Abbas, Yan Chen, Young-Ho Kim, and Sang Won Lee. CHOIR: A chatbot-mediated organizational memory leveraging communication in university research labs. In *Proceedings of the 2026 CHI Conference on Human Factors in Computing Systems*. Association for Computing Machinery, 2026. doi: 10.1145/3772318.3791314. URL https://doi.org/10.1145/3772318.3791314. To appear.

[106] Yi-Chieh Lee, Naomi Yamashita, Yun Huang, and Wai Fu. "i hear you, i feel you": Encouraging deep self-disclosure through a chatbot. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, CHI '20, page 1–12, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450367080. doi: 10.1145/3313831.3376175. URL https://doi.org/10.1145/3313831.3376175.

[107] Nancy G. Leveson. System safety in computer-controlled automotive systems. *SAE Transactions*, 109:287–294, 2000. ISSN 0096736X, 25771531. URL http://www.jstor.org/stable/44699139.

[108] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474, 2020.

[109] Jingyi Li, Eric Rawn, Jacob Ritchie, Jasper Tran O'Leary, and Sean Follmer. 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*, UIST '23, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9798400701320. doi: 10.1145/3586183.3606831. URL https://doi.org/10.1145/3586183.3606831.

[110] Brian Y. Lim, Anind K. Dey, and Daniel Avrahami. 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*, CHI '09, page 2119–2128, New York, NY, USA, 2009. Association for Computing Machinery. ISBN 9781605582467. doi: 10.1145/1518701.1519023. URL https://doi.org/10.1145/1518701.1519023.

[111] 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. Rambler: Supporting writing with speech

via llm-assisted gist manipulation. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*, CHI '24, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400703300. doi: 10.1145/3613904.3642217. URL https://doi.org/10.1145/3613904.3642217.

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

[113] Bang Liu, Xinfeng Li, Jiayi Zhang, Jinlin Wang, Tanjin He, Sirui Hong, Hongzhang Liu, Shaokun Zhang, Kaitao Song, Kunlun Zhu, et al. Advances and challenges in foundation agents: From brain-inspired intelligence to evolutionary, collaborative, and safe systems. *arXiv preprint arXiv:2504.01990*, page 20, 2025.

[114] Michael Xieyang Liu, Tongshuang Wu, Tianying Chen, Franklin Mingzhe Li, Aniket Kittur, and Brad A Myers. 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*, CHI '24, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400703300. doi: 10.1145/3613904.3642149. URL https://doi.org/10.1145/3613904.3642149.

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

[116] Robert K Logan. Feedforward, ia richards, cybernetics and marshall mcluhan. *Systema: Connecting Catter, Life, Culture and Technology*, 3(1):177–185, 2015.

[117] Nick Logler, Daisy Yoo, and Batya Friedman. Metaphor cards: A how-to-

guide for making and using a generative metaphorical design toolkit. In *Proceedings of the 2018 Designing Interactive Systems Conference*, DIS '18, page 1373–1386, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450351980. doi: 10.1145/3196709.3196811. URL https://doi.org/10.1145/3196709.3196811.

[118] Tao Long, Sitong Wang, Émilie Fabre, Tony Wang, Anup Sathya, Jason Wu, Savvas Dimitrios Petridis, Ding Li, Tuhin Chakrabarty, Yue Jiang, et al. Facilitating longitudinal interaction studies of ai systems. In *Adjunct Proceedings of the 38th Annual ACM Symposium on User Interface Software and Technology*, pages 1–5, 2025.

[119] Sebastian Lubos, Alexander Felfernig, {Thi Ngoc Trang} Tran, Damian Garber, Merfat {El Mansi}, {Seda Polat} Erdeniz, and {Viet Man} Le. Leveraging llms for the quality assurance of software requirements. In Grischa Liebel, Irit Hadar, and Paola Spoletini, editors, *Proceedings - 32nd IEEE International Requirements Engineering Conference, RE 2024*, pages 389–397, United States, Aug 2024. IEEE Computer Society. 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.

[120] Niklas Luhmann. Kommunikation mit zettelkästen: Ein erfahrungsbericht. In *Öffentliche Meinung und sozialer Wandel/Public Opinion and Social Change*, pages 222–228. Springer, 1981.

[121] Niklas Luhmann. Communicating with slip boxes: An empirical account. https://zettelkasten.de/communications-with-zettelkastens/, 2023. English translation of Luhmann (1981), available online.

[122] Qianou Ma, Weirui Peng, Chenyang Yang, Hua Shen, Kenneth Koedinger, and Tongshuang Wu. What should we engineer in prompts? training humans in

requirement-driven llm use, 2024. URL https://arxiv.org/abs/2409.08775.

[123] Shuai Ma, Qiaoyi Chen, Xinru Wang, Chengbo Zheng, Zhenhui Peng, Ming Yin, and Xiaojuan Ma. Towards human-ai deliberation: Design and evaluation of llm-empowered deliberative ai for ai-assisted decision-making. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*, pages 1–23, 2025.

[124] Allan MacLean, Richard M. Young, Victoria M. E. Bellotti, and Thomas P. Moran. Questions, options, and criteria: Elements of design space analysis. *Human–Computer Interaction*, 6(3-4):201–250, 1991. doi: https://doi.org/10.1080/07370024.1991.9667168.

[125] Sarnoff Mednick. The associative basis of the creative process. *Psychological review*, 69(3):220, 1962.

[126] Christopher Menzel. Possible Worlds. In Edward N. Zalta and Uri Nodelman, editors, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, Summer 2024 edition, 2024.

[127] Bryan Min and Haijun Xia. Feedforward in generative ai: Opportunities for a design space, 2025.

[128] Piotr Mirowski, Kory W. Mathewson, Jaylen Pittman, and Richard Evans. 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*, CHI '23, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9781450394215. doi: 10.1145/3544548.3581225. URL https://doi.org/10.1145/3544548.3581225.

[129] Dave Murray-Rust, Iohanna Nicenboim, and Dan Lockton. Metaphors for designers working with AI. *DRS2022: Bilbao*, 2022.

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

[131] Clifford Nass, Jonathan Steuer, and Ellen R Tauber. Computers are social actors. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 72–78, 1994.

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

[133] Jakob Nielsen. Enhancing the explanatory power of usability heuristics. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '94, page 152–158, New York, NY, USA, 1994. Association for Computing Machinery. ISBN 0897916506. doi: 10.1145/191666.191729. URL https://doi.org/10.1145/191666.191729.

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

[135] Graham Oddie and Gustavo Cevolani. Truthlikeness. In Edward N. Zalta and Uri Nodelman, editors, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, Winter 2022 edition, 2022.

[136] OpenAI. Chatgpt | ai chatbot to discover, learn & create. Web page, 2022. URL https://openai.com/chatgpt/overview/. Accessed: 2025-12-12.

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

[138] OpenAI. Codex. https://openai.com/codex/, 2025. Accessed: 2025-12-12.

[139] OpenAI. Best practices for prompt engineering with the openai api. https://help.openai.com/en/articles/6654000-best-practices-for-prompt-engineering-with-the-openai-ap 2025. Accessed: 2025-12-12. (Page shows: "Updated: 3 months ago" at time of access.).

[140] OpenAI. Gpt-5 system card. Technical report, OpenAI, August 2025. URL https://cdn.openai.com/gpt-5-system-card.pdf. System card. Published August 13, 2025.

[141] OpenAI. Use codex in github. https://developers.openai.com/codex/integrations/github, 2026. OpenAI Developers. Accessed: 2026-03-30.

[142] Charles Packer, Zeyuan Sun, Max Goldstein, Avishkar Basu, Soham Choudhury, Amog Kamsetty Pasarkar, Emma Strubell, and Andrew McCallum. Memgpt: Towards llms as operating systems. *arXiv preprint arXiv:2310.08560*, 2023. URL https://arxiv.org/abs/2310.08560.

[143] Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th annual acm symposium on user interface software and technology*, pages 1–22, 2023.

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

[145] Git Project. Git: Fast version control system, 01 2025. URL https://git-scm.com/. Accessed: 24 Jan. 2025.

[146] Kevin Pu, Daniel Lazaro, Ian Arawjo, Haijun Xia, Ziang Xiao, Tovi Grossman, and Yan Chen. 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, New York, NY, USA, 2025. Association for Computing Machinery. ISBN 9798400713941. doi: 10.1145/3706598.3713357. URL https://doi.org/10.1145/3706598.3713357.

[147] Preston Rasmussen, Pavlo Paliychuk, Travis Beauvais, Jack Ryan, and Daniel Chalef. Zep: a temporal knowledge graph architecture for agent memory. *arXiv preprint arXiv:2501.13956*, 2025.

[148] Partha Pratim Ray. A review on vibe coding: Fundamentals, state-of-the-art, challenges and future directions. *Authorea Preprints*, 2025.

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

[150] Ivor A Richards. The secret of feedforward. *Saturday Review*, 3:14–17, 1968.

[151] Stephen Robertson and Hugo Zaragoza. *The probabilistic relevance framework: BM25 and beyond*, volume 4. Now Publishers Inc, 2009.

[152] Diana Robinson, Christian Cabrera, Andrew D. Gordon, Neil D. Lawrence, and Lars Mennen. Requirements are all you need: The final frontier for end-user software engineering, 2024.

[153] Edmund T Rolls. The memory systems of the human brain and generative artificial intelligence. *Heliyon*, 10(11), 2024.

[154] Arleen Salles, Kathinka Evers, and Michele Farisco. Anthropomorphism in ai. *AJOB neuroscience*, 11(2):88–95, 2020.

[155] Advait Sarkar. Intention is all you need, 2024.

[156] Arvind Satyanarayan and Graham M. Jones. Intelligence as Agency: Evaluating the Capacity of Generative AI to Empower or Constrain Human Action, mar 27 2024. https://mit-genai.pubpub.org/pub/94y6e0f8.

[157] Pete Sawyer, Nelly Bencomo, Jon Whittle, Emmanuel Letier, and Anthony Finkelstein. Requirements-aware systems: A research agenda for re for self-adaptive systems. In *2010 18th IEEE International Requirements Engineering Conference*, pages 95–103, 2010. doi: 10.1109/RE.2010.21.

[158] Daniel L. Schacter. Implicit memory: History and current status. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 13(3):501–518, 1987. doi: 10.1037/0278-7393.13.3.501.

[159] Johannes Schmidt. Niklas luhmann‘s card index: Thinking tool, communication partner, publication machine. In *Forgetting Machines. Knowledge Management Evolution in Early Modern Europe*, volume 53. Brill, 2016.

[160] Donald A Schön. Generative metaphor: A perspective on problem-setting in social policy. *Metaphor and thought*, 2:137–163, 1979.

[161] Omar Shaikh, Kristina Gligoric, Ashna Khetan, Matthias Gerstgrasser, Diyi Yang, and Dan Jurafsky. Grounding gaps in language model generations. In Kevin Duh, Helena Gomez, and Steven Bethard, editors, *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 6279–6296, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.naacl-long.348. URL `https://aclanthology.org/2024.naacl-long.348/`.

[162] Omar Shaikh, Michelle Lam, Joey Hejna, Yijia Shao, Michael Bernstein, and Diyi Yang. Show, don’t tell: Aligning language models with demonstrated feedback, 2024.

[163] Omar Shaikh, Hussein Mozannar, Gagan Bansal, Adam Fourney, and Eric Horvitz. Navigating rifts in human-llm grounding: Study and benchmark, 2025. URL https://arxiv.org/abs/2503.13975.

[164] Omar Shaikh, Shardul Sapkota, Shan Rizvi, Eric Horvitz, Joon Sung Park, Diyi Yang, and Michael S Bernstein. Creating general user models from computer use. In *Proceedings of the 38th Annual ACM Symposium on User Interface Software and Technology*, pages 1–23, 2025.

[165] Shreya Shankar, J.D. Zamfirescu-Pereira, Bjoern Hartmann, Aditya Parameswaran, and Ian Arawjo. 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*, UIST '24, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400706288. doi: 10.1145/3654777.3676450. URL https://doi.org/10.1145/3654777.3676450.

[166] Yijia Shao, Vinay Samuel, Yucheng Jiang, John Yang, and Diyi Yang. Collaborative gym: A framework for enabling and evaluating human-agent collaboration. *arXiv preprint arXiv:2412.15701*, 2024.

[167] D. Spinellis. Version control systems. *IEEE Software*, 22(5):108–109, 2005. doi: 10.1109/MS.2005.140.

[168] Sangho Suh, Bryan Min, Srishti Palani, and Haijun Xia. Sensecape: Enabling multilevel exploration and sensemaking with large language models. In *Proceedings of the 36th annual ACM symposium on user interface software and technology*, pages 1–18, 2023.

[169] Theodore Sumers, Shunyu Yao, Karthik R Narasimhan, and Thomas L Griffiths. Cognitive architectures for language agents. *Transactions on Machine Learning Research*, 2023.

[170] Lev Tankelevitch, Viktor Kewenig, Auste Simkute, Ava Elizabeth Scott, Advait Sarkar, Abigail Sellen, and Sean Rintel. The metacognitive demands and opportunities of generative ai. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*, CHI '24, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400703300. doi: 10.1145/3613904.3642902. URL https://doi.org/10.1145/3613904.3642902.

[171] Jaime Teevan, William Jones, and Robert Capra. Personal information management (pim) 2008. *SIGIR Forum*, 42(2):96–103, November 2008. ISSN 0163-5840. doi: 10.1145/1480506.1480524. URL https://doi.org/10.1145/1480506.1480524.

[172] Valerio Terragni, Annie Vella, Partha Roop, and Kelly Blincoe. The future of ai-driven software engineering. *ACM Trans. Softw. Eng. Methodol.*, 34(5), May 2025. ISSN 1049-331X. doi: 10.1145/3715003. URL https://doi.org/10.1145/3715003.

[173] Yu-Min Tseng, Yu-Chao Huang, Teng-Yun Hsiao, Wei-Lin Chen, Chao-Wei Huang, Yu Meng, and Yun-Nung Chen. Two tales of persona in LLMs: A survey of role-playing and personalization. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen, editors, *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 16612–16631, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-emnlp.969. URL https://aclanthology.org/2024.findings-emnlp.969/.

[174] Endel Tulving. Episodic memory: From mind to brain. *Annual review of psychology*, 53(1):1–25, 2002.

[175] A. M. Turing. Computing machinery and intelligence. *Mind*, 59(236):433–460, October 1950. doi: 10.1093/mind/LIX.236.433. URL https://doi.org/10.1093/mind/LIX.236.433.

[176] Priyan Vaithilingam, Ian Arawjo, and Elena L Glassman. Imagining a future of designing with ai: Dynamic grounding, constructive negotiation, and sustainable motivation. In *Proceedings of the 2024 ACM Designing Interactive Systems Conference*, pages 289–300, 2024.

[177] Priyan Vaithilingam, Munyeong Kim, Frida-Cecilia Acosta-Parenteau, Daniel Lee, Amine Mhedhbi, Elena L Glassman, and Ian Arawjo. Semantic commit: Helping users update intent specifications for ai memory at scale. In *Proceedings of the 38th Annual ACM Symposium on User Interface Software and Technology*, pages 1–18, 2025.

[178] Axel Van Lamsweerde. *Requirements engineering: From system goals to UML models to software*, volume 10. Chichester, UK: John Wiley & Sons, 2009.

[179] Jo Vermeulen, Kris Luyten, Elise van den Hoven, and Karin Coninx. 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*, CHI '13, page 1931–1940, New York, NY, USA, 2013. Association for Computing Machinery. ISBN 9781450318990. doi: 10.1145/2470654.2466255. URL https://doi.org/10.1145/2470654.2466255.

[180] Kendall L Walton. *Mimesis as make-believe: On the foundations of the representational arts*. Harvard University Press, 1993.

[181] Thiemo Wambsganss, Christina Niklaus, Matthias Cetto, Matthias Söllner, Siegfried Handschuh, and Jan Marco Leimeister. Al: An adaptive learning support system for argumentation skills. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, CHI '20, page 1–14, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450367080. doi: 10.1145/3313831.3376732. URL https://doi.org/10.1145/3313831.3376732.

[182] Ruotong Wang, Xinyi Zhou, Lin Qiu, Joseph Chee Chang, Jonathan Bragg, and

Amy X. Zhang. Social-rag: Retrieving from group interactions to socially ground AI generation. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*, CHI '25, New York, NY, USA, 2025. Association for Computing Machinery. ISBN 9798400713941. doi: 10.1145/3706598.3713749. URL https://doi.org/10.1145/3706598.3713749.

[183] Xinru Wang, Hannah Kim, Sajjadur Rahman, Kushan Mitra, and Zhengjie Miao. Human-llm collaborative annotation through effective verification of llm labels. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*, CHI '24, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400703300. doi: 10.1145/3613904.3641960. URL https://doi.org/10.1145/3613904.3641960.

[184] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35: 24824–24837, 2022.

[185] Cat Wu. Press # to instruct claude code to add a memory. then, type your memory and hit enter to add it to the claude.md file., April 2025. URL https://x.com/_catwu/status/1904941906867413054. Tweet by the Product Designer of Claude Code.

[186] Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-mem: Agentic memory for llm agents. *arXiv preprint arXiv:2502.12110*, 2025.

[187] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=WE_vluYUL-X.

[188] Catherine Yeh, Gonzalo Ramos, Rachel Ng, Andy Huntington, and Richard

Banks. Ghostwriter: Augmenting collaborative human-ai writing experiences through personalization and agency. *arXiv preprint arXiv:2402.08855*, 2024.

[189] Ryan Yen and Jian Zhao. Memolet: Reifying the reuse of user-ai conversational memories. In *Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology*, pages 1–22, 2024.

[190] Young Seok Yoon and Brad A. Myers. Supporting selective undo in a code editor. In *Proceedings of the 37th International Conference on Software Engineering - Volume 1*, ICSE '15, page 223–233, Florence, Italy, 2015. IEEE Press. ISBN 9781479919345.

[191] Ann Yuan, Andy Coenen, Emily Reif, and Daphne Ippolito. Wordcraft: Story writing with large language models. In *Proceedings of the 27th International Conference on Intelligent User Interfaces*, IUI '22, page 841–852, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450391443. doi: 10.1145/3490099.3511105. URL https://doi.org/10.1145/3490099.3511105.

[192] JD Zamfirescu-Pereira, Eunice Jun, Michael Terry, Qian Yang, and Björn Hartmann. Beyond code generation: Llm-supported exploration of the program design space. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*, pages 1–17, 2025.

[193] Zeyu Zhang, Quanyu Dai, Xiaohe Bo, Chen Ma, Rui Li, Xu Chen, Jieming Zhu, Zhenhua Dong, and Ji-Rong Wen. A survey on the memory mechanism of large language model-based agents. *ACM Transactions on Information Systems*, 43(6): 1–47, 2025.

[194] Zheng Zhang, Jie Gao, Ranjodh Singh Dhaliwal, and Toby Jia-Jun Li. 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*, pages 1–30, 2023.

[195] Dora Zhao, Diyi Yang, and Michael S Bernstein. Knoll: Creating a knowledge ecosystem for large language models. In *Proceedings of the 38th Annual ACM Symposium on User Interface Software and Technology*, pages 1–23, 2025.

[196] Zixin Zhao, Damien Masson, Young-Ho Kim, Gerald Penn, and Fanny Chevalier. 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, New York, NY, USA, 2025. Association for Computing Machinery. ISBN 9798400713941. doi: 10.1145/3706598.3713161. URL https://doi.org/10.1145/3706598.3713161.