

Cling: Supporting Cross-Page Coordination with Human-Directed, In-Situ, and Small-Step Automation

Zhicheng Huang
University of California San Diego
La Jolla, California, United States
zhichenghuang@ucsd.edu

Yining Cao
University of California San Diego
La Jolla, California, United States
rimacyn@ucsd.edu

Haijun Xia
University of California San Diego
La Jolla, California, United States
haijunxia@ucsd.edu

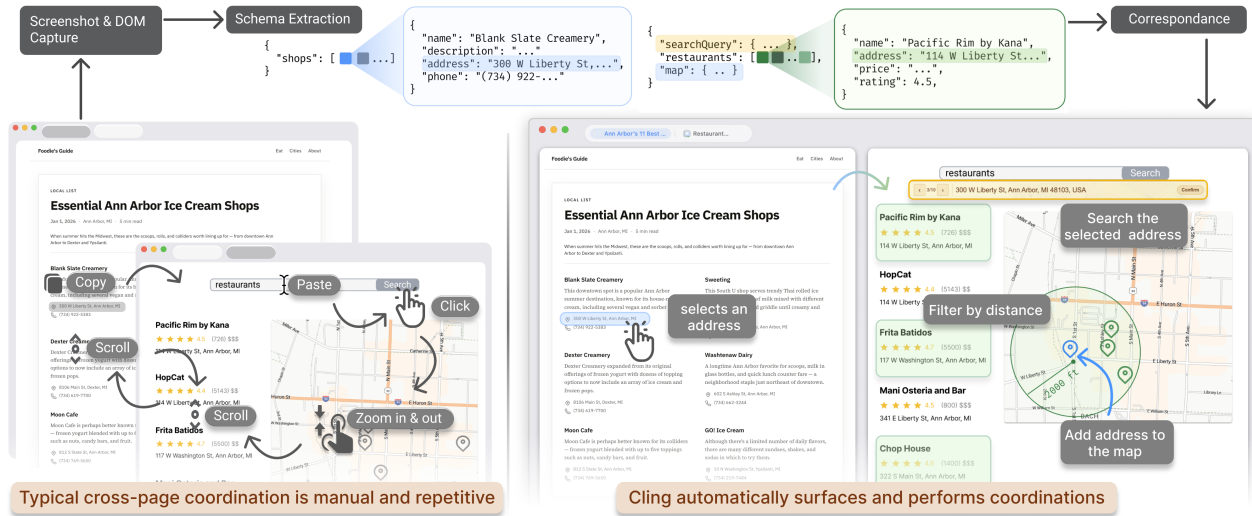


Figure 1: CLING Supports Cross-Page Coordination. Left: typical cross-page coordination is manual and repetitive. Right: CLING automatically surfaces coordination opportunities via schema correspondence and executes them in-situ. Top: CLING extracts structured data models from pages and identifies cross-page correspondences to surface suggested coordination actions.

Abstract

Web-based information work spans independently developed pages, with goals and coordination needs often emerging through browsing. Manual coordination requires repeated low-level actions, yet explicitly specifying operations for automation can interrupt engagement with information that shapes users' next steps. Our approach constructs reusable, schema-based page representations and identifies coordination actions through correspondences across schemas. We instantiate this approach in CLING, which surfaces coordination opportunities in situ for users to select, configure, and execute incrementally while directing the broader task. A technical evaluation demonstrates accurate identification of useful opportunities across diverse domains. A 12-participant study found that CLING reduced task durations by more than half compared with both manual interaction and an end-to-end AI agent, while requiring

61% fewer clicks and 80% fewer tab switches than manual interaction. Its small-step automation supported control and verification as actions unfolded, while accommodating intent as it emerged and evolved through information encountered on the page.

ACM Reference Format:

Zhicheng Huang, Yining Cao, and Haijun Xia. 2026. Cling: Supporting Cross-Page Coordination with Human-Directed, In-Situ, and Small-Step Automation. In . ACM, New York, NY, USA, 17 pages. <https://doi.org/10.1145/nnnnnnnn.nnnnnnnn>

1 Introduction

Web-based information work often spans many pages [33]. When searching for an apartment before relocating, for example, a person may examine a rental listing, view its location on a map, browse its reviews on a community website, compare alternatives, and record selected details in a personal shortlist. In this process, the user needs to frequently transfer information across pages, cross-examine content on different pages, and integrate details from different pages. Such *cross-page coordination* is a recurring pattern of everyday information work, yet it is often tedious and disruptive: users must mentally keep track of numerous disconnected pages while manually performing low-level navigation and information-transfer operations.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
Conference'17, Washington, DC, USA

© 2026 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-XXXX-X/2018/06
<https://doi.org/10.1145/nnnnnnnn.nnnnnnnn>

Supporting cross-page coordination, however, presents both structural and interaction challenges. Structurally, webpages are independently developed and generally lack incentives for interoperability. Shared data structures and protocols can support coordination, but typically only within the boundaries of integrated enterprise applications or ecosystems [29, 58, 66]. Meanwhile, coordination often emerges opportunistically as work unfolds, and therefore, cannot be anticipated and supported with system features [55]. A detail encountered on one page may prompt a new search, introduce a new comparison criterion, or shift the user's priorities. Cross-page coordination can therefore range from a one-off lookup to repeated information transfers, and from a step in a predefined workflow to part of an exploratory activity whose goals and constraints evolve throughout the process.

Client-side web automation tools [16, 17, 72, 73, 89] and recent computer-use agents [8, 67, 68] offer ways to execute operations across page boundaries. As these capabilities advance, an important interaction question is how to make desired automation available within ongoing, human-directed work. Specifying a workflow or delegating a task can be useful when the goals and constraints are already clear. In other situations, users discover what they want to do through inspecting information and making situated [80] and contextual judgments [42, 53, 79, 85, 87]. They may need assistance with an individual lookup or compare corresponding fields while continuing to determine the next step themselves. Supporting this process requires making automation accessible at the moment a need becomes clear, while allowing users to continue directing the activity. Otherwise, users are left to either specify each operation for automation, interrupting their engagement with the participating pages and opportunities to discover their next steps, or to carry them out manually through repeated low-level actions.

We investigate how cross-page coordination can be supported by incrementally surfacing in-situ automation opportunities that users can selectively invoke as needs arise. Our approach uses information correspondences across pages as the basis for identifying potential coordination actions. For example, an address in a listing and a search field on a map page suggest a possible value transfer, while corresponding attributes across listings (e.g., square footage and price) suggest opportunities for cross-referencing or comparison.

Concretely, we model webpage content as a set of entities, each represented by entity schema, runtime values, and references to its original DOM elements. Under this representation, cross-page coordination can be expressed as operations for selecting, filtering, navigating, and transferring information among corresponding entities. Based on this formalization, we define five coordination patterns that capture distinct forms of cross-page correspondence. We use LLMs to construct the schematic representation of each webpage, while deliberately relying on rule-based correspondence detection to surface small-step automation opportunities in real time. As people interact with live webpages, the system dynamically surfaces coordination opportunities derived from cross-page entity correspondences. Depending on their current goals, people can then selectively apply any automation opportunity in-situ to speed though tedious low-level manual operations while retaining control over the direction and progression of the broader task. We instantiate this approach in a prototype system, CLING.

We assess our approach through both technical and user evaluations. In the technical evaluation, we evaluated our technical pipeline (Section 5) across 48 webpages spanning 3 common web task domains. Two human coders annotated all 220 coordination opportunities surfaced by the pipeline, and rated them 92.5% accurate with a mean usefulness of 4.48/5. On average, the page-model extraction took 5.82s across three independent runs.

We further conducted a two-part user study with 12 participants, examine CLING in both structured cross-page coordination and broader exploratory tasks. In structured tasks, CLING combined the efficiency of end-to-end automation (compared with MANUAL, it required 61.1% fewer clicks, 99.6% fewer copy/paste actions, and 80.2% fewer tab switches, while completing tasks 2.07× as fast as MANUAL and 2.14× as END-TO-END AGENT). At the same time, participants reported substantially greater control with CLING than with END-TO-END AGENT (2.83 points higher on a 5-point scale, 95% CI [2.33, 3.33], $p < 0.001$), while reporting a level of control comparable to, and slightly higher than, MANUAL. Moreover, CLING's in-situ and small-step automation allowed participants to inspect and correct mistakes as automation unfold, much as they did during manual work, while accommodating dynamic user intents that emerged and evolved over the course of the task.

In summary, this paper makes the following contributions:

- The formalization of a class of interaction, which we call cross-page coordination as a set of 5 coordination patterns.
- A technical pipeline that constructs reusable entity-based page models and identifies coordination opportunities through page-model correspondence.
- User interfaces and interaction techniques for representing, surfacing, and invoking in-situ, small-step automation.
- CLING, a prototype system that instantiates these technical and interaction approaches.
- Technical and user evaluations that demonstrate the feasibility, effectiveness, and values of supporting cross-page coordination with human-directed, in-situ, and small-step automation

2 Related Work

Information tasks spanning multiple materials and systems often require context switching, information transfer, and coordination across interfaces. Prior work has addressed these demands through integrated environments, interoperability, task automation, and correspondence-driven interaction. We review these approaches to situate CLING's support for human-directed, in situ, small-step coordination across webpages.

2.1 Integrated Workspaces for Distributed Information Tasks

A common strategy for reducing coordination overhead is to build integrated environments for tasks that span multiple materials or processing steps, such as content creation [2, 10, 12], data analysis [24, 59, 71, 81], programming [54, 56], and task management [29, 56, 58, 83]. Although these systems promise an "all-in-one" experience, they typically demand substantial engineering effort and often depend on complex organizational infrastructure.

Integrated environments have also been used to support human-AI collaboration: Buschek et al. integrate diverse AI-powered tools

into a unified writing environment [9], and Cao et al. combine core video-editing structures—such as canvas, document, grid, and timeline—within a single environment to enable human-AI co-creation [12]. In such systems, coordination is often delegated to AI through predefined rules. However, building these environments has traditionally required developers to predefine information sources, structures, and tools. Cao et al. recently reduced this burden with an approach that uses AI to dynamically generate environments in response to evolving user criteria [13].

Browser-based workspaces address similar organizational needs across web materials. Prior work has explored augmenting browsing environments with in-situ support. For example, systems have incorporated query suggestions [69] and note-taking [21, 48, 84] into browsing workflows, transformed tabs into spatial or canvas-based workspaces to better scale browsing activity [37], and externalized users' evolving interests through compositional filters and lenses [15]. Other systems help users externalize mental models such as comparing across collected information [14, 43, 49, 50], and triage during sensemaking [51].

These approaches provide representations for organizing and reasoning across distributed materials. CLING complements this direction by supporting operations between the materials themselves: users can navigate corresponding entities, filter options, and transfer information across the live webpages involved in their work.

2.2 Interoperability and End-User Composition

Information materials and systems that support interoperability generally incur lower correspondence costs, as shared representations mitigate the need for manual transformation and coordination. Nelson's concept of transclusion enables portions of one document to be embedded within another, facilitating the flexible composition of both information and functionality [65]. This idea has been further extended in systems such as Webstrates [39] and subsequent work [40, 41, 74], which support the sharing and synchronization of information and functionality across heterogeneous environments (e.g., web pages and devices). Beaudouin-Lafon et al. proposes a specification language to reason about structural analogies of interfaces [5]. Interoperability has also been pursued through system-level mechanisms, including proprietary approaches such as Microsoft's Object Linking and Embedding (OLE) [57] and Apple's OpenDoc [3, 78].

End-user composition and customization allow people to adapt existing resources to their own needs. Hartmann et al.'s *d.mix* explores programming by sampling existing websites to create new web applications [32]. *Wildcard* exposes application data through a spreadsheet interface, enabling users to customize website behavior [47]. Together, these approaches show how existing data, functionality, and interface structures can become resources for composition. Min et al. explore malleable overview-detail interfaces that let users customize content, composition, and layout, including revealing attributes across views and using AI to transform or generate attributes [63]. These approaches make existing data, functionality, and interface structures available for user-directed adaptation and composition.

However, achieving interoperability typically requires the adoption of shared protocols, data schemas, and frameworks, which introduces substantial development overhead and limits practical adoption. In many cases, such efforts are further constrained within platform-specific ecosystems, resulting in "walled gardens" that hinder integration across heterogeneous systems. More fundamentally, supporting interoperability remains intrinsically challenging, as developers cannot realistically anticipate the full diversity of user needs and usage contexts to define the scope of interoperability [38]. Rather than relying on developers to conform to a single, globally agreed-upon standard, an assumption that is often untenable, we explore an alternative approach: leveraging generative AI to dynamically parse diverse web pages into shared, structured representations, thereby enabling on-demand interoperability across otherwise incompatible systems.

2.3 Task Automation and In-Situ AI Assistance

Traditional web automation systems, including web-scraping scripts [1, 22, 23, 28] and programming-by-demonstration approaches [4, 16, 17, 25, 72, 73], offer reliable, performant ways to navigate websites and collect information at scale. Yet they generally assume well-defined environments and well-specified success criteria, assumptions that often fail in exploratory, open-ended, and uncertain tasks [79, 85–87]. Recent LLM-based agents [8, 52, 67, 68] optimize for task completion by committing to browsing trajectories shaped by model priors and interface cues [27, 77]. In contrast, many information tasks evolve incrementally, motivating in-situ assistance that users can invoke in small steps while deciding how to proceed.

Context-sensitive assistance has a long history in interface agents and mixed-initiative interaction, including anticipating browsing interests, inferring assistance needs, and deciding when to intervene [34, 35, 46]. Recent work revisits these questions: *Fairy Cursor* explores a cursor-based agent for in-the-flow assistance [11], while *Just-In-Time Objectives* infers immediate goals from observed activity to specialize AI-generated tools and responses [44]. CLING builds on this direction by surfacing individual coordination actions as small-step automation that users can selectively execute within ongoing browsing.

Action generalization provides a related precedent for this granularity of support. Prior techniques allow selection, text entry, and clicking to be applied across multiple structurally similar items [45, 60–62]. Zhang et al. recently brought this idea to the web, allowing users to perform the same action simultaneously on multiple selected targets [89]. However, such techniques rely on the involved pages having similar structures, such as in their CSS selectors or DOM placements, an assumption that breaks down for heterogeneous pages. Our approach addresses this limitation by lifting heterogeneous pages to their schemas, where cross-page coordination can be generalized despite structural differences.

2.4 Correspondence-Driven Interaction

A substantial body of research in HCI and information visualization has investigated how semantic correspondences within content can be leveraged to enable coordinated and automated interaction. For example, systems built on time-aligned transcripts—such as *Quick-Cut* [82] and *CrossCast* [88]—exploit temporal alignment between

text and video to support script-based editing workflows. Similarly, Data Particles [10] and CrossData [18] utilize the semantic correspondence between natural language, data operations, and visual encodings to facilitate the authoring of data-driven interactive articles with rich visualizations. In visualization, prior work has also explored leveraging shared data dimensions to support coordinated view management across multiple plots [7, 75]. For web browsing specifically, Contextual Facets transforms webpage elements into controls for filtering and retrieving related pages [55]. Dontcheva et al. enable users to establish relationships between websites, collect related content into personalized cards, and reuse these relationships through search templates [26].

By operationalizing these correspondences, such systems can automate a range of otherwise tedious low-level operations, including navigation, view coordination, content transfer, and transformation. Our work aligns with this line of research in leveraging inherent correspondences within information artifacts. However, rather than relying on developer-specified mappings, we target open-ended web environments where such correspondences are not predefined. We introduce an approach that leverages generative AI to parse heterogeneous web pages into structured schemas and employs heuristic strategies to infer correspondences at interaction time, enabling real-time support for navigation and information transformation across heterogeneous pages.

3 Example Use Case

Polly, staff at a university, is co-organizing a panel on AI's impact on education. She is responsible for identifying potential participants. The organizing team maintains a shared spreadsheet listing candidate invitees, including their name, affiliation, contact information, and a brief justification for inclusion.

To populate this spreadsheet, Polly needs to visit faculty directories from different departments, looks up their profiles on Google Scholar, reviews their publications, and decides whether they are a good fit. In a typical workflow, for each faculty member, she first checks whether they are already in the spreadsheet by copying their name and searching for it (e.g., using `cmd+F`) in another tab. If not found, she needs to then switch to Google Scholar, copies the name into the search box, submits the query, and reviews the profile. When appropriate, Polly needs to manually copy the name, affiliation, and email back into the spreadsheet, carefully aligning each field with the correct column in the spreadsheet. If the faculty member is already listed, Polly still needs to verify whether the contact information is up to date, requiring her to locate the corresponding row and cross-reference multiple sources. She repeats this process for each faculty member.

With CLING, Polly first places the faculty directory, Google Scholar, and the spreadsheet side by side. When Polly selects a faculty name on the directory page, CLING indicates whether the person is already in the spreadsheet. If not, CLING suggests transferring the selected name into the Google Scholar search field and submitting the query. After reviewing the publications, Polly decides to include the faculty member and selects "Add to spreadsheet." CLING then aggregates and reformats the relevant information into a new row matching the schema. After previewing the entry, Polly confirms the insertion. If the faculty member is already in the

spreadsheet, CLING surfaces a "Go to" action that navigates directly to the corresponding row. Polly can then quickly verify and update contact information by cross-referencing the directory and Google Scholar pages, without manually searching or aligning fields. Using CLING's navigation widget, Polly can step to the next researcher while preserving the coordination context across the pages. At each step, CLING updates the relevant actions—for example, whether the researcher is already in the spreadsheet, where their row is, and by issuing the next Google Scholar search automatically. Polly can thus move continuously through researchers without repeating the same search, tab-switching, and viewport-alignment work for each person, allowing her to stay focused on evaluating and selecting participants for the panel.

4 Cross-Page Coordination Patterns

Supporting cross-page coordination requires identifying how information on one page relates to information and actions on another. We use these correspondences as a basis for a catalog of five coordination patterns, each mapping a configuration of page models and their correspondences to applicable coordination actions. We first define the page model, then describe each pattern.

4.1 Pages Viewed as Databases

We conceptualize each page as a *structured representation that encodes the underlying data with a specific schema*. While not all web pages are driven by relational databases, most could be viewed from a database perspective, where each page contains entities $\mathcal{E}$ (i.e., rows), such as products and restaurants; and attributes $\mathcal{A}$ (i.e., columns), such as price and location. With this perspective, we can describe cross-page coordination with the following formalization:

- **Source Page:** a dataset $D_s \subseteq \mathcal{E}_s \times \mathcal{A}_s$ under schema S_s
- **Target Page:** a dataset $D_t \subseteq \mathcal{E}_t \times \mathcal{A}_t$ under schema S_t
- **Cross-page Coordination:** an operation mapping a user's expressed interest on the source page to the target page.

4.2 Coordinations as Database Operations

Taking this database-oriented view, we can comprehensively model cross-page coordination by examining the possible operations across two databases. This leads to five primitive coordination patterns. More complex coordination behaviors can be understood as compositions of these operations. In practice, a single task often involves multiple patterns simultaneously or sequentially.

C1. Entity Cross-Reference ($\mathcal{E}_s \leftrightarrow \mathcal{E}_t$). The user's interest is at the entity level, analogous to selecting a row in a database table. Coordination establishes a correspondence between entities in $\mathcal{D}_s$ and $\mathcal{D}_t$ via primary-key-based joins. This includes one-to-one mappings (e.g., selecting a specific flight in a search result highlights the same flight in a booking tab) and one-to-many relationships (e.g., selecting a faculty member in a directory navigates to a list of all their publications in a proceedings). At the interface level, this appears as selecting an entity in one page to focus, highlight, or navigate to related instances in another.

C2. Attribute Cross-Reference ($\mathcal{A}_s \leftrightarrow \mathcal{A}_t$). In this case, the user's interest is not in a particular entity, but in a specific attribute dimension, analogous to selecting a particular property in a dataset.

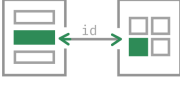
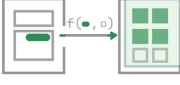
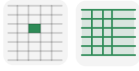
Coordination Pattern	Definition	Database Analogy	Examples
C1. Entity Cross-Reference 	Source entity triggers focus, highlight, or navigation to the corresponding entity on target page.	 Map the selected row in source database to matching row(s) in target database.	Flight booking: Selecting a flight in search results (S) highlights the same flight on a booking page (T).
C2. Attribute Cross-Reference 	Source attribute highlights or selects the corresponding attribute on target page.	 Map the selected column in source database to the corresponding column in target database.	Price comparison: Selecting the price field on Best Buy (S) highlights the corresponding price field on Amazon (T).
C3. Attribute Filter 	The value of a source entity from source page is used as a predicate to filter entities on target page	 Use a value from source database as a condition to filter rows in target database	Shopping within budget: Selecting a budget in a shopping worksheet (S) filters products on a retail page (T) to those within budget.
C4. Attribute Transfer 	Source attribute is transferred to a UI input that can accept the attribute on the target page.	 Attribute value used as part of the search query to the target database	Syncing travel dates: Setting the check-in date on a hotel booking page (S) sets the same date on a tour booking page (T).
C5. Entity Transfer 	Source entity is transferred into the target page adhering to its schema.	 Use a value from source database to fill a query parameter and generate a new result table on target database.	Reading list curation: Selecting a paper in proceedings (S) generates a reading-list entry in a research notebook (T) following the same structure as other papers.

Figure 2: Cross-Page Coordination Patterns. Viewing web pages as databases, we categorize cross-page coordination into Entity Cross-Reference (C1), Attribute Cross-Reference (C2), Attribute Filter (C3), Attribute Transfer (C4), and Entity Transfer (C5).

Selecting an attribute in D_s establishes semantic alignment with a corresponding attribute in D_t . In many cases, attribute names may differ across schemas (e.g., price and cost), coordination operates at the level of shared meaning rather than literal key matching. At the interface level, this may appear as highlighting an attribute (e.g., price or rating).

C3. Attribute Filter ($\mathcal{A}_s \rightarrow \mathcal{E}_t$). This pattern corresponds to a selection operation parameterized by a value from the source page. Formally, a value from an attribute in D_s is interpreted as a predicate over D_t , selecting a subset of the target dataset. Unlike C2, C3 uses the source value to filter D_t rather than identify corresponding target attributes.

C4. Attribute Transfer ($Q_t : \mathcal{A}_s \rightarrow \mathcal{D}_t$). Interfaces often expose *query-construction fields* that dynamically generate datasets (e.g., search bars, date pickers). In such cases, the target page defines a parameterized query template whose results are materialized upon instantiation. A value derived from the source page may populate such a field, thereby generating a new dataset. For example, selecting a faculty member’s name may automatically populate a search field that retrieves publications from Google Scholar.

C5. Entity Transfer ($S_t : \mathcal{E}_s \rightarrow \mathcal{E}_t$). Entities from D_s is transformed and inserted into a different schema S_t . Unlike direct correspondence (C1), the selected entity from D_s does not already exist in D_t , thus requires re-encoded under the target schema. At the

interface level, this occurs when selected entities are extracted and reformatted into a new structural representation such as reformatting paper metadata into BibTeX format and adding deadlines from a course syllabus to a calendar.

These five patterns arise from a structural analysis of cross-page coordination under a relational abstraction, capturing user interest at the level of entities (C1, C5), attributes (C2, C3), and instantiated values (C3, C4).

5 Technical Pipeline

Building on the coordination patterns, we implement a technical pipeline to construct schema-based page representations and derives coordination opportunities from them, as illustrated in Fig. 3.

For each page, the system captures and cleans its DOM to construct a page model comprising entities and their relationships. HTML structures varies substantially independently developer sites: generic tags such as `<div>` are often repurposed for arbitrary interface semantics, accessibility annotations are inconsistent, and markup conventions differ widely across organizations. This heterogeneity makes it brittle to construct page models with rule-based approaches. We therefore use LLM to extract coordination-salient semantics from the underlying markup, transforming heterogenous webpages into consistent page models.

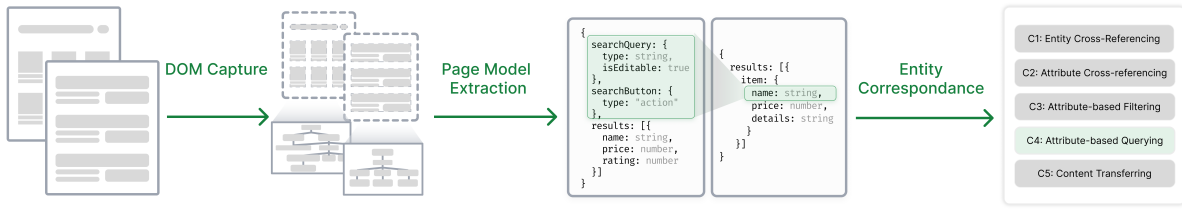


Figure 3: CLING’s Technical Pipeline Overview. For each page, the pipeline (1) captures its DOM, and (2) extracts its page model. It then uses page model correspondences to surface coordination opportunities against the five coordination patterns.

The system then identifies correspondences across page models and maps them to opportunities for surfacing the coordination patterns outlined in Section 4. We adopt a heuristic-based approach, which is beneficial for real-time, interactive use. Specifically, the system uses a set of lightweight signals to infer cross-page correspondences to surface coordination opportunities.

5.1 Per-Page Representation Construction

We represent each page as a set of *entities*: units of information that users interpret and operate on.

5.1.1 Entity model. Each entity consists of a *schema*, a runtime value, and *reference* to the page’s DOM.

Schema: Each entity’s schema specifies its data structure and coordination-relevant interaction properties.

- *Data.* The schema specifies the entity’s *type*. Types may be primitive (e.g., string, date) or composite (e.g., object, list); we refer to named fields of a composite entity as *attributes* (e.g., price within a product entity).
- *Interaction.* The schema encodes properties and roles that determine how an entity can participate in coordination. *isEditable* and *isExtendable* indicate whether values can be modified or added through the interface, *isIdentifier* marks attributes that uniquely identify their enclosing entities, and *role* distinguishes data-bearing entities from action entities (e.g., submit buttons).

Runtime Value: The runtime value contains the data currently associated with the entity. For lists, the system infers one shared schema and instantiates it with the values extracted from each list item.

Reference: Each entity retains one or more handles to the interface objects through which it is presented or manipulated. For webpages, these handles are references to DOM elements. Reference allows the system to map user interactions on the live page to entities and to execute coordinations back to their corresponding interface objects.

5.1.2 Page Model Extraction. Once a page loads, the system scans its live DOM and CSS selectors to identify salient regions (e.g., lists of similarly-structured elements, standalone elements with many nested child elements, forms, and submit buttons), while removing irrelevant elements (e.g., non-content `<script>` and decorative style tags).

It characterizes each salient regions and selects representative items from each list based on how common and structurally rich they are (i.e., a typical item with the most common structure and

examples of other variants that contain substantial text, fields, or links). An LLM reasons over such regions to infer the page’s purpose and determine whether each region has potential to participate in cross-page coordination.

For each retained list, the LLM infers a shared item schema that captures its common attributes and meaningful structural variations. For each list of elements, the LLM derives a schema that unionizes common attributes with optional attributes that are only seen over some list elements. The system then maps each schema attribute to the corresponding DOM element within every list item. It uses these mappings to extract the item’s runtime values and retains references to the matched DOM elements. Standalone entities, forms, and actions are interpreted separately and incorporated into the same page model with their runtime values and DOM references.

Page-model extraction is performed independently for each page, allowing pages to be added to or removed from a cross-page coordination without reprocessing other pages. Completed page models are cached by URL. For paginated pages that share the same list structure, the system can reuse the inferred list schema while extracting the runtime values and DOM references of each page anew.

5.2 Identify Coordination Opportunities

With page models of all participating webpages, the system looks for each coordination pattern we formalized (Section 4). We outline the heuristics for recognizing these patterns below.

C1. Entity Cross-Reference. C1 identifies entities on the target page that refer to the same underlying entity.

$H_1(e_s, \mathcal{D}_t)$ returns entities $e_t \in \mathcal{D}_t$ satisfying both conditions: (1) e_s and e_t have the same type or semantically adjacent attribute names; (2) They share at least one matched attribute pair (a_s, a_t) with similar values, where at least one of the two attributes is an identifier.

C2. Attribute Cross-Reference. C2 maps a source attribute to corresponding attributes across on the target page.

$H_2(a_s, \mathcal{D}_t)$ returns target attributes a_t in $\mathcal{D}_t$ satisfying both conditions: (1) a_s and a_t have semantically adjacent attribute names; (2) Each attribute occurs across multiple entities in their respective schemas.

C3. Attribute Filter. C3 uses a source attribute as a constraint for filtering entities on another page. Let $S_t(v_s, v_t)$ denote a type-specific comparison function that scores the relationship between

a source value v_s and a target value v_t . C3 uses a configurable threshold τ_t to convert this score into a filter predicate.

$H_3(a_s, \mathcal{D}_t)$ first identifies matching attributes $M = H_2(a_s, \mathcal{D}_t)$, then returns entities $e_t \in \mathcal{D}_t$ satisfying both conditions:

- (1) e_t contains an attribute $a_t \in M$; (2) For that attribute, $S_t(\text{val}(a_s), \text{val}(a_t)) \geq \tau_t$, where $\text{val}(a)$ denotes the value of attribute a .

C4. Attribute Transfer. C4 transfers a source attribute's value into a compatible editable field on the target page.

$H_4(a_s, Q_t)$ returns editable fields $q_t \in Q_t$ satisfying at least one condition: (1) a_s and q_t are type-compatible; (2) a_s and q_t have semantically adjacent attribute names.

C5. Entity Transfer. C5 transfers a source entity into an extendable target entity, such as a spreadsheet, calendar, or map. Unlike C1, it adapts the source to the target structure rather than matching an existing target entity.

$H_5(e_s, S_t)$ returns extendable target structures $s_t \in S_t$ for which e_s can be transformed into a format compatible with the target schema.

H_5 surfaces target structures that can accept a transformed version of the selected source entity e_s . When invoked, the system supplies source and target context—including the selected entity, its parent, and the surrounding target entities—to transform e_s into a representation compatible with s_t .

5.3 Coordination Generalization

Coordination opportunities are first identified between a source-target entity pair. The system then generalizes the pairwise coordination to analogous entities in the source and target schemas via structural relationships such as array membership, entity containment, and relative paths within composite entities, enabling reuse of the established coordination for other suitable pairs.

5.4 Incremental Entity Updates

After constructing a page model, the system keeps it synchronized with the live page by monitoring the DOM elements that ground its entities. When the DOM changes, these references identify the affected portion of the page model, allowing the system to update it without re-extracting the page.

For an updated list item, the system reapplies the list's existing schema to the affected DOM subtree. The item's *identifier* attribute determines whether the result updates an existing entity or creates a new one, and removing the subtree removes the corresponding entity. For an updated object attribute, the system parses the updated content using its existing type and updates only that attribute. If its referent is replaced, the system reconnects it to the replacement element. If the updated content cannot be interpreted as the attribute's existing type, parses it as a new entity and removes the old entity.

5.5 Implementation

The pipeline is implemented in TypeScript as part of an Electron application. The current implementation uses `gemini-3.5-flash-lite`

by default. Semantic comparisons between entities are also performed locally using `all-MiniLM-L6-v2` with cosine similarity.

6 CLING: User Interfaces and Interactions

We designed CLING to reify cross-page coordination by making implicit relationships between heterogeneous pages explicit. Fig. 4 illustrates the end-to-end flow from intent specification to how coordination opportunities are surfaced and executed.

6.1 Specifying Coordination Intent and Surfacing Coordination Opportunities

CLING allows users to place two pages into side-by-side views to indicate interest in coordinating between them. For any pair of pages with extracted page models, CLING does not speculatively correspond compatible entities to compute all coordination opportunities because it would warrant exhaustively exploring a large correspondence space. Instead, it waits for users to specify their coordination intent by `cmd+clicking` a DOM element on the page. CLING uses the bidirectional reference to the page model to identify its corresponding entity as the source entity. Then, CLING runs the correspondence heuristics to identify all possible coordination opportunities involving the specified source entity.

Once coordination opportunities are surfaced following a user specification, CLING presents them in a popover menu next to the selected DOM element (Fig. 4a), for users to select the intended coordination. When proposed, each coordination pattern in Fig. 2 deterministically maps to one option on the popover menu. Some coordination patterns allow the user to further refine the coordination requirement on the popover. For example, when attribute filter (C3) is proposed (Fig. 4b), the user can specify the constraints with more concrete criterion (e.g., "total price below 500") in natural language.

6.2 Executing and Updating Coordination

When users accepts a coordination opportunity, CLING parses the runtime value for the source entity and the selected coordination pattern into a script that queries the target page's page model. Running this script over the target page model executes the selected coordination, and the the bidirectional references encoded by the page model is used to present coordination progress and results in-situ.

CLING presents the currently active coordination as a "binder" between the pair of participating pages (Fig. 4c). The binder displays a compact chip displaying the coordination pattern, and optionally the current filtering constraints whose numeric values are editable. The binder also supports navigation among all current source and target entities inferred through binding generalization Section 5.3. Stepping to a different target entity auto-scrolls it into view. Stepping to a different target or updating the numeric value of the constraint updates the script used to query the target page's page model and reruns over it to updates the latest satisfying target entities.

For coordination patterns that modify the target page (e.g., changing values, inserting content, or navigating to another page), CLING uses a two-step execution flow. It first shows an overlay next to

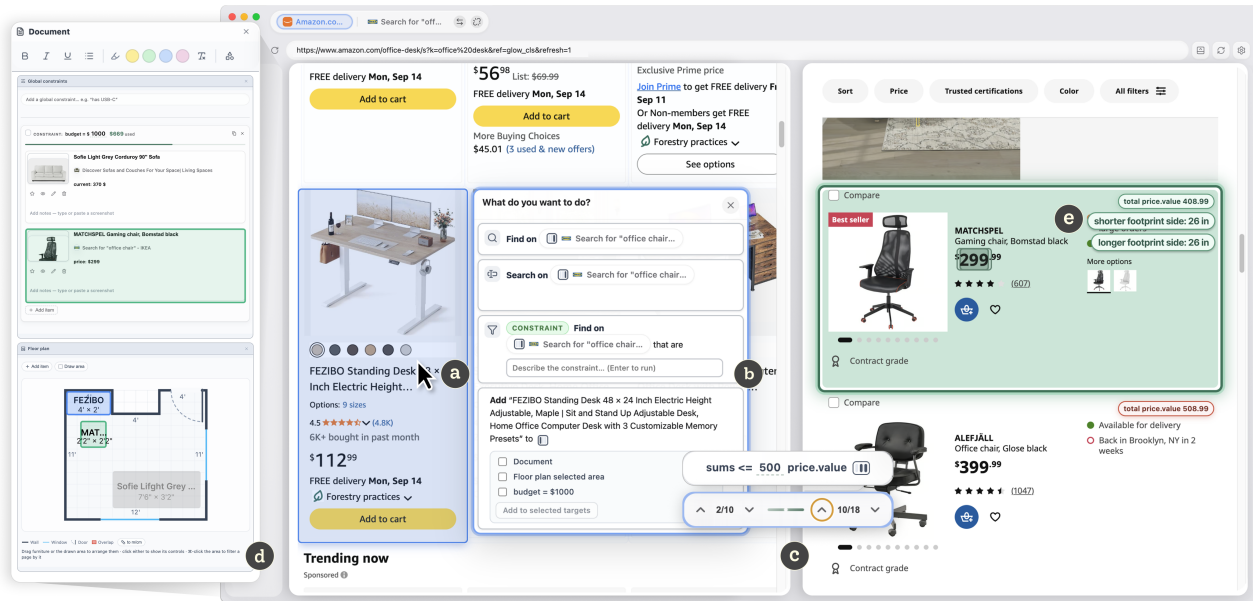


Figure 4: Using CLING. The user selects an desk as a source by cmd+clicking it (a). CLING surfaces coordination opportunities in a pop-up panel, where the user can select actions and specify their criteria (b). Once accepted, active coordination actions appear in a “binder” between the participating pages (c). To extend coordination across additional pages and information sources, CLING provides a side panel for saving and reusing actions (d) and retrieves external attributes to supplement the page model (e).

the target entity’s corresponding DOM element to preview the effect of the change, and ask the user to confirm before committing the action. When the user confirms the same type of coordination for the same source entity once, subsequent confirmations can be optionally skipped. For coordination patterns that only update the target page’s presentation (e.g., highlighting corresponding entities or scrolling them into view), CLING applies the results immediately.

6.3 Extending the Scope of Coordination

CLING extends coordination support through action reuse across additional pages and retrieval of attributes beyond the initially extracted page models.

6.3.1 Reusing Actions Across Pages. Cross-page coordination in web tasks often involves more than two pages. To extend coordination beyond two pages arranged side by side, CLING allows users to save pairwise coordination actions and reuse them across additional pages. Saved actions, such as those applying a shared budget or spatial constraints from a floor plan, appear in a side panel (Fig. 4d), where users can selectively apply them to new page pairs.

This provides a visual way to reuse actions across pages and incrementally apply multiple constraints. For example, when selecting furniture under budget and spatial constraints, users can save actions that check prices against a shared budget and dimensions against available space in a floor plan. As they browse additional listings, they can reapply these actions to assess each option against both constraints without repeatedly establishing the same cross-page relationships.

6.3.2 Retrieving External Attributes. Coordination actions may depend on attributes that are not available on the existing extraction of the page model. For example, when selecting furniture, the dimensions the user looks for may be nested in the product detail pages. Likewise, when filtering a houseplant catalog, a user may specify a “cat-safe” criterion that requires searching for general knowledge beyond the catalog. To retrieve these missing attributes, CLING uses the Gemini API with Google Search grounding [30] to obtain the requested attribute values for relevant entities on the page (Fig. 4e).

6.4 Example Coordination Flows with CLING

We illustrate the user-facing interaction flow of CLING through four examples, covering all coordination patterns (Section 4).

Cross-referencing schools across university rankings (C1). Jackie puts a university ranking page and a page listing top universities for statistics major side by side. She cmd-clicks the name of a school on the general ranking page, and CLING highlights the C1 coordination chip in the binder and shows the school was found on the major ranking. Jackie then clicks *Go To* on the chip, and CLING scrolls the matching school on the other page into view so she can inspect its details. As she works through schools she is considering applying to, she uses the navigation widget to move through the ranking list, skipping entries for which CLING reports *No Match*.

Filtering restaurants by proximity to a hotel (C2/C3). Lily places a hotel listing page beside a restaurant listing page. She cmd-clicks a hotel’s address, and CLING activates the C2/C3 coordination chip on the binder, highlighting corresponding address

values on the restaurant page. Hovering over the chip reveals filters for location-typed attributes. She selects a radius filter and sets it to 1 mile. The chip then displays “Within 1 mile,” and only restaurant addresses within 1 mile of the selected hotel remain highlighted. Lily expands the navigation widget into a list showing each filtered restaurant’s distance from the hotel and clicks the closest one to scroll it into view. When she moves to the next hotel, CLING automatically reapplies the same filter relative to the new hotel.

Querying winery availability from forecasted dates (C4). Nina places a weather page beside a winery booking page. She browses the forecast for a sunny Saturday and cmd-clicks a date. CLING activates the C4 coordination chip in the binder, indicating that the selected date can transfer to a compatible date picker on the booking page. Invoking the chip opens an overlay next to the date picker that previews the value to be inserted. After Nina confirms, CLING populates the field and updates the booking page to show available tasting slots for that day. When she selects another date, CLING reuses the same confirmation preference and directly updates the date picker to show availability for the new date.

Curating Reading List During Literature Review (C5). Russell places a conference proceedings page beside his research notebook, a markdown editor. As he goes through the proceedings list, he cmd-clicks a relevant paper. CLING highlights the C5 coordination chip in the binder, indicating that the selected paper can be transformed into a structure compatible with the notebook, and shows an overlay previewing the insertion. After he confirms, CLING inserts a new entry with the same structure, including the title, venue, and abstract. Russell then continues through the proceedings, transferring additional relevant papers the same way.

7 Technical Evaluation

We conducted a technical evaluation of CLING’s end-to-end pipeline by first extracting page schemas and then applying schema correspondence to surface cross-page coordination opportunities. Although the evaluation focuses on surfaced opportunities, it also reflects schema extraction quality, as downstream coordination depends on it. Our evaluation was guided by the following research questions:

RQ1: How *accurate* are the coordination opportunities surfaced by the pipeline?

RQ2: How *useful* are the surfaced coordination opportunities in the context of the pages being coordinated?

7.1 Methodology

We constructed an evaluation dataset spanning three task domains commonly used in prior evaluations of web-based systems: *travel planning* [15, 36, 64, 70], *shopping* [14, 43, 51], and *information work* [17, 43, 48, 50, 76]. Within each domain, we identified five representative tasks. For each task, we selected three web pages that were central to the completion of the task and that exhibited diverse interface structures. We then combined these pages into pairs, yielding three pairs per task. In total, this produced 45 page pairs. To ensure stable evaluation conditions, we archived each page by downloading its HTML and associated assets and serving them locally, thereby avoiding nondeterministic changes in live

web content. We then ran the schema extraction pipeline for each archived page using gemini-2.5-flash-lite [20].

For each page pair, we sampled representative source entities from the extracted schemas by selecting one instance of each distinct entity type on the page. For each sampled source entity, we ran the pipeline to surface all possible coordination opportunities with the paired page. When appropriate, we manually added reasonable task-level constraints, such as a shared budget across two shopping pages, to reflect realistic coordination scenarios. This process yielded 220 coordination opportunities. Across the 5 patterns, the dataset included 17 Entity Cross-reference (C1), 87 Attribute Cross-reference (C2), 30 Attribute Filter (C3), 71 Attribute Transfer (C4), and 15 Entity Transfer (C5). The large numbers of C2 and C4, compared to the others, are expected, as entities often contain multiple attributes. Two external human coders independently annotated each surfaced coordination opportunity for **accuracy**, whether the pipeline correctly identified the opportunity, and **usefulness**, how helpful the opportunity would be in the context of the paired pages and task scenario. Usefulness was rated on a 5-point ordinal scale.

Our accuracy measure evaluates the correctness of surfaced opportunities, rather than recall over all potentially useful coordinations. Measuring recall would require a ground truth of all salient coordination opportunities across all possible tasks involving a page pair, which is difficult to define because relevance depends on users’ intents, constraints, and workflows.

To assess whether the technical pipeline supports interactive use during browsing, we measured entity model extraction time for each of the 48 webpages in the evaluation suite over three independent runs.

7.2 Results

For accuracy, the two coders achieved high inter-rater reliability across the 220 evaluated coordination opportunities. We report Gwet’s AC1 [31], as very high raw agreement in several categories can make measures such as Cohen’s κ unstable [19]. Overall, coders achieved Gwet’s AC1 = 0.910 with 92.3% observed agreement, indicating substantial agreement. We did not seek consensus on usefulness because, unlike correctness, it was intentionally rated with respect to each coder’s own lived experience.

Accuracy of Surfaced Opportunities. Overall, the pipeline surfaced coordination opportunities with high accuracy, achieving an average accuracy of **92.5%** across all 220 opportunities. Broken down by coordination pattern type, the average accuracies were 94.1%, 92.5%, 85.0%, 94.4%, and 96.7% for C1 through C5, respectively.

Usefulness of Surfaced Opportunities. Surfaced opportunities were generally rated as useful, with a mean score of **4.48/5**. Mean ratings for C1–C5 were 4.94, 4.08, 4.57, 4.72, and 4.93, respectively.

Entity Model Extraction Latency. Across individual pages, the mean extraction time was 5.82 seconds (95% CI: 5.31–6.33 seconds), with a median of 5.02 seconds and a 95th percentile of 11.46 seconds.

7.3 Takeaways

The results suggest that the pipeline can reliably surface coordination opportunities that are both accurate and useful across coordination patterns. Accuracy was high overall and strongest for C1 and C5, indicating that the pipeline performs especially well when coordinating at the entity-level. Usefulness was also high overall, but varied more across coders and coordination patterns, suggesting that perceived value depends more on task context and prior habits than correctness does.

Most inaccuracies stemmed from attribute misalignment. Schema extraction sometimes retained under-specified labels such as name, even when multiple entity types on the same page (e.g., foods and restaurants) exposed that label, causing incorrect matches. Repeated entities on a page also did not always share identical attribute structures: some items included additional or variant attributes, such as discounted to alongside price. This could cause the pipeline to miss the intended correspondence or coordinate on an available but less relevant attribute (e.g., using the listed rather than discounted price when applying a budget constraint during shopping). C3 was further affected by our use of a single semantic similarity threshold, which could be either too strict (e.g., failing to match product titles when one page omitted a brand name) or too permissive (e.g., matching address strings that shared city, state, and ZIP code despite referring to different locations).

Variation in usefulness ratings reinforces our decision to present coordination opportunities in a lightweight, non-distracting manner rather than as a persistent or intrusive feature. This is especially important for C2, which is inexpensive to propose and thus easy to surface frequently, but may not always be useful for the user.

8 User Study

We conducted a two-part user study of CLING, examining its effectiveness in structured cross-page coordination tasks and its role in broader exploratory information activities. We sought to answer the following research questions:

RQ1: How effectively and efficiently does CLING support cross-page coordination?

RQ2: What are the *values* and *limitations* do people experience with CLING's in-situ, small-step automation during cross-page tasks?

RQ3: How do people incorporate cross-page coordination support into broader exploratory tasks?

8.1 Participants

We recruited 12 participants (6 female, 6 male, aged 20–28) through internal communication channels within a large public university. Participants included undergraduate and Ph.D. students. All participants use browsers on a daily bases and find themselves using multiple pages to complete a single task many times a day. All sessions were conducted in person and lasted approximately 100 minutes. Each session was screen- and audio-recorded. Participants received a 35 USD Amazon gift card as compensation.

8.2 Prototype Conditions

All three conditions used the same underlying Electron/Chromium browser shell, which supported side-by-side page views and provided lightweight spreadsheet, calculator, and rich-text editor tools. The conditions differed only in the coordination and automation support layered on top of this common environment.

MANUAL. The manual condition used the base browser shell without any CLING-specific cross-page coordination or AI automation support.

END-TO-END AGENT. The END-TO-END AGENT agentic condition augmented the same browser shell with an AI assistant implemented using Browser Use v0.13.8¹, an open-source framework for connecting LLMs to web browsers, with GPT-5.6 Sol as the underlying model. To provide a strong browser-automation baseline, we enabled DOM-based page inspection, bounded JavaScript execution for batched edits, and a scratchpad for tracking multi-step tasks. The agent was instructed to ground task-specific claims in evidence observed during the task rather than rely solely on pretrained knowledge.

CLING. The CLING condition augmented the same browser shell with the technical backend described in Section 5 and the cross-page coordination interactions described in Section 6.

8.3 Procedure

The study consisted of the following phases.

Introduction (5 mins). Participants were introduced to study context and procedure; informed consent was obtained.

Tutorial (10 min). Participants went through structured tutorial for all three prototypes. For each tutorial, the experimenter first demonstrated the relevant interactions using a separate example task, after which participants completed a brief hands-on practice task and could ask clarification questions. Practice materials were not reused in the measured tasks.

Part 1: Structured Cross-Page Coordination (40 min). Participants completed three structured cross-page coordination tasks under each of the three interface conditions: MANUAL, END-TO-END AGENT, and CLING. We used a within-subject design where interface and task orders were completely counterbalanced across the 12 participants. Each task had three matched content variants, which were counterbalanced across interface conditions. For each participant, task order was held constant across interface conditions. The tasks were designed around representative cross-page coordination episodes spanning all five coordination patterns introduced in Section 4, while retaining objectively specified outcomes that allowed us to measure task completion, correctness and completeness, and completion time. In the END-TO-END AGENT condition, participants were instructed to use the AI assistant as the primary means of interacting with webpages, asking the agent to carry out page actions on their behalf whenever possible. They remained free to inspect the pages themselves to understand the information or verify the agent's work.

¹<https://pypi.org/project/browser-use/0.13.8/>

- **Cross-Page Find (C1, C2).** Cross-reference a spreadsheet of PC cases inventory with an editorial recommendation guide and mark which cases in the spreadsheet are recommended by the guide.
- **Cross-Page Filter (C2, C3).** Cross-reference landmarks and restaurants across two city guides, identify landmark–restaurant pairs satisfying a specified distance constraint, and report the qualifying pair that best satisfies the objective selection criterion. In the MANUAL condition, participants were given access to Google Maps, and END-TO-END AGENT was given Google Maps API to query distance.
- **Cross-Page Transfer (C4, C5).** For each paper on a conference proceedings page, look it up on an archived Semantic Scholar interface, identify papers satisfying a citation threshold, and transfer the qualifying papers and their metadata to a spreadsheet.

To ensure stable environments, all pages were archived and served locally and their schemas were loaded from cached extractions. Based on pilot sessions, we scoped each task to 12–16 coordination instances (e.g., 12 landmark–restaurant pairs), balancing sufficient repetition with a reasonable chance of completion in the manual condition.

After completing all tasks using one prototype, participants completed a questionnaire with 5-point Likert-scale related to their experience of working on the tasks using the prototype.

Break (5 min). Participants took a 5-minute break.

Part 2 Tutorial (5 min). Before Part 2, participants received a brief guided tutorial on the additional capabilities used in the broader furnishing tasks. The experimenter first introduced the interactive floor-plan widget embedded in the task document for both END-TO-END AGENT and CLING, which allowed participants or agent to add dimensioned representations of selected furniture and arrange them spatially. For END-TO-END AGENT, participants were informed that the assistant could additionally draw on pretrained knowledge and Web search to support the broader task. For CLING, the experimenter demonstrated persisting and reusing coordination beyond page pairs (Section 6.3.1). Participants briefly practiced these features using separate example materials before beginning the measured tasks.

Open-ended Floor Planning Tasks 20 min. Participants completed one 10-minute furnishing task with END-TO-END AGENT and one with CLING, with condition order and room assignment counterbalanced. In the bedroom variant, participants were asked to select a bed frame and at least one additional major item; in the living-room variant, they were asked to select a sofa and at least one additional major item. Beyond these requirements, participants were free to determine their final choices, evaluation criteria, ordering of actions, and which pages to visit, subject only to the constraint that their selected furniture fit within the provided floor plan.

In each condition, participants could choose whether and when to use the available support—CLING or the AI assistant—and could otherwise browse manually. Participants recorded their current selections in the provided document. They were not required to complete the furnishing plan within the 10-minute period, but were asked to work toward a plan they would be satisfied with.

Questionnaire and interview (15 min). After each Part 2 trial, participants completed a brief questionnaire with 5-point Likert-scale items about their experience using the prototype. After both trials completed, we ended with a semi-structured interview for additional qualitative feedback.

8.4 Results and Findings

CLING Offers Effective and Efficient Cross-Page Coordination Support.

Fig. 5 shows that, across the three Part 1 tasks, CLING substantially reduced the primary manual operations required to complete the workflows. Compared with MANUAL, CLING required 61.1% fewer clicks (38.7 vs. 99.4 per task; paired-bootstrap 95% CI for the reduction = [49.5%, 68.6%]), 76.6% fewer typed characters (10.9 vs. 46.7; 95% CI = [56.1%, 87.2%]), 99.6% fewer copy/paste actions (0.1 vs. 29.3; 95% CI = [99.1%, 100.0%]), and 80.2% fewer tab switches (5.8 vs. 29.3; 95% CI = [75.9%, 84.1%]). END-TO-END AGENT required even fewer clicks (10.4 per task) and tab switches (1.4 per task), but shifted substantial manual effort to prompt construction as participants entered an average of 381.6 characters per task with END-TO-END AGENT (95% CI = [307.2, 467.3]), compared with 10.9 with CLING (95% CI = [6.1, 16.6]) and 46.7 with MANUAL (95% CI = [26.9, 69.1]).

These operational savings translated into shorter task durations, as Fig. 6 shows. Using the 240-s timeout threshold for the sensitivity analysis, the overall participant-level geometric mean duration was 83.0 s with CLING (bootstrap 95% CI = [67.7, 105.3]), compared with 171.8 s for MANUAL (95% CI = [151.7, 194.4]) and 177.3 s for END-TO-END AGENT (95% CI = [164.4, 192.8]). CLING therefore required 51.7% less time than MANUAL (paired-bootstrap 95% CI = [41.8%, 59.3%]) and 53.2% less time than END-TO-END AGENT (95% CI = [40.2%, 62.0%]). We found a significant overall effect of technique on duration (Friedman $\chi^2(2) = 15.17$, $p < 0.001$, Kendall's $W = 0.63$). Bonferroni-adjusted exact paired Wilcoxon comparisons showed that CLING was faster overall than both MANUAL ($p_B = 0.001$, $r_{tb} = 1.00$) and END-TO-END AGENT ($p_B = 0.004$, $r_{tb} = 0.95$), while MANUAL and END-TO-END AGENT did not differ ($p_B = 1.000$). At the task level, CLING was faster than MANUAL in all three tasks ($p_B = 0.015$, .003, and .006) and faster than END-TO-END AGENT in Tasks 2 and 3 (both $p_B = 0.003$), but not significantly so in Task 1 ($p_B = 0.157$). Only 2 of 36 CLING trials timed out, compared with 7 of 36 MANUAL trials and 11 of 36 END-TO-END AGENT trials.

CLING Enables Low-Effort Yet Controllable Automation. We compared questionnaire ratings between conditions using two-sided Wilcoxon signed-rank tests, as shown in Fig. 7. There is a stark tradeoff of the cost and benefit between END-TO-END AGENT and MANUAL. By offloading laborious work to END-TO-END AGENT, participants perceived lower mental demand (END-TO-END AGENT rated more favorably over MANUAL by 1.67 points (95% CI [1.00, 2.25], $p = 0.002$) and low effort (END-TO-END AGENT rated more favorably over MANUAL by 2.42 points, 95% CI [1.58, 3.17], $p = 0.002$), at the cost of a reduced sense of control (END-TO-END AGENT rated lower than MANUAL by 2.92 points, 95% CI [2.25, 3.50], $p < 0.001$).

By allowing participants to automate with small steps and incrementally as needs arose, CLING was rated 1.33 points higher than MANUAL for low mental demand (95% CI [0.58, 2.08], $p = 0.012$).

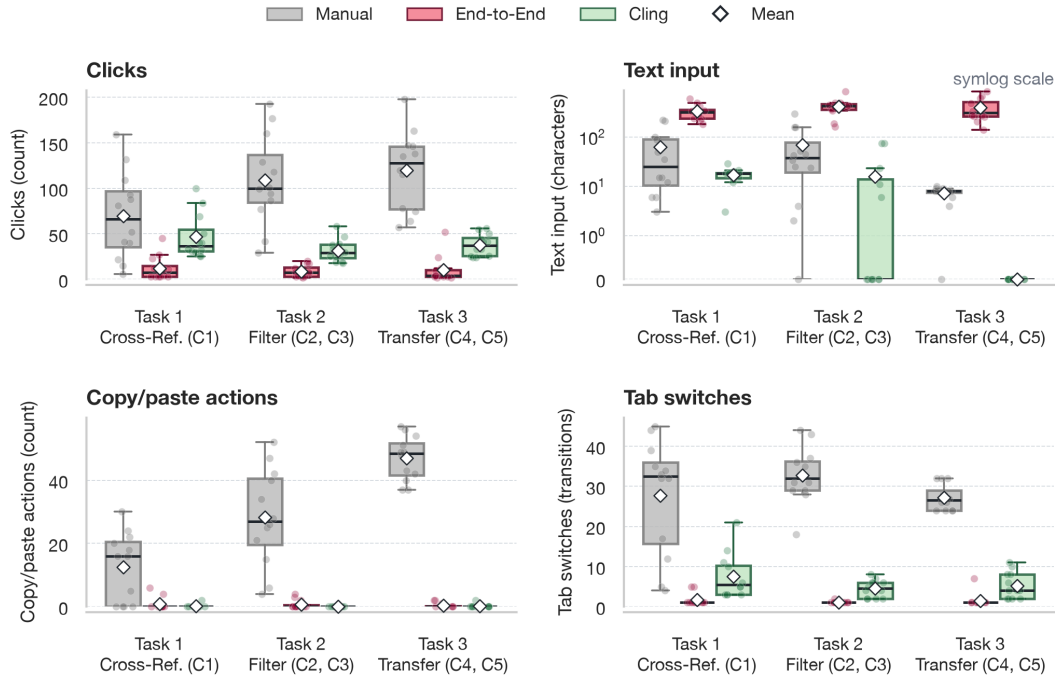


Figure 5: Interaction Counts Across the Three Part 1 Tasks. Compared with MANUAL, CLING reduced clicks, typed characters, copy/paste actions, and tab switches. END-TO-END AGENT required fewer clicks and tab switches than CLING, but substantially more typed characters for prompting.

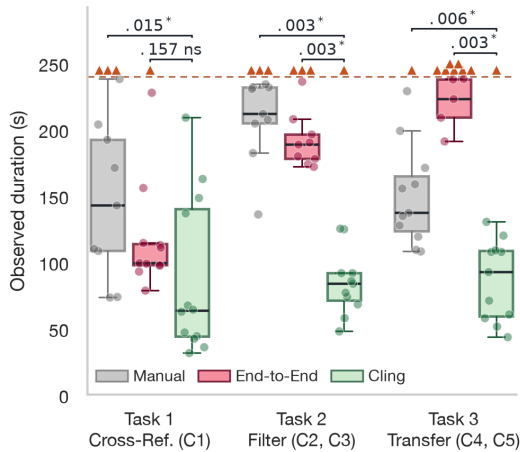


Figure 6: Task Durations Across the Three Part 1 Tasks. With timed-out trials assigned 240 s, CLING reduced overall geometric mean duration by 51.7% relative to MANUAL and 53.2% relative to END-TO-END AGENT.

and 2.08 points higher for low effort (95% CI [1.33, 2.83], $p = 0.002$). At the same time, CLING was rated 2.83 points higher than END-TO-END AGENT for control (95% CI [2.33, 3.33], $p < 0.001$). At the same time, CLING was rated 2.83 points higher than END-TO-END AGENT

for control (95% CI [2.33, 3.33], $p < 0.001$). P9 described CLING as a tool they could control, whereas END-TO-END AGENT felt like “[it’s] working on [its] own.” They compared CLING to a self-retracting tape measure—the button automates retraction, but the user still does the measuring and interpretation.

In-situ, Small-Step Automation Supports Inspection As Automation Unfolds. Among completed attempts, precision and recall were high across all three conditions. MANUAL yielded 259 TP, 599 TN, 13 FP, and 3 FN (participant-level mean precision: 94.6%; recall: 98.9%). END-TO-END AGENT yielded 167 TP, 394 TN, 4 FP, and 5 FN (98.3%; 96.9%). CLING yielded 260 TP, 576 TN, 4 FP, and no FN (97.9%; 100%). Friedman tests found no significant effect of condition on precision, $\chi^2(2) = 2.70$, $p = 0.260$, $W = 0.112$, or recall, $\chi^2(2) = 2.00$, $p = 0.368$, $W = 0.083$. We therefore did not conduct the planned pairwise comparisons.

To capture errors that final-artifact analyses would miss, we also examined errors and corrections during task execution. These execution errors included typos, insertion of entries that did not satisfy the inclusion criteria, and entry of incorrect information. In MANUAL, 6 of 12 participants made at least one execution error, with 12 errors recorded in total. Four participants corrected 9 of these errors (75%). No such errors were observed in END-TO-END AGENT. In CLING, 2 of 12 participants made one execution error each, and both errors were corrected.

Participants attributes their ability to catch and correct mistakes to being able to apply small step automation using CLING in-situ,

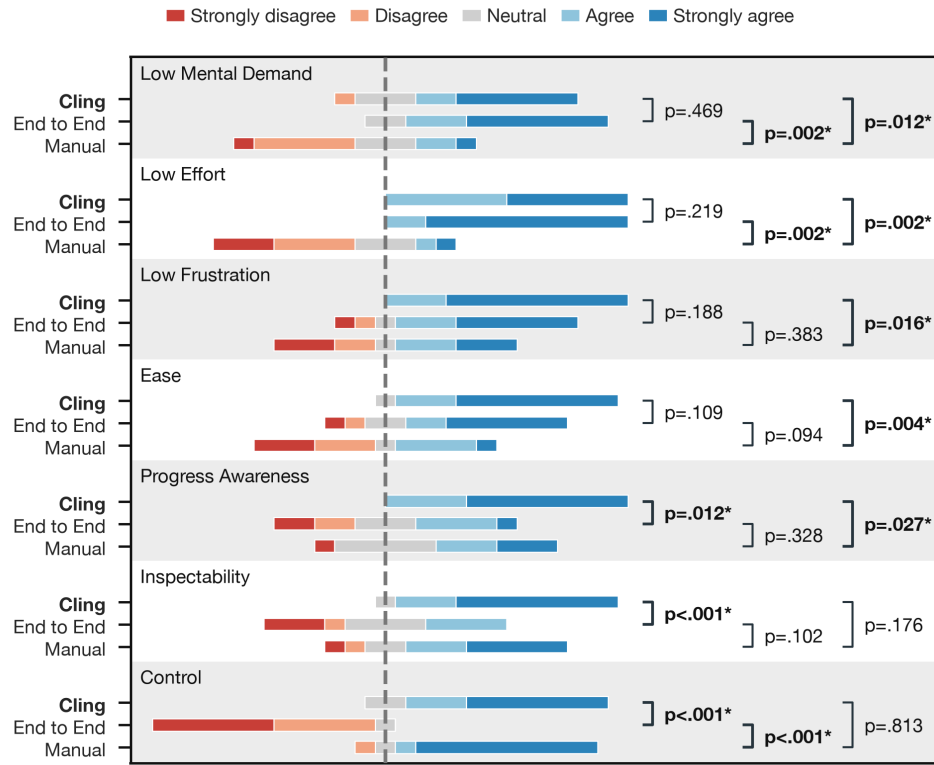


Figure 7: Part 1 Questionnaire Ratings. Both CLING and END-TO-END AGENT received more favorable ratings than MANUAL for mental demand and effort, while CLING offered a greater sense of control than END-TO-END AGENT. Bars show response distributions; brackets report two-sided Wilcoxon signed-rank test p -values, with asterisks indicating $p < 0.05$.

similar to manual experience. Because of CLING’s tightly-coupled intent specification and automation, P5 felt they were handling “input and output at the same time” and therefore “[built] trust along the way.” P8 valued seeing “where the things are coming from,” while P2 emphasized that CLING navigated directly to the relevant content on the page and made inspection easy.

Inspection of END-TO-END AGENT, by contrast, involves assessing whether the agent’s procedure matched participants’ expectations. P10 described “using the thinking trace to compare how I would be doing it” and wanting to inspect further when the procedure diverged from their mental model. Such mismatches could raise concerns without necessarily indicating an error. In the paper-transfer task, for example, P3 worried that an eligible paper had been missed when the agent did not immediately enter it into the spreadsheet, although the agent was retaining results for a later batch update. Verifying END-TO-END AGENT’s outputs could also require independently repeating parts of the task. P8 questioned the benefit of automation when comprehensive verification would require another manual pass: “if I am doing that already, then why to use AI?”

Moreover, the different inspectability influenced participants’ comfort level in sharing the resulting artifacts with collaborators as is. Most participants (P1, P2, P3, P5, P7, P9, P10, P11) placed CLING as a close second to MANUAL. P1 and P2 placed CLING above

MANUAL, citing concerns about the reliability of manual Command+F searches (P1) and easier inspection through automatic navigation to source content (P2).

CLING Accommodates Intent As It Was Discovered and Evolved Through On-Page Information. Throughout the Part 2 tasks, people used information on the webpages as evidence for evaluating options and as a resource through which participants formed, refined, and redirected their intent, even when the information was peripheral to the initial task. P3, for example, described “discovering [their] taste” while browsing furniture, including preferences for colors and materials they had not known they had. P7 developed criteria from what was visible on the page: browsing bed-frame images prompted them to consider adding a headboard, while a lamp page’s built-in filters reminded them that they did not want a portable lamp because it would require another desk to place it on. Page content also shifted existing priorities: after inspecting roughly ten options, P7 reconsidered an initial emphasis on price and decided that “money is less of a priority” than they had expected. P12 mentioned that seeing different couches could “jog my memory” of comfortable furniture at friends’ homes, providing visual cues that helped them clarify what they wanted.

In-situ information could also prompt participants to reconsider constraints or redirect their plans. P12 valued seeing options beyond

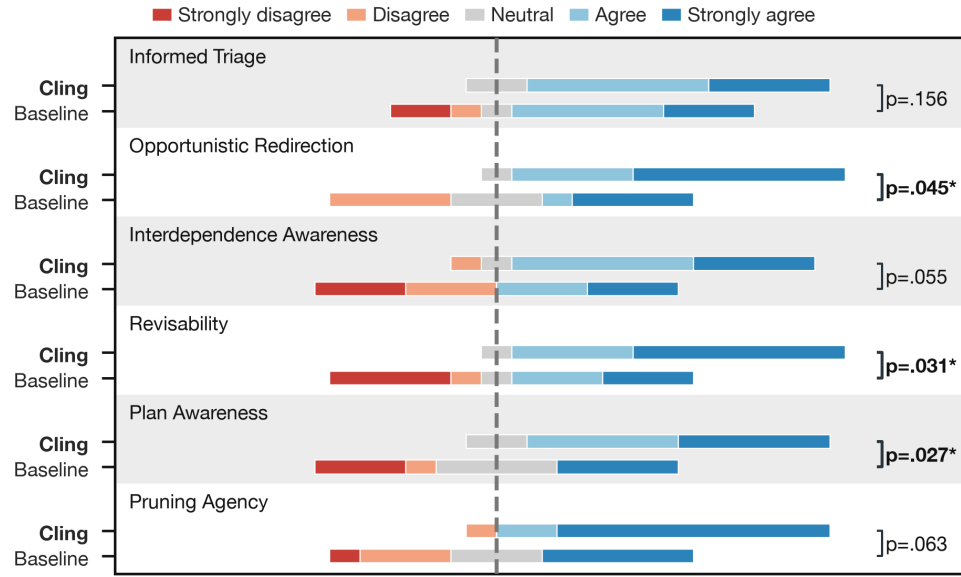


Figure 8: Part 2 Questionnaire Ratings. CLING received more favorable ratings than the baseline for opportunistic redirection, revisability, and plan awareness. Bars show response distributions; brackets report two-sided Wilcoxon signed-rank test p -values, with asterisks indicating $p < 0.05$.

their current criteria because an appealing \$150 item encountered during a \$100 search, for example, might justify revising the budget rather than simply being filtered away. P11’s plan changed more substantially: they initially intended to find a desk, but encountering a gaming-room category led them to pursue a gaming desk instead. Reflecting on such changes, P11 described “being able to improvise” as important to the exploratory process.

Results from Part 2’s questionnaire responses corroborates the theme above (Fig. 8). Compared with the baseline, CLING was rated 1.17 points higher for plan awareness (95% CI [0.33, 2.00], $p = 0.027$), 1.50 points higher for revisability (95% CI [0.50, 2.50], $p = 0.031$), and 1.08 points higher for opportunistic redirection (95% CI [0.33, 1.83], $p = 0.045$).

To complement the findings above, we analyzed how participants updated their criteria during the open-ended furnishing tasks. We counted criteria articulated during think-aloud or externalized in the artifact, including filtering constraints reused in CLING and from prompts sent to the AI assistant in the baseline. The combined count of criterion additions and updates was higher with CLING than with the baseline ($M = 6.58$ vs. 3.83 ; Holm-adjusted $p = 0.41$). Criteria updates specifically were also more frequent with CLING ($M = 1.50$ vs. 0.08 ; two-sided exact paired sign-flip test, Holm-adjusted $p = 0.008$). Although updates were modest in absolute number, eleven of twelve participants updated at least one criterion with CLING, compared with only one participant in the baseline.

9 Discussion

9.1 Situated Automation vs. End-to-End Delegation

CLING makes automation available as coordination needs arise during ongoing exploration. This is particularly beneficial when tasks involve tacit, dynamic, or compounded constraints that emerge through exploration. For example, when deciding which furniture to buy, P7 reported that only after extended comparison did a small set of options begin to “make sense.” What makes an option desirable may depend on many interdependent considerations and be burdensome to articulate. The situated automation supported by CLING does not need to know why an option is desirable; it only needs enough information to carry out the coordination that follows from the user’s choice. This contrasts with task-level delegation in end-to-end automation, which pressures users to translate contextual and still-evolving intent into language. P1 described the “inertia to put things into words,” while P10 preferred directly changing an explicit filter rather than explaining through chat that a previous constraint no longer applied.

Participants also reported that CLING allowed them to remain actively involved for enjoyment, learning, or a sense of responsibility, without being burdened by manual work they considered “noise” (P5). For example, P7 explained that when exploring an unfamiliar space, they might work more manually or “offload a lot” depending on “how much I want to learn.” P2 chose to rely on their own judgment when selecting furniture simply because doing so was “fun.” P4 connected the cross-page transfer task to a recurring work activity in which they would shortlist papers from a large pool for a mentee. They felt CLING would be a good fit because

it could offload the manual foraging of papers while leaving the substantive judgment of suitability to her.

More broadly, this suggests an automate-as-you-go model in which exploration, automation, and assessment remain tightly coupled. Rather than requiring users to specify a complete task and then inspect its outcome, automation can be invoked incrementally as new needs emerge. Instead of merely remaining present to approve or correct automated actions, the user would exert his or her own judgements and redirect the task as their intent develops.

9.2 Schema Correspondence as a Basis for Controllable Automation

In this work, we formalize cross-page coordination as a pervasive interaction pattern in everyday information work and introduce a schema-correspondence perspective that provides a concrete vocabulary for a previously fragmented set of low-level tasks, including copying and pasting, information retrieval, content reformatting, and transfer. Rather than treating these as separate interaction techniques, schema correspondence characterizes them in terms of how entities and attributes on one page relate to those on another.

This abstraction also provides a basis for more controllable automation. LLMs can recover schemas and semantic correspondences from heterogeneous webpages, while explicit rules deterministically specify how information should be selected, aligned, transformed, and transferred. This separation confines probabilistic reasoning to interpreting heterogeneous interfaces while keeping execution predictable and inspectable. Because the resulting rules operate over semantic structures rather than page-specific DOM elements, they can also be inspected, modified, reused, and composed as coordination needs evolve.

9.3 Toward a Semantically Structured Web

In the current Web ecosystem and the design of CLING, the schema of each page must often be reconstructed on the client side at use time, causing different users to repeatedly infer similar semantic structures from the same webpages. A more efficient architecture would allow webpages to expose their entities, attributes, and relationships directly as machine-readable schemas, bringing the Web closer to the Semantic Web vision [6].

The historical difficulty of this vision was partly practical: authoring and maintaining semantic representations imposed additional work on webpage developers and required agreement on representation conventions. LLMs substantially lower this barrier by making it possible to generate and maintain such representations from existing content and interfaces during authoring time. Importantly, this need not require every webpage to share a universal schema. Individual pages could expose their own local semantic structures, while systems such as CLING dynamically infer correspondences between them as coordination needs arise. This suggests a path toward semantic interoperability that combines authoring-time semantic structure with use-time semantic alignment.

10 Conclusion

To summarize, we conceptualize cross-page coordination as a distinct class of interaction and characterize it through five recurring coordination patterns. We then introduce a technical pipeline that

reconstructs reusable webpage schemas and identifies coordination opportunities through schema correspondence, operationalized in our prototype, CLING. Our technical evaluations show that this approach can surface accurate and useful coordination opportunities, while findings from the user study show that in-situ, small-step automation enables users to perform coordination tasks efficiently while remaining in control. More broadly, this work points toward an information environment in which independently developed webpages become interoperable at the point of use, allowing users and AI to collaboratively compose information and functionality across page boundaries to accomplish complex information tasks more effectively.

References

- [1] 2026. Selenium IDE. <https://www.selenium.dev/selenium-ide/>. Accessed: 2026-03-29.
- [2] Adobe Inc. [n. d.]. Adobe Creative Cloud. <https://www.adobe.com/creativecloud.html>. Adobe Creative Cloud product page.
- [3] Apple Inc. 1995. *OpenDoc*. <https://en.wikipedia.org/wiki/OpenDoc>. Software framework for compound documents.
- [4] Shaon Barman, Sarah Chasins, Rastislav Bodik, and Sumit Gulwani. 2016. Ringer: web automation by demonstration. In *Proceedings of the 2016 ACM SIGPLAN international conference on object-oriented programming, systems, languages, and applications*. 748–764.
- [5] Matthew T Beaudouin-Lafon, Devamardeep Hayatpur, Arvind Satyanarayan, and Haijun Xia. 2026. Belidor: A Specification Language for Operationalizing Structural Analogies Between User Interfaces. In *Proceedings of the 2026 CHI Conference on Human Factors in Computing Systems*. 1–21.
- [6] Tim Berners-Lee, James Hendler, and Ora Lassila. 2001. The Semantic Web. *Scientific American* 284, 5 (2001), 34–43.
- [7] Nadia Boukhelifa, Jonathan C Roberts, and Peter J Rodgers. 2003. A coordination model for exploratory multiview visualization. In *Proceedings International Conference on Coordinated and Multiple Views in Exploratory Visualization-CMV 2003*. IEEE, 76–85.
- [8] Browser Use Authors. 2025. Browser-use Documentation: Introduction. <https://docs.browser-use.com/introduction>. Accessed: 2026-02-19.
- [9] Daniel Buschek. 2024. Collage is the New Writing: Exploring the Fragmentation of Text and User Interfaces in AI Tools. In *Proceedings of the 2024 ACM Designing Interactive Systems Conference (Copenhagen, Denmark) (DIS '24)*. Association for Computing Machinery, New York, NY, USA, 2719–2737. doi:10.1145/3643834.3660681
- [10] Yining Cao, Jane L E, Zhutian Chen, and Haijun Xia. 2023. Dataparticles: Block-based and language-oriented authoring of animated unit visualizations. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*. 1–15.
- [11] Yining Cao, James D Hollan, and Haijun Xia. 2026. Exploring Fairy Cursor as a Form of AI Agent for In-the-Flow Assistance: Design Opportunities and Challenges. In *Proceedings of the 2026 Designing Interactive Systems Conference*. 1153–1169.
- [12] Yining Cao, Yiyi Huang, Anh Truong, Hujung Valentina Shin, and Haijun Xia. 2025. Compositional Structures as Substrates for Human-AI Co-creation Environment: A Design Approach and A Case Study. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*. 1–25.
- [13] Yining Cao, Peiling Jiang, and Haijun Xia. 2025. Generative and Malleable User Interfaces with Generative and Evolving Task-Driven Data Model. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems (CHI '25)*. Association for Computing Machinery, New York, NY, USA, Article 686, 20 pages. doi:10.1145/3706598.3713285
- [14] Joseph Chee Chang, Nathan Hahn, and Aniket Kittur. 2020. Mesh: Scaffolding comparison tables for online decision making. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology*. 391–405.
- [15] Joseph Chee Chang, Nathan Hahn, Adam Perer, and Aniket Kittur. 2019. SearchLens: composing and capturing complex user interests for exploratory search. In *Proceedings of the 24th International Conference on Intelligent User Interfaces*. 498–509.
- [16] Sarah E Chasins, Maria Mueller, and Rastislav Bodik. 2018. Rousillon: Scraping distributed hierarchical web data. In *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology*. 963–975.
- [17] Weihao Chen, Xiaoyu Liu, Jiacheng Zhang, Ian Long Lam, Zhicheng Huang, Rui Dong, Xinyu Wang, and Tianyi Zhang. 2023. Miwa: Mixed-initiative web automation for better user control and confidence. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*. 1–15.

- [18] Zhutian Chen and Haijun Xia. 2022. CrossData: Leveraging Text-Data Connections for Authoring Data Documents. In *CHI '22: CHI Conference on Human Factors in Computing Systems, New Orleans, LA, USA, 29 April 2022 - 5 May 2022* (New Orleans, LA, USA) (CHI '22). ACM, New York, NY, USA, Article 95, 15 pages. doi:10.1145/3491102.3517485
- [19] Jacob Cohen. 1960. A coefficient of agreement for nominal scales. *Educational and psychological measurement* 20, 1 (1960), 37–46.
- [20] Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, et al. 2025. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261* (2025).
- [21] Anita Crescenzi, Yuan Li, Yinglong Zhang, and Rob Capra. 2019. Towards Better Support for Exploratory Search through an Investigation of Notes-to-self and Notes-to-share. In *Proceedings of the 42nd international ACM SIGIR conference on research and development in information retrieval*. 1093–1096.
- [22] Crummy. 2026. Beautiful Soup. <https://www.crummy.com/software/BeautifulSoup/>. Accessed: 2026-03-29.
- [23] Cypress.io. 2026. Cypress. <https://www.cypress.io/>. Accessed: 2026-03-29.
- [24] Databricks. [n.d.]. Databricks: Leading Data and AI Platform for Enterprises. <https://www.databricks.com/> Databricks homepage.
- [25] Rui Dong, Zhicheng Huang, Ian Long Lam, Yan Chen, and Xinyu Wang. 2022. WebRobot: web robotic process automation using interactive programming-by-demonstration. In *Proceedings of the 43rd ACM SIGPLAN International Conference on Programming Language Design and Implementation*. 152–167.
- [26] Mira Dontcheva, Steven M Drucker, David Salesin, and Michael F Cohen. 2007. Relations, cards, and search templates: user-guided web data integration and layout. In *Proceedings of the 20th annual ACM symposium on User interface software and technology*. 61–70.
- [27] Simret Araya Gebreegziabher, Yukun Yang, Charles Chiang, Hojun Yoo, Chaoran Chen, Hyo Jin Do, Zahra Ashktorab, Werner Geyer, Diego Gómez-Zarà, and Toby Jia-Jun Li. 2026. The Behavioral Fabric of LLM-Powered GUI Agents: Human Values and Interaction Outcomes. *arXiv preprint arXiv:2601.16356* (2026).
- [28] Google. 2026. Puppeteer. <https://pptr.dev>. Accessed: 2026-03-29.
- [29] Google. n.d.. Google Workspace: Secure Online Productivity & Collaboration Tools. <https://workspace.google.com/> Google Workspace product page.
- [30] Google. n.d.. Grounding with Google Search. <https://ai.google.dev/gemini-api/docs/google-search> Accessed: 2026-09-11.
- [31] Kilem Li Gwet. 2008. Computing inter-rater reliability and its variance in the presence of high agreement. *Brit. J. Math. Statist. Psych.* 61, 1 (2008), 29–48.
- [32] Björn Hartmann, Leslie Wu, Kevin Collins, and Scott R Klemmer. 2007. Programming by a sample: rapidly creating web applications with d. mix. In *Proceedings of the 20th annual ACM symposium on User interface software and technology*. 241–250.
- [33] James Hollan, Edwin Hutchins, and David Kirsh. 2000. Distributed cognition: toward a new foundation for human-computer interaction research. *ACM Transactions on Computer-Human Interaction (TOCHI)* 7, 2 (2000), 174–196.
- [34] Eric Horvitz. 1999. Principles of mixed-initiative user interfaces. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*. 159–166.
- [35] Eric J Horvitz, John S Breese, David Heckerman, David Hovel, and Koos Rommelse. 2013. The Lumiere project: Bayesian user modeling for inferring the goals and needs of software users. *arXiv preprint arXiv:1301.7385* (2013).
- [36] Zhicheng Huang and Jeff Huang. 2025. IRCHIVER: An Information-Centric Personal Web Archive for Revisiting Past Online Sensemaking Tasks. In *Proceedings of the 2025 ACM SIGIR Conference on Human Information Interaction and Retrieval*. 231–246.
- [37] Peiling Jiang and Haijun Xia. 2025. Orca: Browsing at scale through user-driven and ai-facilitated orchestration across malleable webpages. *arXiv preprint arXiv:2505.22831* (2025).
- [38] David R. Karger. 2007. Haystack: Per-User Information Environments Based on Semistructured Data. In *Beyond the Desktop Metaphor: Designing Integrated Digital Work Environments*, Victor Kaptelinin and Mary Czerwinski (Eds.). MIT Press, Cambridge, MA.
- [39] Clemens N Klokmoose, James R Eagan, Siemen Baader, Wendy Mackay, and Michel Beaudouin-Lafon. 2015. Webstrates: shareable dynamic media. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*. 280–290.
- [40] Clemens Nylandsted Klokmoose, James R Eagan, and Peter Van Hardenberg. 2024. MyWebstrates: Webstrates as local-first software. In *Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology*. 1–12.
- [41] Clemens N Klokmoose, Christian Remy, Janus Bager Kristensen, Rolf Bagge, Michel Beaudouin-Lafon, and Wendy Mackay. 2019. Videostrates: Collaborative, distributed and programmable video manipulation. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology*. 233–247.
- [42] Bill Kules and Robert Capra. 2008. Creating exploratory tasks for a faceted search interface. In *Second Workshop on Human-Computer Interaction (HCIR 2008)*.
- [43] Andrew Kuznetsov, Joseph Chee Chang, Nathan Hahn, Napol Rachatasumrit, Bradley Breneisen, Julina Coupland, and Aniket Kittur. 2022. Fuse: In-Situ sense-making support in the browser. In *Proceedings of the 35th Annual ACM Symposium on User Interface Software and Technology*. 1–15.
- [44] Michelle S Lam, Omar Shaikh, Hallie Xu, Alice Guo, Diyi Yang, Jeffrey Heer, James A Landay, and Michael S Bernstein. 2026. Just-In-Time Objectives: A General Approach for Specialized AI Interactions. In *Proceedings of the 2026 CHI Conference on Human Factors in Computing Systems*. 1–26.
- [45] Jürgen Landauer and Masahito Hirakawa. 1995. Visual AWK: a model for text processing by demonstration. In *Proceedings of Symposium on Visual Languages*. IEEE, 267–274.
- [46] Henry Lieberman et al. 1995. Letizia: An agent that assists web browsing. *IJCAI (1)* 1995 (1995), 924–929.
- [47] Geoffrey Litt and Daniel Jackson. 2020. Wildcard: spreadsheet-driven customization of web applications. In *Companion Proceedings of the 4th International Conference on Art, Science, and Engineering of Programming*. 126–135.
- [48] Michael Xieyang Liu, Jane Hsieh, Nathan Hahn, Angelina Zhou, Emily Deng, Shaun Burley, Cynthia Taylor, Aniket Kittur, and Brad A Myers. 2019. Unakite: Scaffolding developers' decision-making using the web. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology*. 67–80.
- [49] Michael Xieyang Liu, Aniket Kittur, and Brad A Myers. 2021. To reuse or not to reuse? A framework and system for evaluating summarized knowledge. *Proceedings of the ACM on Human-Computer Interaction* 5, CSCW1 (2021), 1–35.
- [50] Michael Xieyang Liu, Aniket Kittur, and Brad A Myers. 2022. Crystalline: Lowering the cost for developers to collect and organize information for decision making. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*. 1–16.
- [51] Michael Xieyang Liu, Andrew Kuznetsov, Yongsung Kim, Joseph Chee Chang, Aniket Kittur, and Brad A Myers. 2022. Wigglyte: Low-cost information collection and triage. In *Proceedings of the 35th Annual ACM Symposium on User Interface Software and Technology*. 1–16.
- [52] Yimeng Liu, Misha Sra, Jeevana Priya Inala, and Chenglong Wang. 2026. Reuseit: Synthesizing reusable ai agent workflows for web automation. In *Proceedings of the 31st International Conference on Intelligent User Interfaces*. 885–908.
- [53] Gary Marchionini. 2006. Exploratory search: from finding to understanding. *Commun. ACM* 49, 4 (2006), 41–46.
- [54] M McIlroy, EN Pinson, and BA Tague. 1978. UNIX time-sharing system. *The Bell system technical journal* 57, 6 (1978), 1899–1904.
- [55] Yevgeniy Medynskiy, Mira Dontcheva, and Steven M. Drucker. 2009. Exploring websites through contextual facets. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Boston, MA, USA) (CHI '09). Association for Computing Machinery, New York, NY, USA, 2013–2022. doi:10.1145/1518701.1519007
- [56] Microsoft. [n.d.]. Visual Studio Code. <https://code.visualstudio.com/> Visual Studio Code homepage.
- [57] Microsoft. 2024. [MS-OLEDS]: Object Linking and Embedding (OLE) Data Structures. https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-ole/85583d21-c1cf-4afe-a35f-d6701c5fbb6f Protocol specification; published version dated 2024-04-23, revision 13.0.
- [58] Microsoft. n.d.. Microsoft 365 Products, Apps, and Services. <https://www.microsoft.com/en-us/microsoft-365/products-apps-services> Microsoft 365 web page.
- [59] Microsoft. n.d.. The Unified Data Platform. <https://www.microsoft.com/en-us/microsoft-fabric> Microsoft Fabric product page.
- [60] Robert C Miller and Alisa M Marshall. 2004. Cluster-based find and replace. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. 57–64.
- [61] Robert C Miller and Brad A Myers. 2001. Interactive Simultaneous Editing of Multiple Text Regions.. In *USENIX ATC, General Track*. 161–174.
- [62] Robert C Miller and Brad A Myers. 2002. Multiple selections in smart text editing. In *Proceedings of the 7th international conference on Intelligent user interfaces*. 103–110.
- [63] Bryan Min, Allen Chen, Yining Cao, and Haijun Xia. 2025. Malleable Overview-Detail Interfaces. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems (CHI '25)*. Association for Computing Machinery, New York, NY, USA, Article 688, 25 pages. doi:10.1145/3706598.3714164
- [64] Meredith Ringel Morris, Jarrod Lombardo, and Daniel Wigdor. 2010. WeSearch: supporting collaborative search and sensemaking on a tabletop display. In *Proceedings of the 2010 ACM conference on Computer supported cooperative work*. 401–410.
- [65] Theodor Holm Nelson. 1995. The heart of connection: hypermedia unified by transclusion. *Commun. ACM* 38, 8 (Aug. 1995), 31–33. doi:10.1145/208344.208353
- [66] Notion Labs, Inc. n.d.. The AI workspace that works for you. <https://www.notion.com/> Notion homepage.
- [67] OpenAI. 2025. Operator. <https://operator.chatgpt.com/>. Accessed: 2026-02-19.
- [68] OpenClaw Authors. 2025. OpenClaw. <https://openclaw.ai/>. Accessed: 2026-02-19.
- [69] Srishti Palani, Zijian Ding, Austin Nguyen, Andrew Chuang, Stephen MacNeil, and Steven P Dow. 2021. CoNotate: Suggesting queries based on notes promotes

- knowledge discovery. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. 1–14.
- [70] Sharoda A Paul and Meredith Ringel Morris. 2009. CoSense: enhancing sense-making for collaborative web search. In *Proceedings of the SIGCHI conference on human factors in computing systems*. 1771–1780.
- [71] Project Jupyter. [n.d.]. Project Jupyter | Home. <https://jupyter.org/> Project Jupyter homepage.
- [72] Kevin Pu, Rainey Fu, Rui Dong, Xinyu Wang, Yan Chen, and Tovi Grossman. 2022. Semanticon: Specifying content-based semantic conditions for web automation programs. In *Proceedings of the 35th Annual ACM Symposium on User Interface Software and Technology*. 1–16.
- [73] Kevin Pu, Jim Yang, Angel Yuan, Minyi Ma, Rui Dong, Xinyu Wang, Yan Chen, and Tovi Grossman. 2023. Dilogics: Creating web automation programs with diverse logics. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*. 1–15.
- [74] Roman Rädle, Midas Nouwens, Kristian Antonsen, James R Eagan, and Clemens N Klokmoose. 2017. Codestrates: Literate computing with webstrates. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology*. 715–725.
- [75] Jonathan C Roberts. 2007. State of the art: Coordinated & multiple views in exploratory visualization. In *Fifth international conference on coordinated and multiple views in exploratory visualization (CMV 2007)*. IEEE, 61–71.
- [76] Omar Shaikh, Shardul Sapkota, Shan Rizvi, Eric Horvitz, Joon Sung Park, Diyi Yang, and Michael S Bernstein. 2025. Creating general user models from computer use. In *Proceedings of the 38th Annual ACM Symposium on User Interface Software and Technology*. 1–23.
- [77] Nikhil Sharma, Q Vera Liao, and Ziang Xiao. 2024. Generative echo chamber? effect of llm-powered search systems on diverse information seeking. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*. 1–17.
- [78] Melissa E Sleeter. 1996. OpenDoc—building online help for a component-oriented architecture. In *Proceedings of the 14th annual international conference on Systems documentation: Marshaling new technological forces: building a corporate, academic, and user-oriented triangle*. 87–94.
- [79] Ayah Soufan, Ian Ruthven, and Leif Azzopardi. 2022. Searching the literature: An analysis of an exploratory search task. In *Proceedings of the 2022 conference on human information interaction and retrieval*. 146–157.
- [80] Lucille Alice Suchman. 1987. *Plans and situated actions: The problem of human-machine communication*. Cambridge university press.
- [81] Tableau. [n.d.]. Business Intelligence and Analytics Software. <https://www.tableau.com/> Tableau homepage.
- [82] Anh Truong, Floraine Berthouzoz, Wilmot Li, and Maneesh Agrawala. 2016. Quickcut: An interactive tool for editing narrated video. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*. 497–507.
- [83] Stephen Volda, Elizabeth D. Mynatt, and W. Keith Edwards. 2008. Re-framing the desktop interface around the activities of knowledge work. In *Proceedings of the 21st Annual ACM Symposium on User Interface Software and Technology (Monterey, CA, USA) (UIST '08)*. Association for Computing Machinery, New York, NY, USA, 211–220. doi:10.1145/1449715.1449751
- [84] Austin R Ward and Robert Capra. 2021. OrgBox: Supporting cognitive and metacognitive activities during exploratory search. In *Proceedings of the 44th international ACM SIGIR conference on research and development in information retrieval*. 2570–2574.
- [85] Ryen W White, Gary Marchionini, and Gheorghe Muresan. 2008. Evaluating exploratory search systems. *Information Processing and Management* 44, 2 (2008), 433.
- [86] Ryen W White and Resa A Roth. 2009. *Exploratory search: Beyond the query-response paradigm*. Number 3. Morgan & Claypool Publishers.
- [87] Barbara M Wildemuth and Luanne Freund. 2012. Assigning search tasks designed to elicit exploratory search behaviors. In *Proceedings of the symposium on human-computer interaction and information retrieval*. 1–10.
- [88] Haijun Xia, Jennifer Jacobs, and Maneesh Agrawala. 2020. Crosscast: Adding Visuals to Audio Travel Podcasts. In *UIST '20: The 33rd Annual ACM Symposium on User Interface Software and Technology, Virtual Event, USA, October 20–23, 2020*. ACM, 735–746. doi:10.1145/3379337.3415882
- [89] Jiacheng Zhang, Jiawen Li, Maryam Arab, and Steve Oney. 2025. Multi-Click: Cross-Tab Web Automation via Action Generalization. In *Proceedings of the 38th Annual ACM Symposium on User Interface Software and Technology*. 1–10.