Maru: Information Architecture as a Shared Language for Generating Aligned and Persistent User Interfaces

Eunhye Kim
School of Computing, KAIST
Daejeon, Republic of Korea
gracekim027@kaist.ac.kr

DaEun Choi
School of Computing, KAIST
Daejeon, Republic of Korea
daeun.choi@kaist.ac.kr

Bryan Min
University of California San Diego
La Jolla, California, USA
bdmin@ucsd.edu

Hyunjung Yi
Korea University
Seoul, Republic of Korea
ruby3672@korea.ac.kr

Yue Jiang
University of Utah
Salt Lake City, Utah, USA
yue.jiang@utah.edu

Juho Kim
School of Computing, KAIST
Daejeon, Republic of Korea
SkillBench
Santa Barbara, California, USA
juhokim@kaist.ac.kr

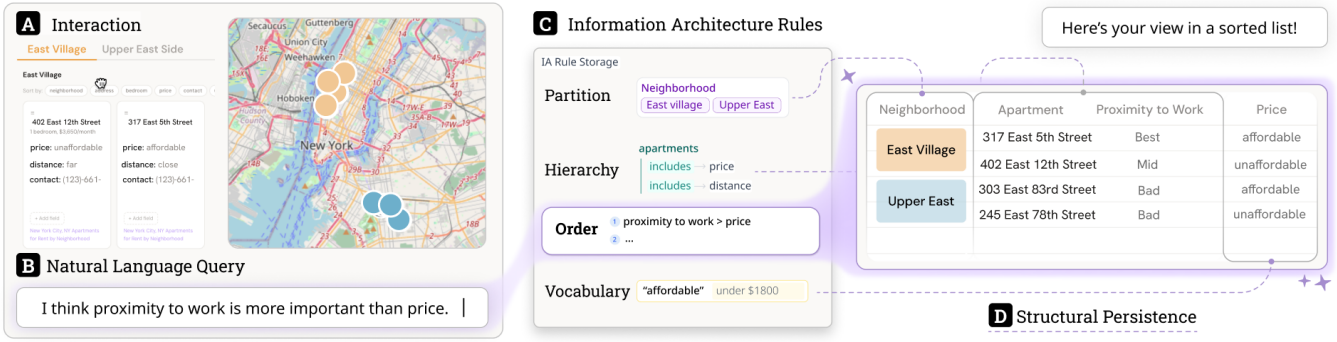

Figure 1: Maru carries forward what users establish as IA rules across generations. A user browses apartments in a map+tabs view and states *"I think proximity to work is more important than price,"* (B) adding a new Order rule to their accumulated IA state. The same rule could be created by sorting the list directly in the generated UI (A). Maru surfaces the full IA state (C), with Partition, Hierarchy, Order, and Vocabulary rules captured across prior interactions and the new Order rule highlighted. The next generation produces a sorted table reflecting structural persistence (D). *What changed:* the layout type and item ordering. *What persisted:* the neighborhood partition, hierarchy fields, and affordability vocabulary, carried forward without re-specification.

Abstract

Generative user interfaces (GenUIs) promise on-demand components tailored to users' needs. As users iterate on information tasks, they construct personal structures over information they encounter—how items are grouped, what gets prioritized, and what terms mean in their context. Yet, current systems leave these structural decisions to the model at each generation, ignoring the structural logic users have established. Without a persistent representational structure shared between user and system, GenUIs have no basis to remain aligned with what users have established. We draw on Information Architecture (IA), a design practice for organizing and structuring information, as a shared language to bridge user-constructed structure and system generation. We present a framework identifying four IA elements—partition, hierarchy, order, and vocabulary—and characterize how each maps to concrete UI generation decisions. We instantiate this framework in Maru, a conversational system that captures user prompts and interactions as IA preferences, persisting as rules both user and system draw on across generations. A user study revealed that IA persistence kept generated UIs aligned as sessions progressed, while alignment without it degraded, with diverse patterns emerging across users and contexts, pointing to the value of IA persistence in aligning GenUI to individual needs.

CCS Concepts

• **Human-centered computing** → **User interface programming**; **HCI theory, concepts and models**.

Keywords

Generative User Interfaces, Information Architecture

This work is licensed under a Creative Commons Attribution 4.0 International License.

UIST '26, Detroit, MI, USA

© 2026 Copyright held by the owner/author(s).

ACM ISBN 979-8-4007-2856-3/2026/11
https://doi.org/10.1145/3830398.3830621

ACM Reference Format:
Eunhye Kim, DaEun Choi, Bryan Min, Hyunjung Yi, Yue Jiang, and Juho Kim. 2026. Maru: Information Architecture as a Shared Language for Generating Aligned and Persistent User Interfaces. In *The 39th Annual ACM Symposium on User Interface Software and Technology (UIST '26), November 02–05, 2026, Detroit, MI, USA.* ACM, New York, NY, USA, 20 pages. https://doi.org/10.1145/3830398.3830621

1 Introduction

Advances in large language models (LLMs) are making the vision of generative UI (GenUI) a reality—systems that can produce structured, personalized user interfaces on demand. Given a natural language query, these systems extract relevant information and render it into widgets, comparison tables, or interactive layouts tailored to the user's context [5, 12, 28, 48]. This capability is shaping GenUI as an exciting opportunity towards achieving personal software across commercial tools, websites, and applications [40, 50].

To make GenUI usable for real-world websites and commercial software, communities in industry, AI, and HCI have been exploring how to improve the quality of generated UIs [36, 101], breadth of types of UIs that LLMs can generate [5, 18, 48], and customizability of the generated UIs [12, 27]. Yet these advances say little about which UI elements should change or stay between prompts, and how much importance each element carries for the user throughout their task.

Although GenUI's ability to modify and regenerate nearly any aspect of an interface makes it highly customizable, this same flexibility introduces risks in modifying aspects of the UI that the user has come to rely on. For example, suppose a user receives a generated table listing available apartments in New York and finds the column-and-row structure personally valuable for comparing options. The system has no way of recognizing this value, and cannot guarantee that the next generated UI will retain it. As a result, users must re-articulate their UI preferences with each generation. Thus, to build GenUI systems that stay aligned with users over time, we must design systems that capture and preserve UI elements that the user values across generations.

In this paper, we aim to identify and provide this persistence in GenUI by drawing from the practice of **Information Architecture** (IA). IA is a UX design practice that operationalizes the mapping of user needs to concrete UI content and layouts [10, 30, 81]. An IA design process for an apartment-finding interface might involve understanding the user's key criteria, prioritizing them to surface the most valued ones, grouping related content into navigable sections, and structuring the information into a list and map for users to explore. We propose that IA offers a promising abstraction for this problem: specific enough to map directly onto concrete UI decisions, yet general enough to capture the structural logic users build across a task, and crucially, *explicit* enough to be *inspectable* and *editable* by users. Our goal is to investigate whether integrating IA into the generation pipeline can enable the system to carry forward what users have established, producing UIs that remain aligned across generations.

We conducted a systematic codebook analysis of 42 papers spanning sensemaking theory and interactive systems, from which we derived four IA elements: **partition**, **hierarchy**, **order**, and **vocabulary**. We then characterized how each maps into concrete generation decisions, and then instantiated this framework in Maru, a probe that captures IA rules from natural language prompts and direct interactions with generated outputs, and applies them as a persistent layer across generations. Finally, we conducted a user study (N=12), observing when persistence helped or hindered users, how users utilized the IA rules, and how they helped them personalize the UI.

Our user study found that Maru maintained approval rates across sessions (74% to 61%) while the baseline collapsed from 71% to 33%, that 97% of user-created rules were built through normal interaction and query behavior rather than deliberate schema editing, and that Maru produced distinct layout types where baseline participants converged on similar ones. However, IA persistence degraded alignment during sub-task transitions and when rules over-accumulated beyond relevance. Overall, our results suggest that integrating IA rules into the UI generation pipeline helps users shape the interface to their needs, and that persistence needs boundaries to stay useful. We open-source Maru, including the IA layer, in hopes that it can support others in building GenUI systems that carry user-constructed structure forward across generations.[1]

In summary, we contribute:

- A framework of four IA elements, mapping user behavior in sensemaking to UI generation decisions.
- Maru, a probe instantiating this framework as an explicit, user-controllable IA layer for generative UI.
- Empirical findings on how IA persistence shapes UI alignment, user engagement, and personalization.

2 Related Work

2.1 Achieving Personal Software with Generative UI

Traditional GenUI approaches primarily involved breaking down key UI components into modular elements assembled automatically. Model-Based User Interface Design (MBUID) developed high-level models mapping to UI specifications, describing task stages [38, 73, 93] or the data, functionality, and view [9], enabling the system to translate high-level user needs into UI components [24, 66–69]. These models were authored before deployment and, by design, remained hidden from end-users. Other approaches generated UIs from interaction feedback rather than models [95]. Although these approaches could generate rich, interactive UIs, a common barrier was domain specificity: these systems were limited to dialog UIs [24], command-line GUIs [95], or predefined templates [55, 66, 67].

LLMs removed this barrier by enabling UI generation across domains—with chatbots generating UI components and widgets [5, 94] and SVGs [6], and full front-end prototypes [1, 28, 97]. This capability is also advancing software customization and end-user programming through UI mashups [49, 106], shareable and reusable UI components [26, 29, 44], and personalized UI styles [41].

However, a key challenge for LLM-powered GenUI systems is keeping generated UIs aligned to the user's needs across prompt iterations as the user's task progresses. A general approach the

[1] https://github.com/kixlab/Maru

HCI community has pursued involves generating intermediate outputs before the final UI to enrich the LLM's context. Recent work proposes generating task-driven data models [12, 102], while other work infers user needs "in-the-moment" to produce more contextually relevant UIs at each generation stage [47, 89].

However, these approaches still delegate to the LLM the decision of which UI elements to generate from context. Our work instead makes this mapping explicit. We provide a structured encoding that determines which elements to persist and which to change, giving end-users direct visibility and control over how their needs map to UI elements. In doing so, our work joins a broader tradition of declarative specifications for interfaces [8, 86], which differ from one another in the semantics that they encode—in our case, how a user organizes information.

2.2 Designing for Persistence in AI Systems

Persistence has long been recognized as essential for continuity in information-intensive tasks. Users actively construct internal structures—schemas, criteria, priorities, and relational maps—to make sense of what they encounter [43, 75, 76], with significant effort directed toward finding the right representational schema rather than consuming content [42, 74, 84]. Early work addressed this through structured representations: sensemaking systems demonstrated that externalizing these structures reduces reconstruction cost [14, 15, 54, 92], and that exploratory search advances as users build increasingly rich organizational structures [57, 78]. However, externalizing structure remains effortful even with lightweight tools [13, 31, 46, 53, 65, 82], and the structures captured remain session-specific, and cannot transfer beyond the tool's own immediate affordances [104].

LLM-era work has approached persistence more flexibly, inferring preferences [60, 89], conversational memories [103], and intent specifications [96, 109] from interaction data. Across these approaches, the underlying motivation is consistent: without carry-forward, users must re-establish context, and alignment degrades over time. Systems have shown that making accumulated context visible and editable gives users greater confidence and control [96, 103], and that surfacing what the system has inferred or would infer differently helps users refine their intent [25, 105].

What remains underexplored is persistence at the structural level of generated interfaces, where LLM-era approaches carry context fluidly but were not designed to preserve the structural preferences that shape what a useful interface looks like for a given user. Our work addresses this gap directly for GenUI.

2.3 Information Architecture for UI Persistence

Information architecture (IA) is a UX design practice concerned with organizing, labeling, and structuring information for effective navigation and sensemaking [10, 30, 81]. In practice, IA work begins with understanding user needs and making structural decisions: how content is grouped, what hierarchies govern navigation, how items are sequenced, and what terminology is used [81]. These decisions are externalized through sitemaps, taxonomies, and card sorting exercises that resolve diverse organizational schemes into a single canonical structure [11, 39]. The result is a set of structural rules established before any UI is rendered, which acts as a blueprint shaping how users explore and understand information [11, 85, 87]. When interface architecture conflicts with a user's own logic it creates information anxiety; when it aligns, it acts as a cognitive scaffold [70, 88].

In traditional practice, these decisions are author-owned and static—settled at design time and applied uniformly to users [10, 30, 81]. Recent work challenges this: research on malleable interfaces show that end-users produce diverse and personally meaningful arrangements when given control over content and layout [3, 61, 63], and adjacent systems have proposed structured layers that sit alongside generation [12]. Yet in current GenUI systems, the mapping from user context to UI decisions is delegated entirely to the LLM, which neither exposes nor persists the structural logic it applies. Thus, the user has limited ability to inspect, correct, or build on it.

We propose that IA offers a promising abstraction for this problem. Unlike specifications written for a system, IA describes how information is organized for a person to make sense of it, which is a concern users already have about their own information. By making structural mapping explicit, persistent, and user-editable, IA can serve as an enabling layer through which the system captures structural rules derived from user behavior, and the user can inspect and modify those same rules to further shape the interface [34, 70, 90].

3 A Framework of IA Elements for GenUI

The goal of this framework is to guide the use of IA elements in mapping user needs and activities to concrete UI patterns. We built this framework through three steps: (1) collecting literature and tools, (2) identifying the user needs and resulting UI patterns supported by each system, and (3) synthesizing common elements that recur across systems as mappings between user behavior and UI structure.

In this paper, we ground our framework in sensemaking literature. Sensemaking underlies a wide range of everyday information tasks such as conversing with LLMs or searching, and the research around it has theorized both sides of this mapping—how users actively build structure organizing information and how interfaces support that [43, 75, 84]. This makes sensemaking literature a principled and broadly applicable source for deriving IA elements that can serve as a shared language between user behavior and system generation. Applying the same approach to other domains such as writing [19, 47], data analysis, or learning [18] would likely yield a different set of elements. Generative systems for other domains surface structures organized around how behavior decomposes or how variations branch, rather than how information is organized [4, 7]. We therefore scope our framework to sensemaking, where user-constructed structure is central, and view the four elements as a starting vocabulary for identifying what persists in other domains.

3.1 Derivation Method

We conducted a thematic analysis of existing systems, theories, and empirical studies. We included papers that describe systems or interaction techniques supporting the organization or navigation of information during information tasks, or provide theoretical or empirical accounts of user behavior during such tasks.

Our analysis proceeded in three phases. Two coders independently analyzed an initial set of 12 seed papers spanning seminal sensemaking theory and systems, developing an initial codebook on the user needs, UI elements, and interactions supported through discussion. They then coded 8 additional papers identified through snowball sampling, refining and stabilizing the codebook until no new codes emerged. A single coder then applied the finalized codebook to the remaining 22 papers, reaching a total of 42 papers. For each IA element, a full mapping of the coded systems, their supported interactions, and corresponding UI patterns is provided in Appendix A.2, and the final set of papers is provided in Appendix A.1.

3.2 Defining and Characterizing the Elements

Our analysis yielded four IA elements—partition, hierarchy, order, and vocabulary—along with a set of system support mechanisms and visual/interaction patterns. The four IA elements are divided into two categories: structural elements (partition and hierarchy), which describe how users organize information, and semantic elements (order and vocabulary), which capture the meaning and criteria users bring to that structure.

To illustrate our framework, we use an example of an apartment search task, in which a user browses multiple listings, reads reviews, and compares options across several sessions.

3.2.1 Partition. Partition is an organizational structure in which items are grouped together because they are similar or related, without any item assuming a special role relative to another. Items within a partition are *lateral peers*—they share a boundary but not a hierarchy. In our apartment search example, a user might group listings into neighborhoods (*Downtown, Midtown, Uptown*) or divide their criteria into *must-haves* and *nice-to-haves*.

For example, in ForSense [77], users dragged information clips into color-coded spatial clusters on a canvas, regrouping them as their mental model of a topic evolved. Each cluster is a peer of the others with no group subordinating another. In SearchLens [15], users tagged keywords into Lenses rendered as a button group, keeping each facet distinct and independently adjustable. Together these systems show that partition groups create boundaries that are explicit without implying role relationships between them.

Beyond these, systems supported partition construction through drag-and-drop into groups [45, 77], tag or label assignment [15], and spatial arrangement on a canvas [92]. These were rendered through spatial clusters [77, 92], tab groups and card clusters [16, 45], color coding as a boundary signal [17, 77], and faceted result sections [15, 21].

3.2.2 Hierarchy. Hierarchy is an organizational structure in which items bear differentiated roles relative to one another, encoding a *semantic role relationship*—position signals how an item stands relative to its neighbors. A user might place evidence clips about safety, commute time, and price under Apartment A, treating each clip as *evidence* that characterizes the *option*. Hierarchy applies to information items, partitions, and other hierarchies.

Tabs.do [16] illustrates hierarchy as decomposition: users nested task bundles under parent bundles through drag-and-drop, building persistent tree structures. The role relationship here is one of containment, where each level qualifies and contextualizes the one below it. Unakite [51] shows a different form of hierarchy, where users organized programming options against evaluation criteria in a comparison table, with evidence clips nested under their respective criteria and options. Here the role relationship is one of evidence: position signals what supports what.

Beyond these, systems supported hierarchy construction through drag-and-drop into containers [45, 51], context menus for nesting [16], and tag creation [103]. These were rendered through tables [14, 51], accordions and toggles [16, 37], nested cards [45, 100], and node-link diagrams [71, 92, 103].

3.2.3 Order. Order is the assignment of *sequence* or *weight* to items, partitions, or hierarchies based on the user's subjective sense of importance, priority, or relevance. Unlike hierarchy and partition, order describes a *property* of elements within a structure. In our apartment seeking example, a user might weight *safety* as three times more important than *commute time*, or rank Apartment A as their current top choice.

Mesh [14] illustrates order as discrete relative weighting: users assigned importance multipliers to criteria and dragged rows to reflect their intuitive ranking, with options reordered by aggregate score so the most preferred rose to the top. The user's priorities are directly encoded as structural properties of the output. CueFlik [21] shows a different form of order: users manipulated the relative importance of each active rule through per-rule weight sliders, with the final ranking of results continuously shifting to reflect the combined weighting.

Beyond these, systems encoded order through drag-and-drop interactions [14, 51], priority selectors and pin or star actions [16, 31], and weight sliders [21]. These were rendered through spatial placement reflecting importance [14, 16], weight badges and emphasis on high-priority items [15, 83], and collapsing or filtering of low-priority items [14, 53].

3.2.4 Vocabulary. Vocabulary is the set of personal terms a user establishes with task-specific or session-specific meaning, functioning as a *controlled personal lexicon*: once defined, terms serve as persistent semantic anchors for organizing, labeling, and evaluating information consistently across the task. A user might define *affordable* as any listing under $1,800/month—an evaluation criterion that applies consistently across all listings.

CheatSheet [98] illustrates vocabulary well: users created and assigned task-specific tags through direct text input, building a personal navigational vocabulary that persisted across their note library. These terms were rendered as tag chips and surfaced as tooltips on hover, keeping the user's own terminology visible at the point of use.

Beyond this, systems supported vocabulary construction through inline text input [51, 52], tag creation [53, 80], and term extraction from annotated content [91, 92]. These were rendered through persistent label components using the user's own terminology [14, 51], tag chips and filter labels [53, 54], and tooltips and inline highlights [17, 110].

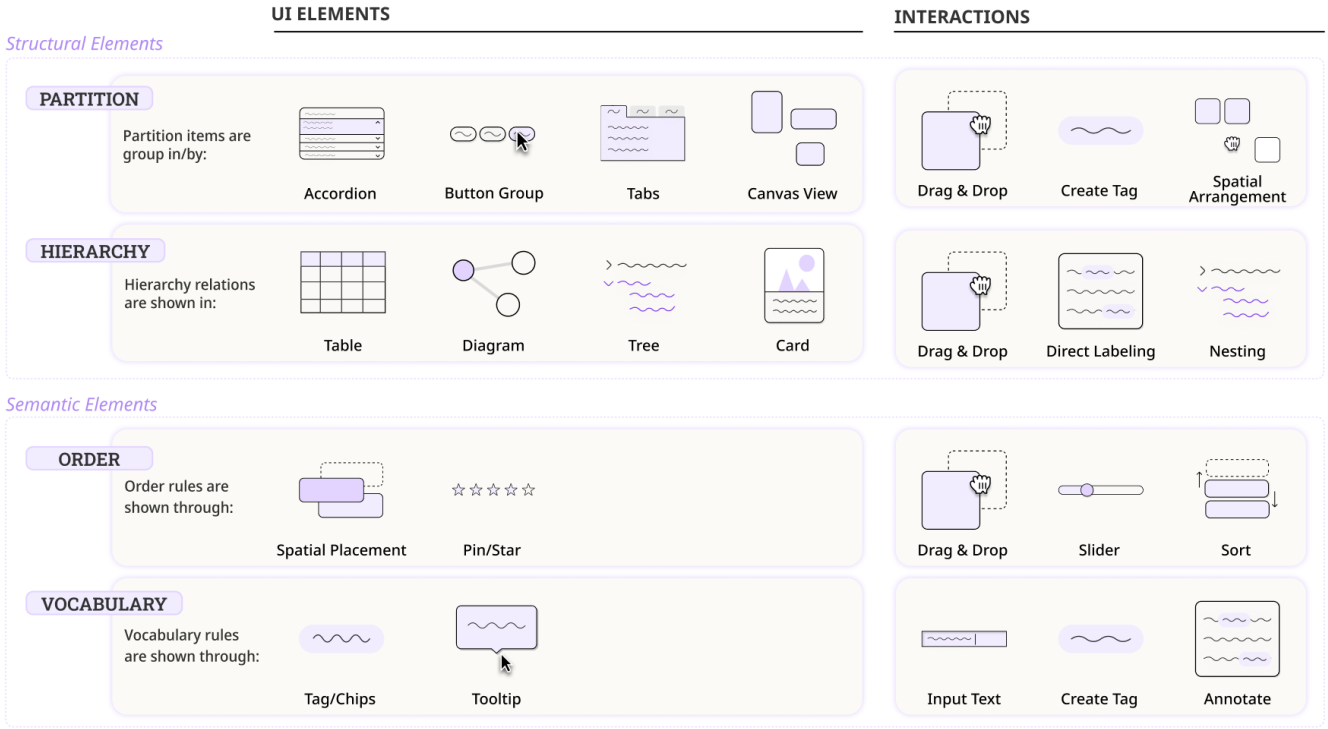

Figure 2: Our framework of IA elements, with representative UI components and construction interactions for each element. Each element can be both displayed and constructed through the UI itself. A partition rule, for instance, can be shown as a set of tabs grouping related items, and edited by dragging an item from one tab to another, which moves it between groups.

3.3 Persistence of IA Elements

Because IA elements capture the structural and semantic preferences users build over the course of a task, they are persistent by nature. Yet our analysis found that most systems treat user-constructed structure as ephemeral, discarding it when a query ends or a session closes [14, 54, 92]. Several systems documented this kind of persistence as a desired behavior that went unsupported [15, 20, 21], while systems that did support it showed measurably lower reconstruction costs [15, 16]. We therefore define ***persistence*** as the property of IA elements that carry forward across generations within a session and should be retrieved and reapplied to new generations, with the user being able to explicitly add or drop them.

4 The Maru Probe

We instantiate the framework in Maru, a chat-based GenUI system. The IA layer is implemented as a set of skill specifications independent of Maru's UI components, so other GenUI systems can consume the rule schema directly without adopting our rendering pipeline. Both the IA layer and the Maru implementation are available at https://github.com/kixlab/Maru.

4.1 Envisioned User Scenario

We introduce Maru with an envisioned user scenario.

Max opens Maru to search for apartments in New York. She types *"Find me apartments in New York under $2,500"* and receives a card-grid. The system scaffolds two initial IA rules–a Partition grouping apartments by neighborhood and a Hierarchy nesting each apartment under it (Figure 3-A). Noticing inconsistent use of "affordable", she types *"Affordable means under $1,800 and within 20 minutes of downtown,"* adding a Vocabulary rule applied to all subsequent generations (Figure 3-B).

As her search deepens, Max asks *"Which neighborhoods should I focus on?"* The system retrieves her Partition rules and selects a Kanban layout–each neighborhood becoming a column. She drags a Brooklyn apartment into a new column she labels *"Strong Candidates."* The drag updates her Partition rule, adding the new group without any explicit rule editing (Figure 3-C). She then opens the IA panel directly and types an Order rule: *"proximity to work > pet-friendly > parking availability,"* (Figure 3-D). After a few iterations, her view has settled into a map alongside a table of listings. Wanting to understand how her neighborhoods and criteria relate to one another, she regenerates it as a diagram without changing any rules—the same partitions and hierarchy now render as a node-link graph (Figure 3-E).

Finally, Max asks: *"Plan my apartment visits for this Saturday."* The system generates a Timeline ordered by her priority rule, grouped by neighborhood partition, and annotated with her affordability vocabulary—a view that without accumulated IA context would produce a generic list with no awareness of her priorities, hierarchies, groupings, or definitions (Figure 3-F).

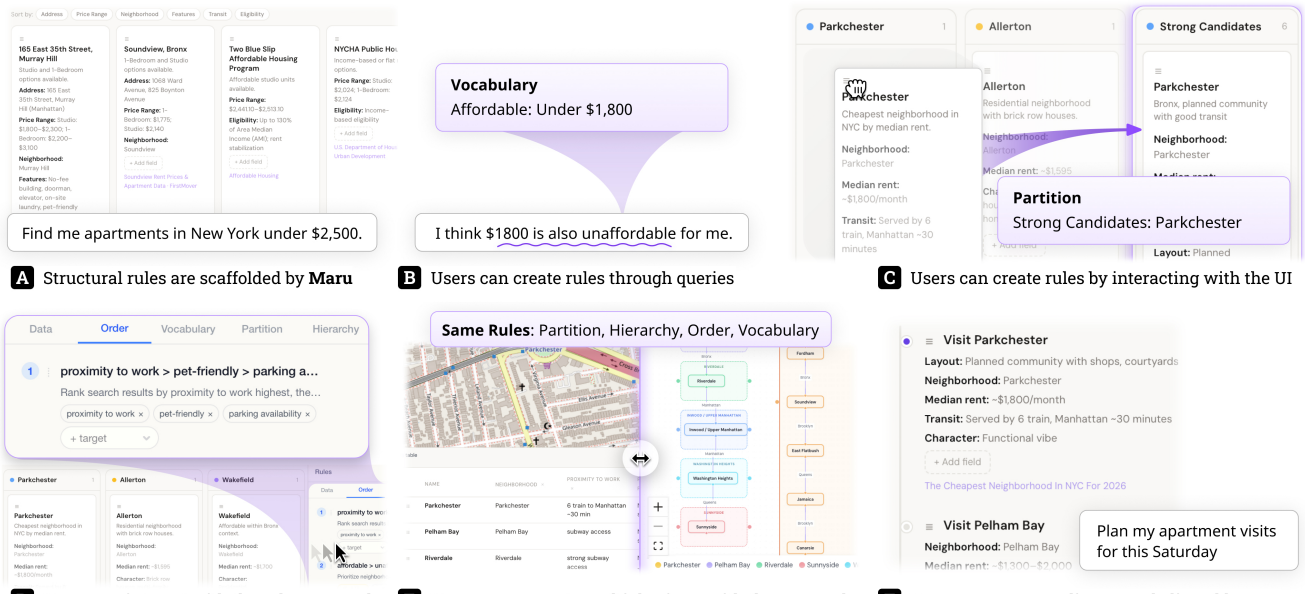

Figure 3: An envisioned scenario of a user searching for apartments in New York with Maru. (A) Structural rules are scaffolded automatically from the initial query. (B) The user refines rules through natural language, adding a Vocabulary rule for "affordable." (C) The user drags an apartment into a new column within the generated UI itself, which updates a Partition rule without any explicit rule editing. (D) Separately, the user adds an Order rule by editing the IA panel directly. (E) The same accumulated rules produce multiple layout types without re-specification. (F) Rules accumulated across all three channels drive a task-specific Timeline layout for visit planning.

4.2 IA Construction

Maru is mainly composed of (1) an IA construction module and (2) an IA-based generation module. For IA construction, the user can manipulate the rules via chat, UI interaction or direct editing in the IA panel. When a session starts, Maru detects the hierarchy and partition rules scaffolded from the generated UI as the starting point for the user's rules to build on.

4.2.1 Query-based Rule Detection. The system detects IA rules from the user query. The user can submit a natural-language request to the system, such as *"I want to see the apartments by neighborhood, help me compare them"*. The rules are inferred by an LLM, which uses the summarized chat context alongside the full active rule set, since rules can be applied on top of other rules as described in the framework.

4.2.2 Interaction-based Detection. Users can interact with the generated UI to communicate their structural preferences. Interactions are of two types: layout-agnostic and layout-specific, each mapping to specific IA elements according to the framework. An example of a layout-agnostic interaction is dragging a child component from one parent to another (e.g., a card from one tab to another), which triggers a partition rule update. A layout-specific interaction is connecting nodes with labeled links in a node-link diagram, which triggers a hierarchy rule. All supported layout types and their corresponding interactions are listed in Appendix A.2.2.

We enable interaction-based detection by rendering each layout component with built-in gesture handlers. Each component type exposes a defined set of interactions (e.g., drag-to-reorder, inline edit, lasso-group) that are captured as typed IA events and mapped to rule updates. All supported interactions update the rendered layout in place and write their IA rule directly, without invoking the generation pipeline. A new generation occurs only when the user issues a chat-level action—a new query, a reply, or an explicit regeneration request. Extending Maru to a new layout type involves defining the gestures that the component exposes and mapping each to one of the four IA elements. New layouts therefore require new gesture handlers but not new rule types, and the mapping in Table A.4 can grow without changing the framework.

4.2.3 The IA Panel. The rules and data they are bound to are visualized and made accessible to the user through a panel. Each rule has a distinct visual representation, and the user can directly inspect, modify, and delete rules through the IA panel.

4.3 IA-based Generation

4.3.1 Rule Retrieval. Given a user prompt, the summarized chat context and the full active rule set are provided to the LLM to retrieve rules relevant to the current generation. For instance, if the user has partitions for "Neighborhood 1", "Neighborhood 2", and queries *"Let's plan out the apartments in Neighborhood 1."*. The partition rule and its bound data entities for "Neighborhood 1" are retrieved.

4.3.2 Layout Selection. UI generation proceeds in two steps: layout selection and layout filling. Layout selection takes as input the

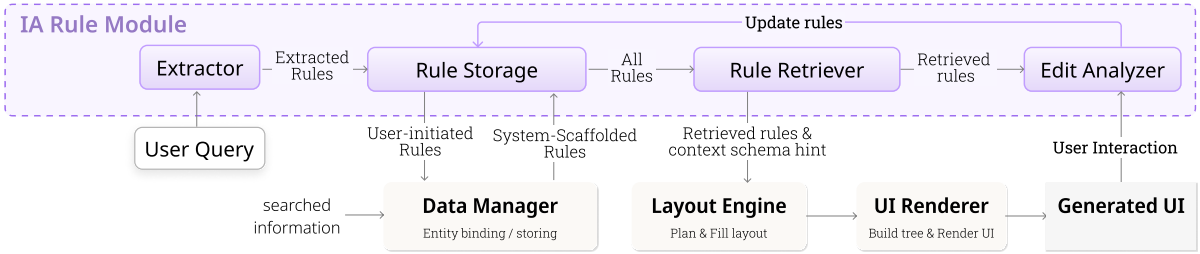

Figure 4: The Maru pipeline. User queries are processed by the Extractor to detect IA rules, which are stored and retrieved alongside system-scaffolded rules to guide the Layout Engine. The UI Renderer builds and renders the generated interface, while the Edit Analyzer captures user interactions and feeds rule updates back into Rule Storage.

user query, few-shot examples of similar user needs and system views drawn from our literature codebook, and the retrieved hierarchy and partition rules. This step outputs a layout skeleton as a JSON structure specifying component types and group labels for the filling step. The user query is incorporated because some layouts carry task-specific priorities (e.g., a map view) that cannot be inferred from the rules alone. Since hierarchy and partition are the structural rules that determine grouping and nesting, these are used to guide layout selection. All supported layout types are listed in Appendix A.2.2 and an overview of the Maru interface can be found in Appendix A.2.1.

4.3.3 Layout Filling. Once the layout type is selected, the layout skeleton, the user query, and all four IA element rules with their bound data are sent to the LLM for content filling. Order and vocabulary rules participate in this stage because they are semantic rules that affect which items surface, how they are labeled, and how they are ranked within the structure. Both steps output JSON, which is parsed into typed, interactive block components for rendering.

4.4 Implementation Details

Maru is built with Electron, React, and TypeScript as a desktop client, with a main process handling all LLM calls and IA logic and a renderer process running the React UI. Rule detection, rule retrieval, layout planning, edit analysis, and response attribution use `gpt-4o-2024-11-20`.

UI Rendering. Each assistant response is rendered through a two-stage pipeline. First, a layout planner selects a component type (e.g. card-grid, kanban, tabs, timeline) based on the query, retrieved data, and active IA rules. Second, a filler populates the chosen layout with content, producing a typed plan tree. The renderer walks this tree recursively, instantiating each node as an interactive React component with gesture handlers (drag-to-reorder, inline edit, delete, sort, lasso-group). Because the plan tree is a declarative data structure, the same response can be re-rendered or locally modified without re-invoking the LLM.

Entity Storing and Binding. Rules reference items via stable UUID pointers or loose concept labels. When data items are extracted from source content, concept references are automatically upgraded to UUID-based links, ensuring partition memberships and hierarchy relationships remain intact across item renames and field edits.

5 User Study

To evaluate whether IA-based persistence produces more aligned and personalized generative UIs, we conducted a comparative study with Maru and a baseline system without the IA pipeline. We address three research questions:

[RQ1] What are the benefits and costs of IA-based persistence for UI alignment? Maru accumulates IA rules across queries and interactions, applying them persistently on subsequent generations, while the baseline system does not. This distinction lets us isolate the effect of IA-based persistence: does carrying forward user-constructed structure improve alignment over the course of a session, or does accumulated rule state introduce its own costs?

[RQ2] How do users communicate IA preferences, and what does their engagement reveal? Maru provides three channels through which users can communicate structural preferences. Understanding how rules are constructed in practice and how engagement varies across users reveals whether the shared IA language is accessible across different kinds of users.

[RQ3] How does IA-based persistence bound the space of generated UIs? IA rules translate user-constructed structure into concrete layout decisions. For IA rules to produce structurally distinct, user-personalized outputs, different rule states must yield meaningfully different UIs.

5.1 Baseline System

We designed the baseline as an ablation of the IA layer. The baseline probe shared the same pipeline as Maru, supporting the same UI interactions, layout types, and queries. However, the IA module was removed, with no rule detection, rule storage, codebook-derived schema hints, or rule-based layout examples. Both conditions received the same base context at each generation, including the current user query, the retrieved data items, a summary of the immediately preceding generation, and the user's full prior interaction history. User interactions were instead logged semantically (e.g., `Moved item "Brooklyn Apt" to Tab "To Consider"`) and included in the prompt as prior interaction context. The two conditions therefore differed only in how that history was represented. In Maru, the same interaction would be encoded as a typed Partition rule; in the baseline, it remains an unstructured log entry. The prompt differences are explained in detail in Appendix A.10.

5.2 Participants, Procedure, and Measurements

We recruited participants through an online community at our institution. A total of 12 participants (average age = 24.3, 5 male, 7 female) were recruited. All participants reported daily or near-daily LLM use and prior experience with information-intensive tasks.

Each participant completed three task sessions (S1-3). In the first two sessions (S1 and S2), participants completed two assigned tasks: graduate school program or career exploration and picnic planning. Each task was completed with one system–Maru or the baseline–with system order counterbalanced across participants. In the third session (S3), participants used Maru on a real information task they had prepared in advance. Full session procedures and detailed task descriptions are provided in Appendix A.9. Each participant was compensated approximately $20 USD, and the study was conducted under institutional IRB approval.

The system logged timestamped records of all participant actions (queries, UI interactions, and rule edits) and system outputs (LLM prompts, responses, chosen layouts, and generation durations) to per-session log files. Session metadata including participant ID, condition, and duration was tracked in a separate log file for cross-session analysis.

6 User Study Results

Across 36 sessions, participants generated 220 UIs and performed 585 interactions, constructing a total of 1,292 IA rules (838 user-attributable and 454 system scaffolded). Sessions spanned nine of the supported layout types, with participants in the Maru condition producing layouts across an average of 2.7 distinct types per session. Prompt lengths were comparable across conditions (median 3.4k tokens per model call in the baseline vs. 3.3k in Maru), indicating the conditions differed in how prior context was represented rather than how much was provided. We report findings across three RQs: how IA-based persistence shapes UI alignment, how users construct IA rules across interaction channels, and how accumulated rule states produce personalized UI trajectories. We provide a gallery of all generated widgets as supplementary material.

6.1 RQ1. What are the benefits and costs of IA-based persistence for UI alignment?

Maru's IA-based persistence produces more aligned UIs across the session. After each generation, participants rated UI alignment with their needs on a binary scale (accept/reject). Maru's UIs were more likely to be accepted overall (OR = 2.08, 95% CI $[1.25, 3.47]$, $p = .005$), and this advantage was not attributable to context volume—the effect persisted after statistically adjusting for the prompt tokens consumed by each generation (adjusted OR = 2.40, 95% CI $[1.28, 4.47]$, $p = .006$). In the baseline condition, this approval rate started at 71% in the first half of the session but collapsed to 33% in the second half (OR = 0.19, 95% CI $[0.08, 0.45]$, $p < .001$). In contrast, Maru-generated UIs started at 74% and held at 61% throughout the second half (OR = 0.63, 95% CI $[0.15, 2.65]$, $p = .53$). This stability is reflected in generation efficiency as well: participants reached a satisfactory output in fewer generations under Maru ($M = 5.58$, $SD = 1.62$) than the baseline ($M = 7.75$, $SD = 2.09$; Wilcoxon signed-rank $V = 0$, $p = .008$). When asked which condition felt more aligned with their preferences, 9 participants preferred Maru. Participants attributed their preference to the system's ability to carry forward what they had established: P8 noted that they could "see effort in the results—it was persistent with what I had valued before...the system understood what I was trying to convey."

Participants leveraged IA rules to align the UI to their needs. Without IA-based persistence, baseline users developed manual compensatory strategies to maintain structural continuity. This is visible in query behavior: in Maru, the average query length of user prompts declined by 10.6 words (28.7 → 18.1) from the first half to the second half of the session, as accumulated rules absorbed the specificity that users would otherwise have to re-articulate. In the baseline, the decline was only 2.5 words (21.8 → 19.3), with some participants copy-pasting previous queries and appending new conditions to compensate for the lack of carry-over. Others avoided generating new outputs altogether, returning to a previous UI and continuing to interact within it, since the state of a single widget persisted internally even if it would not carry over to the next generation. P2 illustrates the failure that drove this: "When I ask the baseline system to add the activities into the calendar that it had created before, it just creates a new calendar and ignores the calendar that it created." Both strategies—re-articulating their preferences or limiting their interactions to a single output—reflect the same underlying problem: without a persistent structural layer, users had no basis to expect that a new generation would preserve what they had already established. In Maru, this overhead largely disappeared, and short refinements like "show me more options" replace multi-sentence specifications as rules carried structural context forward.

IA rules hindered alignment when persisting across sub-tasks during context shifts. While IA-based persistence improved alignment overall, it degraded in two conditions. The first was sub-task transitions. When participants shifted focus within a session, rules from the prior sub-task were retrieved and applied to the new context. Generations where cross-topic rules were retrieved had a 23% thumbs-down rate compared to 18% for on-topic retrievals. P5 shifted from PhD program search to HCI lab exploration and rated three consecutive UIs as unsatisfactory, each carrying 5–6 rules from the cost-of-living sub-task—"Housing" partitions and "total" ordering that had no relevance to lab selection. P11 similarly observed an "Off-campus" partition from venue search leak into food recommendations during picnic planning. In each case, the system correctly persisted rules that had been valid in their original context but failed to recognize that the user had moved on. Users found this frustrating, with P9 saying that "I didn't like that something from a separate task carried over into the next." This suggests that IA persistence in generative UI systems needs to take in consideration a task-aware scoping [12] that tracks which rules belong to which sub-task context to help scope retrieval appropriately.

IA rules hindered alignment when over-accumulated. Rule accumulation was another condition where IA-based persistence degraded alignment. Alignment suffered when active rule counts rose continuously across sessions: from a mean of 15.2 in the first quarter to 37.5, 59.1, and 101.6 in subsequent quarters. This growth was driven by two mechanisms. First, the system's scaffolding

	Partition	Hierarchy	Order	Vocabulary
UI interaction	305	149	53	132
Query extraction	54	65	19	32
Manual panel edit	5	5	9	10
Total (838)	364	219	81	174

Table 1: Distribution of the 838 user-attributable IA rules by extraction method and IA element.

mechanism generated new structural rules with each generation by scanning the data schema of the generated plan—an average of approximately 24 system-scaffolded hierarchy rules per generation that were added to the rule state without user action or awareness. Second, user interactions accumulated refinement rules that were never pruned, merged, or expired, even when they were no longer relevant. Although retrieval matched queries against the active rule set, Maru lacked a mechanism to identify rules the user had mentally discarded without explicit pruning. The two participants with the highest rule counts—P5 (194 rules, 33% approval) and P11 (198 rules, 50% approval)—had the lowest approval ratings in the study, and the slight second-half decline (74% → 61%) in Maru's approval rate is partially attributable to this accumulation. This suggests that both user-created and system-scaffolded rules need context-sensitive lifespans, with relevance weighting that accounts for recency.

6.2 RQ2. How do users communicate IA preferences, and what does their engagement reveal?

IA rules emerge from a collaborative loop of system scaffolding and user-expressed preferences. Across all sessions, Maru's combination of scaffolding and user refinement produced 1,292 active rules, of which 838 were user-attributable, created or modified through interactions, queries, or manual panel edits, and 454 were system-inferred rules used for layout population. When the system surfaced rules inferred from user behavior, participants largely agreed: of 94 rules that received explicit feedback, 88 were accepted (94%, 95% CI [87%, 97%]). This suggests that the IA framework functions as a genuinely shared language, and the structural inferences the system makes from user behavior are ones users recognize as their own.

Users rarely created new IA rules, but frequently interacted with the ones they were given. Of the 838 user-attributable rules, 76% originated from UI interactions, 20% from natural language extraction of queries, and only 3% from manual panel editing (Table 1)—meaning 97% of user-created rules were built without any deliberate engagement with the IA schema. Only 29 rules across all 12 participants were manually created or modified through the panel. The interaction-to-rule mapping was the main channel through which users communicated their preferences: sorting items generated order rules, removing or adding fields generated hierarchy rules, deleting items generated partition rules, and lasso-grouping generated both partition and vocabulary rules. The reliability of this mapping suggests that interaction behavior is the most natural channel through which users express structural preferences—and therefore future IA-based generative UI systems should be designed to capture a broad range of interaction types as structural signals.

Users produced and modified unintuitive IA rules through direct manipulation, while articulating intuitive ones through prompts. Although users found semantic rules (vocabulary, order) to be the most intuitive dimensions, they overwhelmingly generated, interacted with, and modified structural ones (partition, hierarchy). In post-session interviews, participants ranked vocabulary as the most natural and immediately understandable, followed by order, partition, and hierarchy—which multiple participants described as difficult to grasp (P5-7, P11). However, our behavioral data revealed that partition was the most common user-created rule type at 43% (364/838), followed by hierarchy at 26% (219), vocabulary at 21% (174), and order at 10% (81). UI interactions produced a balanced spread across all four types (21% vocabulary, 8% order, 48% partition, 23% hierarchy), while natural language extraction skewed toward hierarchy (38%) as users mentioned fields in queries. Participants' accounts suggest this ordering followed how much each element matched concepts they already had words for. Sorting and defining a term are operations users bring from everyday software, while hierarchy had no such counterpart. Participants performed hierarchy operations constantly, but adding or removing a field read to them as adjusting what they wanted to see rather than editing a structural relation. Partition fell in between, with P4 and P6 both reporting that they only came to understand how a partition worked after seeing one applied. This finding extends prior knowledge about the value of both prompting and direct manipulation in AI systems [59]. Users engaged with IA structurally through their behavior even when they could not articulate it, suggesting that interaction is the primary channel through which the shared language operates.

Users without design expertise increasingly customized the UI with IA rules throughout their task. Although Maru added persistence to what users expressed, it did not change how control felt: perceived control was nearly identical across conditions (Maru: 3.25/5, Baseline: 3.00/5, Δ=+0.25, p=.555), since the main control channel itself (interaction and query) was the same in both conditions. What differed was engagement depth. Participants without design experience rarely opened the IA panel, interacting through queries and direct manipulation alone. Yet over the course of their sessions, these participants became more intentional in how they communicated with the system. P3 noted the system was "aligning well when I ask narrowing questions," P4 described shifting from exploration to deliberate comparison, and P10 began leveraging persistence intentionally: "Because I asked to include safety, it now always includes it." Design-experienced participants engaged more explicitly from the outset: P5 prompted with IA terminology directly, P12 requested specific layout types, and P11 remarked they would rather build the interface themselves. These patterns suggest users engaged with the IA layer at different depths depending on their design expertise, and that the IA framework—even when operating invisibly—can serve as an entry point through which users with no design background gradually develop structural agency over their generated UIs.

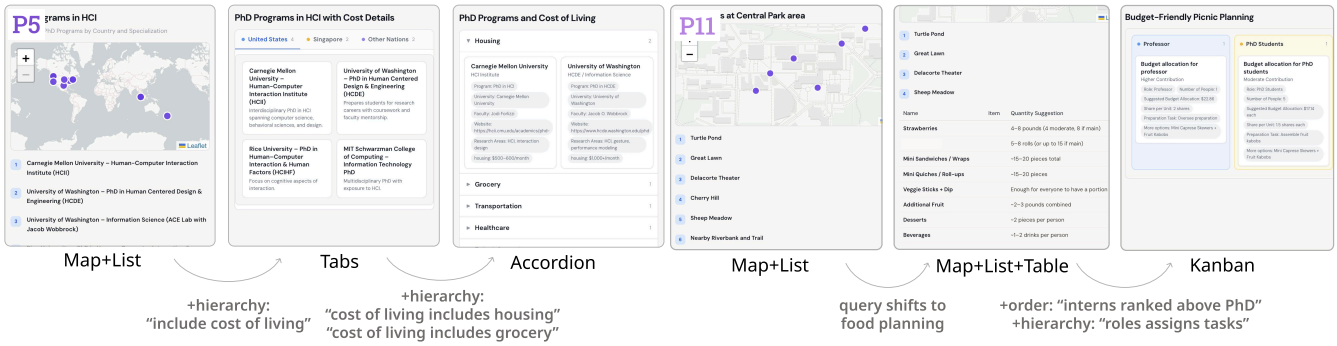

Figure 5: UI layout trajectories for P5 (grad school search) and P11 (picnic planning) illustrating how IA rules shape layout changes differently across users. P5's accumulating hierarchy rules trigger progressive layout shifts (Map+List → Tabs → Accordion), each transition reflecting the growing parent–child structure in their rule state. P11's rules accumulate across queries without producing visible layout changes until a new hierarchy and order rule combination triggers a shift to kanban.

6.3 RQ3. How does IA-based persistence bound the space of generated UIs?

IA rules bound layout divergence across users. Twelve participants each performed two assigned tasks—career/grad school finding and picnic planning—one in each condition. In both tasks and both conditions, first-generation layouts showed approximately 50% agreement, as both systems made comparable initial guesses based on the query alone. Over the course of each session, however, the conditions diverged. In the career task, all six baseline participants' final layouts converged to tables. Without persistent structural preferences to differentiate on, every participant ended at that default regardless of their individual needs. In Maru, the six career participants ended on four distinct layout types—maps, accordion, tabs, and list. The picnic task followed the same directional pattern: Maru produced six distinct final layout types compared to the four in the baseline. The career task converged more strongly because it has a clearer default representation: picnic planning, with its multiple facets (venue, food, schedule, budget), lacks a single layout, so both conditions produced more variety.

IA elements route to distinct layout types, and rule changes precede layout transitions. Cross-tabulating active rule states with generated layouts across all system sessions reveals systematic mappings between IA dimensions and layout types. When partition rules were present, 33% of generated layouts used grouped representations—kanban views, tabbed interfaces, or accordions—compared to 0% without. When five or more hierarchy rules had accumulated, 10% of layouts were tables, compared to 0% with fewer. Order rules were associated with a 17% rate of table or list layouts versus 5% without. Notably, these mappings reflect layout type rather than complexity, since average section count remained between 1.2 and 1.6 regardless of how many rules were active. These rule changes preceded and plausibly drove the layout transitions that followed: of 45 layout transitions observed across Maru sessions, 29 (64%) showed a clear link between a newly created rule and the subsequent layout change. Partition additions were followed by a shift to grouped layouts in 16 cases; order rule additions preceded transitions to sequenced or ranked layouts in 7 cases, such as P1's shift to a ranked list after creating an order rule and P9's shift to a timeline after expressing temporal ordering preferences; hierarchy accumulation preceded table layouts in 6 cases as growing field-level rules made tabular representations increasingly appropriate.

Individual rule profiles produce distinct UI trajectories within the same task in Maru. The personalization effect is most visible when comparing participants who performed the same assigned task. Among the six picnic planners in the Maru condition, each user's layout trajectory was shaped by their individual rule profile: P4's temporal order rules (*"order lunch first"*) produced timeline layouts, while P9's partition rules across snacks, drinks, and activities produced accordion and tabbed layouts. The divergence is sharpest among the three participants who independently planned Barcelona trips in S3. P6, with rich rules across all four IA elements (8 vocabulary, 5 order, 20 partition, 55 hierarchy), progressed through accordion, diagram, and timeline layouts. P11, with minimal interaction and no vocabulary or order rules (38 total), remained on map+tabs layout for five consecutive generations. P12, whose interactions were predominantly deletions creating partition exclusion rules (24 total), converged to simplified tabs. Across all system sessions, this relationship between rule richness and layout diversity was consistent: participants with fewer than 20 rules experienced an average of 1.5 distinct layout types, rising to 2.3 types with 20-50 rules and 3.0 with 50 or more. This suggests that different rule states produce meaningfully different output spaces, and that accumulated structural intent can bound which layouts a GenUI system arrives at rather than converging on a single optimal one.

7 Discussion

7.1 Designing for Persistence Across Levels and Contexts in GenUI

Our results suggest that persistence is a key factor in making user-aligned GenUI for information tasks; however, we address only IA-level structural persistence. Yet UIs have many more semantic and visual layers, and our findings hint that different users want different layers persisted. Non-expert users were largely satisfied with IA-level persistence, while users with more design experience sought control at finer granularities like layout type and component

structure. This points toward a layered model of persistence for future GenUI systems: from high-level structural rules accessible to all users, down to component-level handles for users with the vocabulary to use them.

If levels concern how much of the UI persists, scope concerns how long and where. Our findings show that unbounded rules accumulated beyond their useful context by bleeding across sub-task boundaries and persisting long after users had moved on. What our findings point to is not less persistence but a boundary on it. Future systems need mechanisms that let rules expire or be scoped to sub-task contexts. A task-schema [12] could determine whether a rule belongs to the current sub-task and should be carried into it at all, while a decay mechanism over accumulated context [89] could down-weight rules the user has stopped acting on. Alongside these, systems that surface what the model has assumed [25, 105] could elicit preferences the user never thought to state.

Beyond scoping within a session, a natural extension is reuse across instances of a task. In exploring this, we found the rules as we designed them were difficult to carry across tasks without more observation of the user and an explicit mechanism for generalization. Their concreteness is what lets them determine specific layout and content decisions, but it also ties them to the task they were formed in. What survived abstraction was a general sense of what the user cared about rather than something that could map to a UI decision. One direction may be to treat abstracted user models as a basis for IA rules, where approaches that synthesize recurring behavior across sessions [108] supply the higher-level structure from which task-specific rules are formed.

7.2 IA as a Shared Language for End-User Customization

It has long been observed that people rarely customize their software due to high interaction costs and the requirement for technical expertise [64]. GenUI begins to shift this dynamic by replacing programming with natural language and interaction as the medium for customization [41, 62]. Our findings suggest that IA rules represent a promising layer for this shift—with 97% of rules built without deliberate schema engagement, the IA layer absorbs customization intent from natural interaction rather than requiring users to engage in architectural modeling. Hierarchy illustrates this most clearly. It was the element participants found hardest to grasp, yet it accounted for the second-largest share of user-created rules—users produced structure they could not name. Rather than teaching users a schema, then, systems can let them act through operations they already recognize and infer the structure from what they do.

Throughout the study, we observed a behavioral shift. Users gradually learned to communicate more deliberately with Maru by specifying preferences explicitly once they saw the system would respond. Over time, users shifted to more targeted refinements, trusting the system to carry forward established preferences as accumulated rules handled more of the structure. Within our sessions, this suggests IA rules can operate as a shared language users grow more comfortable working through, though how such a shift develops over longer use remains open.

7.3 Establishing Meaningful Boundaries in Non-Deterministic GenUI

Our findings suggest GenUI research should shift from identifying the single optimal layout for a query toward ensuring the generative design space is shaped by the user's preferences. The trip comparison shows this well, as users doing the same task produced their own distinct and valid UI trajectories through Maru. This points to a broader shift in how we think about GenUI. Rather than asking "What is the best layout for this task?" the more useful question is "What conditions make a layout satisfying for this user?" IA rules represent one way to establish those conditions by guiding the system toward outputs that fit within the user's accumulated structural intent. The non-deterministic nature of generation therefore becomes less something to eliminate and more something to structure, since variety can be acceptable and even desirable as long as outputs stay within meaningful boundaries. This also suggests that generation becomes more valuable later in a task than at the start. The difference between Maru and the baseline was largest in the second half of sessions, once users had built up structure of their own. Early on, a well-designed default may work just as well, and IA-guided personalization matters more as a user's needs move away from it. What remains open is how to design systems that help users establish and refine those boundaries over time, and how to ensure that the range of outputs the system can produce is visible enough that users can steer it intentionally.

7.4 Towards the GenUI "Stack"

Software applications are built on stacks—layered technologies each handling a distinct concern. We envision a similar stack for GenUI will also begin to take shape: task-driven data models structure what information to surface for a given activity [12, 47], UI specification languages provide a shared grammar for describing interface intent [8, 27], and malleable interface frameworks determine how generated outputs can be customized [61, 63]. IA sits between these layers, translating the structural logic users build during a task into concrete generation decisions and persisting it across the session. The stack must also span activities, not only layers, as structural abstractions of this kind appear in generative systems for other domains [4, 7]. Participants who took Maru beyond sensemaking suggest the four elements can hold as a foundation while their relative importance shifts across activities. The broader challenge of building GenUI systems aligned with individual users over time will require the community to develop and connect these layers together. We hope IA persistence serves as a concrete starting point for enriching the tooling around GenUI through multiple layers.

Acknowledgments

This work was supported by Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No.2021-0-01347, Video Interaction Technologies Using Object-Oriented Video Modeling). This work was also supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No.RS-2024-00406715). We thank all our study participants for their time and valuable insights. The first author sincerely thanks all team members for their unwavering support and dedication.

References

[1] 2026. Lovable. https://lovable.dev/. Accessed January 2026.

[2] Jae-wook Ahn and Peter Brusilovsky. 2010. What you see is what you search: adaptive visual search framework for the web. In *Proceedings of the 19th international conference on World wide web*. 1049–1050.

[3] Sérgio Alves, Ricardo Costa, Kyle Montague, and Tiago Guerreiro. 2024. Citizen-Led Personalization of User Interfaces: Investigating How People Customize Interfaces for Themselves and Others. *Proceedings of the ACM on Human-Computer Interaction* 8, CSCW2 (2024), 1–23.

[4] Tyler Angert, Miroslav Suzara, Jenny Han, Christopher Pondoc, and Hariharan Subramonyam. 2023. Spellburst: A node-based interface for exploratory creative coding with natural language prompts. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*. 1–22.

[5] Anthropic. 2024. Introducing Artifacts. https://www.anthropic.com/news/introducing-artifacts. Accessed: March 30, 2026.

[6] Anthropic. 2026. Claude now creates interactive charts, diagrams and visualizations. https://claude.com/blog/claude-builds-visuals. Accessed: 2026-03-31.

[7] Timothy J Aveni, Hila Mor, Armando Fox, and Björn Hartmann. 2025. Generative Trigger-Action Programming with Ply. In *Proceedings of the 38th Annual ACM Symposium on User Interface Software and Technology*. 1–17.

[8] 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.

[9] Edward O. Benson and David R. Karger. 2013. Cascading tree sheets and recombinant HTML: better encapsulation and retargeting of web content. In *Proceedings of the 22nd International Conference on World Wide Web* (Rio de Janeiro, Brazil) *(WWW '13)*. Association for Computing Machinery, New York, NY, USA, 107–118. doi:10.1145/2488388.2488399

[10] Dan Brown. 2010. Eight principles of information architecture. *Bulletin of the American Society for Information Science and Technology* 36, 6 (2010), 30–34.

[11] Josep Maria Brunetti. 2013. Design and evaluation of overview components for effective semantic data exploration. In *Proceedings of the 3rd International Conference on Web Intelligence, Mining and Semantics*. 1–8.

[12] 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*. 1–20.

[13] Joseph Chee Chang, Nathan Hahn, Yongsung Kim, Julina Coupland, Bradley Breneisen, Hannah S Kim, John Hwong, and Aniket Kittur. 2021. When the tab comes due: challenges in the cost structure of browser tab usage. In *Proceedings of the 2021 CHI conference on human factors in computing systems*. 1–15.

[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] Joseph Chee Chang, Yongsung Kim, Victor Miller, Michael Xieyang Liu, Brad A Myers, and Aniket Kittur. 2021. Tabs. do: Task-centric browser tab management. In *The 34th Annual ACM Symposium on User Interface Software and Technology*. 663–676.

[17] Joseph Chee Chang, Amy X Zhang, Jonathan Bragg, Andrew Head, Kyle Lo, Doug Downey, and Daniel S Weld. 2023. Citesee: Augmenting citations in scientific papers with persistent and personalized historical context. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*. 1–15.

[18] Jiaqi Chen, Yanzhe Zhang, Yutong Zhang, Yijia Shao, and Diyi Yang. 2025. Generative interfaces for language models. *arXiv preprint arXiv:2508.19227* (2025).

[19] Xiang'Anthony Chen, Tiffany Knearem, and Yang Li. 2025. The genui study: Exploring the design of generative ui tools to support ux practitioners and beyond. In *Proceedings of the 2025 ACM Designing Interactive Systems Conference*. 1179–1196.

[20] Mira Dontcheva, Steven M Drucker, Geraldine Wade, David Salesin, and Michael F Cohen. 2006. Summarizing personal web browsing sessions. In *Proceedings of the 19th annual ACM symposium on User interface software and technology*. 115–124.

[21] James Fogarty, Desney Tan, Ashish Kapoor, and Simon Winder. 2008. CueFlik: interactive concept learning in image search. In *Proceedings of the sigchi conference on human factors in computing systems*. 29–38.

[22] Raymond Fok, Joseph Chee Chang, Marissa Radensky, Pao Siangliulue, Jonathan Bragg, Amy X Zhang, and Daniel S Weld. 2025. Facets, taxonomies, and syntheses: Navigating structured representations in llm-assisted literature review. *arXiv preprint arXiv:2504.18496* (2025).

[23] Raymond Fok, Nedim Lipka, Tong Sun, and Alexa F Siu. 2024. Marco: Supporting business document workflows via collection-centric information foraging with large language models. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*. 1–20.

[24] Krzysztof Gajos and Daniel S Weld. 2004. SUPPLE: automatically generating user interfaces. In *Proceedings of the 9th international conference on Intelligent user interfaces*. 93–100.

[25] Simret Araya Gebreegziabher, Yukun Yang, Elena L Glassman, and Toby Jia-Jun Li. 2025. Supporting Co-Adaptive Machine Teaching through Human Concept Learning and Cognitive Theories. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*. 1–18.

[26] Camille Gobert and Michel Beaudouin-Lafon. 2023. Lorgnette: Creating Malleable Code Projections. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology* (San Francisco, CA, USA) *(UIST '23)*. Association for Computing Machinery, New York, NY, USA, Article 71, 16 pages. doi:10.1145/3586183.3606817

[27] Google. 2025. A2UI: A Protocol for Agent-Driven Interfaces. https://a2ui.org/. Accessed: 2026-03-31.

[28] Google Labs. 2025. Disco. https://labs.google/disco. Accessed: March 30, 2026.

[29] Jens Emil Grønbæk, Banu Saatçi, Carla F. Griggio, and Clemens Nylandsted Klokmose. 2021. MirrorBlender: Supporting Hybrid Meetings with a Malleable Video-Conferencing System. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (Yokohama, Japan) *(CHI '21)*. Association for Computing Machinery, New York, NY, USA, Article 451, 13 pages. doi:10.1145/3411764.3445698

[30] Mariam Guizani. 2022. A decade of information architecture in HCI: a systematic literature review. *arXiv preprint arXiv:2202.13412* (2022).

[31] Nathan Hahn, Joseph Chee Chang, and Aniket Kittur. 2018. Bento browser: Complex mobile search without tabs. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. 1–12.

[32] Han L Han. 2020. Designing Representations for Digital Documents. In *Adjunct Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology*. 174–178.

[33] Marti A Hearst and Duane Degler. 2013. Sewing the seams of sensemaking: A practical interface for tagging and organizing saved search results. In *Proceedings of the symposium on human-computer interaction and information retrieval*. 1–10.

[34] Andrew Hinton. 2009. The machineries of context. *Journal of Information Architecture* 1, 1 (2009), 37–47.

[35] Orland Hoeber. 2025. Design principles for exploratory search interfaces. In *Proceedings of the 2025 acm sigir conference on human information interaction and retrieval*. 12–22.

[36] Yue Jiang, Eldon Schoop, Amanda Swearngin, and Jeffrey Nichols. 2025. Iluvui: Instruction-tuned language-vision modeling of uis from machine conversations. In *Proceedings of the 30th International Conference on Intelligent User Interfaces*. 861–877.

[37] Hyeonsu B Kang, Tongshuang Wu, Joseph Chee Chang, and Aniket Kittur. 2023. Synergi: A mixed-initiative system for scholarly synthesis and sensemaking. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*. 1–19.

[38] David J Kasik. 1982. A user interface management system. In *Proceedings of the 9th annual conference on Computer graphics and interactive techniques*. 99–106.

[39] Christos Katsanos, Nikolaos Tselios, and Nikolaos Avouris. 2008. AutoCardSorter: Designing the information architecture of a web site using latent semantic analysis. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. 875–878.

[40] Alan Kay and Adele Goldberg. 1977. Personal dynamic media. *Computer* 10, 3 (1977), 31–41.

[41] Tae Soo Kim, DaEun Choi, Yoonseo Choi, and Juho Kim. 2022. Stylette: Styling the web with natural language. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*. 1–17.

[42] Aniket Kittur, Andrew M Peters, Abdigani Diriye, Trupti Telang, and Michael R Bove. 2013. Costs and benefits of structured information foraging. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 2989–2998.

[43] Gary Klein, Brian Moon, and Robert R Hoffman. 2006. Making sense of sensemaking 1: Alternative perspectives. *IEEE intelligent systems* 21, 4 (2006), 70–73.

[44] Clemens N Klokmose, 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.

[45] Andrew Kuznetsov, Joseph Chee Chang, Nathan Hahn, Napol Rachatasumrit, Bradley Breneisen, Julina Coupland, and Aniket Kittur. 2022. Fuse: In-Situ sensemaking support in the browser. In *Proceedings of the 35th Annual ACM Symposium on User Interface Software and Technology*. 1–15.

[46] Andrew Kuznetsov, Michael Xieyang Liu, and Aniket Kittur. 2024. Tasks, Time, and Tools: Quantifying Online Sensemaking Efforts Through a Survey-based Study. *arXiv preprint arXiv:2411.07206* (2024).

[47] Michelle S Lam, Omar Shaikh, Hallie Xu, Alice Guo, Diyi Yang, Jeffrey Heer, James A Landay, and Michael S Bernstein. 2025. Just-In-Time Objectives: A General Approach for Specialized AI Interactions. *arXiv preprint arXiv:2510.14591* (2025).

[48] Yaniv Leviathan, Dani Valevski, et al. 2025. *Generative UI: LLMs are effective UI generators*. Technical Report. Technical report, Google Research, 2025. Available at generativeui. github. io.

[49] James Lin, Jeffrey Wong, Jeffrey Nichols, Allen Cypher, and Tessa A Lau. 2009. End-user programming of mashups with vegemite. In *Proceedings of the 14th international conference on Intelligent user interfaces*. 97–106.

[50] Geoffrey Litt, Josh Horowitz, Peter van Hardenberg, and Todd Matthews. 2025. Malleable Software: Restoring User Agency in a World of Locked-Down Apps. https://www.inkandswitch.com/essay/malleable-software/.

[51] 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.

[52] 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.

[53] Michael Xieyang Liu, Andrew Kuznetsov, Yongsung Kim, Joseph Chee Chang, Aniket Kittur, and Brad A Myers. 2022. Wigglite: Low-cost information collection and triage. In *Proceedings of the 35th Annual ACM Symposium on User Interface Software and Technology*. 1–16.

[54] Michael Xieyang Liu, Tongshuang Wu, Tianying Chen, Franklin Mingzhe Li, Aniket Kittur, and Brad A Myers. 2024. Selenite: Scaffolding online sensemaking with comprehensive overviews elicited from large language models. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*. 1–26.

[55] Ping Luo, Pedro Szekely, and Robert Neches. 1993. Management of interface design in HUMANOID. In *Proceedings of the INTERACT'93 and CHI'93 Conference on Human Factors in Computing Systems*. 107–114.

[56] Rongjun Ma, Henrik Lassila, Leysan Nurgalieva, and Janne Lindqvist. 2023. When browsing gets cluttered: exploring and modeling interactions of browsing clutter, browsing habits, and coping. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*. 1–29.

[57] Gary Marchionini. 2006. Exploratory search: from finding to understanding. *Commun. ACM* 49, 4 (2006), 41–46.

[58] Gary Marchionini and Ben Brunk. 2003. Towards a general relation browser: A GUI for information architects. (2003).

[59] Damien Masson, Sylvain Malacria, Géry Casiez, and Daniel Vogel. 2024. DirectGPT: A Direct Manipulation Interface to Interact with Large Language Models. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) *(CHI '24)*. Association for Computing Machinery, New York, NY, USA, Article 975, 16 pages. doi:10.1145/3613904.3642462

[60] Yu Mei, Yuanxi Wang, Shiyi Wang, Qingyang Wan, Zhuojun Li, Chun Yu, Weinan Shi, and Yuanchun Shi. 2025. Interquest: A mixed-initiative framework for dynamic user interest modeling in conversational search. In *Proceedings of the 38th Annual ACM Symposium on User Interface Software and Technology*. 1–23.

[61] 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*. 1–25.

[62] Bryan Min, Peiling Jiang, Zhicheng Huang, and Haijun Xia. 2026. Gradual Generation of User Interfaces as a Design Method for Malleable Software. *arXiv preprint arXiv:2601.17975* (2026).

[63] Bryan Min and Haijun Xia. 2025. Meridian: A Design Framework for Malleable Overview-Detail Interfaces. In *Proceedings of the 38th Annual ACM Symposium on User Interface Software and Technology*. 1–14.

[64] Bonnie A Nardi. 1993. *A small matter of programming: perspectives on end user computing*. MIT press.

[65] Phong H Nguyen, Kai Xu, Andy Bardill, Betul Salman, Kate Herd, and BL William Wong. 2016. SenseMap: Supporting browser-based online sensemaking through analytic provenance. In *2016 IEEE Conference on Visual Analytics Science and Technology (VAST)*. IEEE, 91–100.

[66] Jeffrey Nichols, Brad A Myers, Michael Higgins, Joseph Hughes, Thomas K Harris, Roni Rosenfeld, and Mathilde Pignol. 2002. Generating remote control interfaces for complex appliances. In *Proceedings of the 15th annual ACM symposium on User interface software and technology*. 161–170.

[67] Jeffrey Nichols, Brad A Myers, and Kevin Litwack. 2004. Improving automatic interface generation with smart templates. In *Proceedings of the 9th international conference on Intelligent user interfaces*. 286–288.

[68] Jeffrey Nichols, Brad A Myers, and Brandon Rothrock. 2006. UNIFORM: automatically generating consistent remote control user interfaces. In *Proceedings of the SIGCHI conference on Human Factors in computing systems*. 611–620.

[69] Jeffrey Nichols, Brandon Rothrock, Duen Horng Chau, and Brad A Myers. 2006. Huddle: automatically generating interfaces for systems of multiple connected appliances. In *Proceedings of the 19th annual ACM symposium on User interface software and technology*. 279–288.

[70] Fredrik Ohlin. 2012. The role of information architecture in context-aware adaptive systems. *Journal of Information Architecture* 4, 1-2 (2012), 29–39.

[71] Srishti Palani, Yingyi Zhou, Sheldon Zhu, and Steven P Dow. 2022. InterWeave: Presenting Search Suggestions in Context Scaffolds Information Search and Synthesis. In *Proceedings of the 35th Annual ACM Symposium on User Interface Software and Technology*. 1–16.

[72] Haekyu Park, Gonzalo Ramos, Jina Suh, Christopher Meek, Rachel Ng, and Mary Czerwinski. 2023. FoundWright: A System to Help People Re-find Pages from Their Web-history. *arXiv preprint arXiv:2305.07930* (2023).

[73] Paulo Pinheiro da Silva. 2000. User interface declarative models and development environments: A survey. In *International Workshop on Design, Specification, and Verification of Interactive Systems*. Springer, 207–226.

[74] David J Piorkowski, Scott D Fleming, Irwin Kwan, Margaret M Burnett, Christopher Scaffidi, Rachel KE Bellamy, and Joshua Jordahl. 2013. The whats and hows of programmers' foraging diets. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 3063–3072.

[75] Peter Pirolli and Stuart Card. 1995. Information foraging in information access environments. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. 51–58.

[76] Peter Pirolli and Stuart Card. 2005. The sensemaking process and leverage points for analyst technology as identified through cognitive task analysis. In *Proceedings of international conference on intelligence analysis*, Vol. 5. McLean, VA, USA, 2–4.

[77] Napol Rachatasumrit, Gonzalo Ramos, Jina Suh, Rachel Ng, and Christopher Meek. 2021. Forsense: Accelerating online research through sensemaking integration and machine research support. In *Proceedings of the 26th International Conference on Intelligent User Interfaces*. 608–618.

[78] Soo Young Rieh, Kevyn Collins-Thompson, Preben Hansen, and Hye-Jung Lee. 2016. Towards searching as a learning process: A review of current perspectives and future directions. *Journal of Information Science* 42, 1 (2016), 19–34.

[79] George Robertson, Mary Czerwinski, Kevin Larson, Daniel C Robbins, David Thiel, and Maarten Van Dantzich. 1998. Data mountain: using spatial memory for document management. In *Proceedings of the 11th annual ACM symposium on User interface software and technology*. 153–162.

[80] Kevin Ros, Kedar Takwane, Ashwin Patil, Rakshana Jayaprakash, and ChengXiang Zhai. 2024. TextData: Save What You Know and Find What You Don't. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2806–2810.

[81] Louis Rosenfeld and Peter Morville. 2002. *Information architecture for the world wide web*. " O'Reilly Media, Inc.".

[82] Nirmal Roy, Manuel Valle Torre, Ujwal Gadiraju, David Maxwell, and Claudia Hauff. 2021. Note the highlight: incorporating active reading tools in a search as learning environment. In *Proceedings of the 2021 conference on human information interaction and retrieval*. 229–238.

[83] Tuukka Ruotsalo, Jaakko Peltonen, Manuel JA Eugster, Dorota Glowacka, Aki Reijonen, Giulio Jacucci, Petri Myllymäki, and Samuel Kaski. 2014. Intentradar: search user interface that anticipates user's search intents. In *CHI'14 Extended Abstracts on Human Factors in Computing Systems*. 455–458.

[84] Daniel M Russell, Mark J Stefik, Peter Pirolli, and Stuart K Card. 1993. The cost structure of sensemaking. In *Proceedings of the INTERACT'93 and CHI'93 conference on Human factors in computing systems*. 269–276.

[85] Bahareh Sarrafzadeh, Alexandra Vtyurina, Edward Lank, and Olga Vechtomova. 2016. Knowledge graphs versus hierarchies: An analysis of user behaviours and perspectives in information seeking. In *Proceedings of the 2016 ACM on Conference on Human Information Interaction and Retrieval*. 91–100.

[86] Arvind Satyanarayan, Dominik Moritz, Kanit Wongsuphasawat, and Jeffrey Heer. 2016. Vega-lite: A grammar of interactive graphics. *IEEE transactions on visualization and computer graphics* 23, 1 (2016), 341–350.

[87] Paul Van Schaik, Raza Habib Muzahir, and Mike Lockyer. 2015. Automated computational cognitive-modeling: goal-specific analysis for large websites. *ACM Transactions on Computer-Human Interaction (TOCHI)* 22, 3 (2015), 1–29.

[88] Shilad Sen, Shyong K Lam, Al Mamunur Rashid, Dan Cosley, Dan Frankowski, Jeremy Osterhouse, F Maxwell Harper, and John Riedl. 2006. Tagging, communities, vocabulary, evolution. In *Proceedings of the 2006 20th anniversary conference on Computer supported cooperative work*. 181–190.

[89] 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.

[90] Luigi Spagnolo, Davide Bolchini, Paolo Paolini, Nicoletta Di Blas, et al. 2010. Beyond findability: Search-enhanced information architecture for content-intensive Rich Internet applications. *Journal of Information Architecture* 2, 1 (2010), 19–36.

[91] Hariharan Subramonyam, Colleen Seifert, Priti Shah, and Eytan Adar. 2020. Texsketch: Active diagramming through pen-and-ink annotations. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. 1–13.

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

[93] Noi Sukaviriya, Srdjan Kovacevic, James D Foley, Brad A Myers, Dan R Olsen Jr, and Matthias Schneider-Hufschmidt. 1994. Model-based user interfaces: What are they and why should we care?. In *Proceedings of the 7th annual ACM symposium on User interface software and technology*. 133–135.

[94] Priyan Vaithilingam, Elena L. Glassman, Jeevana Priya Inala, and Chenglong Wang. 2024. DynaVis: Dynamically Synthesized UI Widgets for Visualization Editing. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) *(CHI '24)*. Association for Computing Machinery, New York, NY, USA, Article 985, 17 pages. doi:10.1145/3613904.3642639

[95] Priyan Vaithilingam and Philip J. Guo. 2019. Bespoke: Interactively Synthesizing Custom GUIs from Command-Line Applications By Demonstration. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology* (New Orleans, LA, USA) *(UIST '19)*. Association for Computing Machinery, New York, NY, USA, 563–576. doi:10.1145/3332165.3347944

[96] Priyan Vaithilingam, Munyeong Kim, Frida-Cecilia Acosta-Parenteau, Daniel Lee, Amine Mhedhbi, Elena L Glassman, and Ian Arawjo. 2025. Semantic Commit: Helping Users Update Intent Specifications for AI Memory at Scale. In *Proceedings of the 38th Annual ACM Symposium on User Interface Software and Technology*. 1–18.

[97] Vercel. 2023. v0 by Vercel. https://v0.dev

[98] Laton Vermette, Parmit Chilana, Michael Terry, Adam Fourney, Ben Lafreniere, and Travis Kerr. 2015. CheatSheet: A contextual interactive memory aid for web applications. In *Proceedings of the 41st Graphics Interface Conference*. 241–248.

[99] Zehuan Wang, Jiaqi Xiao, Jingwei Sun, and Can Liu. 2025. IntentPrism: Human-AI Intent Manifestation for Web Information Foraging. In *Proceedings of the Extended Abstracts of the CHI Conference on Human Factors in Computing Systems*. 1–11.

[100] 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.

[101] Jason Wu, Eldon Schoop, Alan Leung, Titus Barik, Jeffrey P Bigham, and Jeffrey Nichols. 2024. Uicoder: Finetuning large language models to generate user interface code through automated feedback. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*. 7511–7525.

[102] Yuan Xu, Shaowen Xiang, Yizhi Song, Ruoting Sun, and Xin Tong. 2025. DuetUI: A Bidirectional Context Loop for Human-Agent Co-Generation of Task-Oriented Interfaces. *arXiv preprint arXiv:2509.13444* (2025).

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

[104] Bhada Yun, Dana Feng, Ace S Chen, Afshin Nikzad, and Niloufar Salehi. 2025. Generative ai in knowledge work: Design implications for data navigation and decision-making. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*. 1–19.

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

[106] Xiong Zhang and Philip J. Guo. 2018. Fusion: Opportunistic Web Prototyping with UI Mashups. In *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology* (Berlin, Germany) *(UIST '18)*. Association for Computing Machinery, New York, NY, USA, 951–962. doi:10.1145/3242587.3242632

[107] Xiaolong Zhang, Yan Qu, C Lee Giles, and Piyou Song. 2008. CiteSense: supporting sensemaking of research literature. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 677–680.

[108] Dora Zhao, Michelle S Lam, Diyi Yang, and Michael S Bernstein. 2026. Behavior Latticing: Inferring User Motivations from Unstructured Interactions. *arXiv preprint arXiv:2604.07629* (2026).

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

[110] Chengbo Zheng, Yuanhao Zhang, Zeyu Huang, Chuhan Shi, Minrui Xu, and Xiaojuan Ma. 2024. Disciplink: Unfolding interdisciplinary information seeking process via human-ai co-exploration. In *Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology*. 1–20.

[111] Siyi Zhu, Robert Haisfield, Brendan Langen, and Joel Chan. 2024. Patterns of Hypertext-Augmented Sensemaking. In *Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology*. 1–17.

Appendices

A.1 Literature Coding Process

A.1.1 Full Set of Literature Papers for Each Round

Table A.1: Literature Coding Process Across Rounds

Round	Sensemaking Theory	Interactive Systems
1st	[56, 57, 78, 84]	[13–16, 31, 53, 54, 92]
2nd	[42]	[45, 51, 52, 58, 71, 77, 103]
3rd	[35, 111]	[2, 17, 20–23, 32, 33, 37, 72, 79, 80, 83, 91, 98–100, 102, 107, 110]

A.1.2 Finalized Literature Codebook

Table A.2: Full mapping of IA elements to user needs, UI components, and interactions across coded systems.

Category	Element	User Needs	UI Components	Interactions
Structural	Partition	Group related items laterally for visual comparison	Spatial clusters [2, 33, 35, 71, 72, 77, 79, 92, 100, 103]	Drag-and-drop into groups [16, 45, 77]
		Regroup flexibly as mental model evolves	Tab groups / card clusters [13, 16, 23, 35, 45, 56, 80, 102, 107]	Tag / label assignment [15, 21]
		Keep interest facets distinct and independently adjustable	Color coding [13, 17, 35, 77, 103]	Spatial arrangement [77, 92]
			Faceted sections [13, 15, 16, 21, 35, 57, 58]	
	Hierarchy	Decompose complex tasks into nested subtasks	Tables [14, 22, 51, 52]	Drag-and-drop into containers [16, 45, 51]
		Structure evidence under options	Accordions / toggles [16, 37, 45]	Context menus for nesting [16, 92]
		Maintain provenance of collected information	Nested cards [35, 45, 100, 102]	Inline role assignment [51, 52]
			Node-link diagrams [32, 71, 84, 91, 99, 103]	
Semantic	Order	Surface high-priority items to direct attention	Spatial placement [16, 31, 33, 45, 51, 52, 54, 72]	Drag-and-drop reordering [14, 45, 51, 83]
		Weight criteria by relative importance to reflect current priorities	Weight badges / size emphasis [14, 15, 17, 21, 79, 83]	Weight sliders [14, 21]
		Continuously adjust relative importance across rules	Threshold filtering / collapsing [14, 16, 53]	Priority selectors [16, 31]
				Pin / star actions [31, 52, 53, 98]
	Vocabulary	Anchor comparison structures in personal terminology	Persistent labels / column headers [14, 22, 33, 42, 51, 52, 58, 72]	Inline text input [14, 42, 51, 52, 54]
		Evaluate options consistently using self-defined criteria	Tag chips / filter labels [53, 54, 98, 110]	Tag creation [17, 33, 53, 78, 80, 111]
		Persist personal terms as navigational labels across sessions	Tooltips / inline highlights [17, 20, 54, 80, 98, 107, 110]	Term extraction from content [20, 21, 32, 71, 91, 92]

A.2 Maru Details

A.2.1 Maru Interface

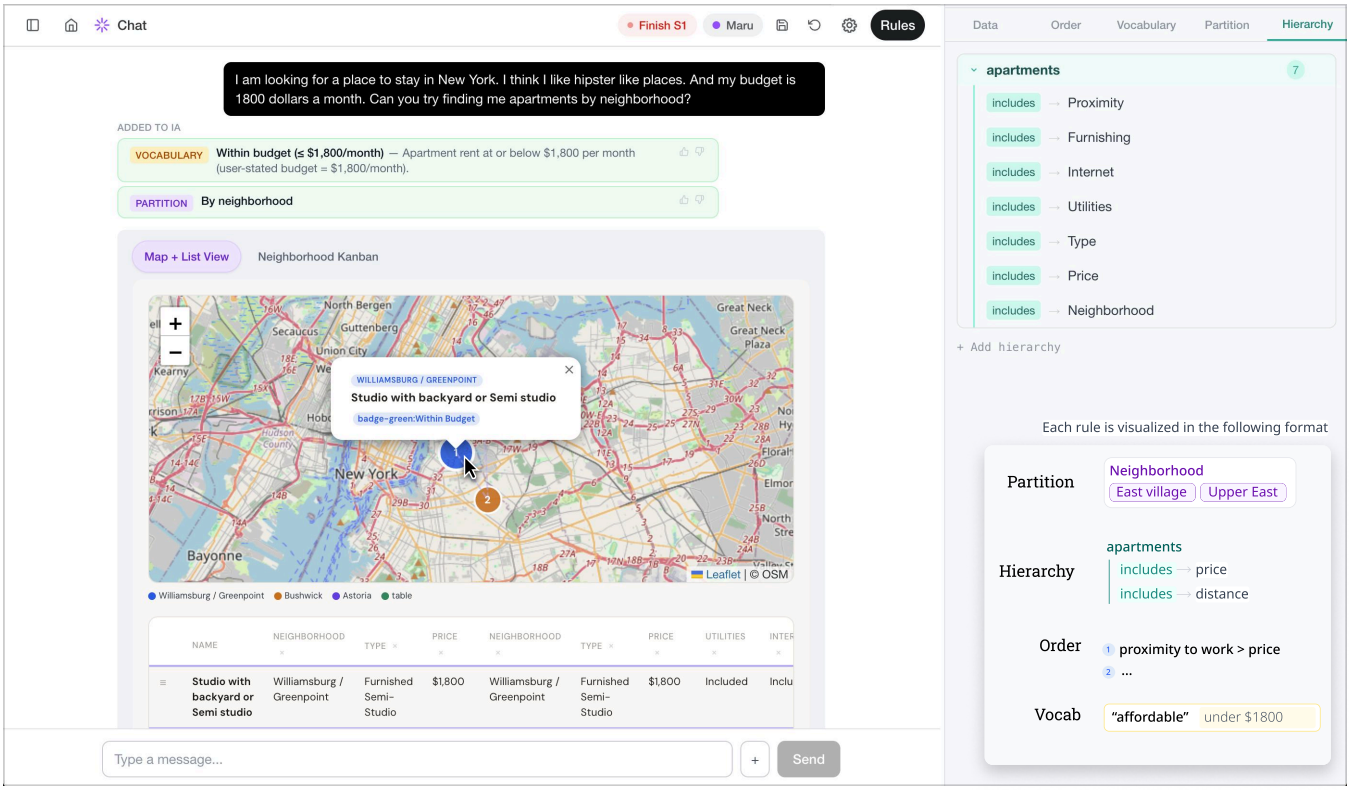

Figure A.1: Overview of the Maru interface. (1) Chat Panel (left): Users submit natural language queries and receive generated UI outputs. Rules detected from queries and interactions are surfaced inline as tagged labels (e.g., Vocabulary, Partition). (2) Generated UI (center): The system renders interactive layout components — here, a Map + List View — populated according to the user's active IA rules. (3) IA Panel (right): The accumulated rule state is visualized across five tabs (Data, Order, Vocabulary, Partition, Hierarchy). Each rule is displayed in a consistent format showing rule type, bound entities, and values, and can be directly inspected, modified, or deleted.

A.2.2 Layouts and Interactions Supported by Maru

Table A.3: The 16 layout types supported by Maru. Components can be composed (e.g., map + tabs, tabs containing card-grid).

Layout	Description
table	Grid layout with sortable columns for uniform items
card-grid	Browsable grid of card-style items with detail fields
tabs	Tab navigation grouping content into selectable sections
accordion	Collapsible sections for hierarchical content exploration
kanban	Board-style columns for drag-and-drop organization
timeline	Sequential/chronological display of events
map	Spatial/geographic layout for location-based data
ranked-list	Ordered list emphasizing ranking and priority
tree-view	Hierarchical tree for nested relationships
diagram	Node-link graph for relationships and connections
calendar	Time-based grid view for scheduling
hero	Large featured item display (fewer than 5 items)
carousel	Sliding/rotating card display
masonry	Grid layout with variable height items
matrix	Multi-dimensional comparison grid
chart	Data visualization (bar, line, pie)

Table A.4: The 10 supported interaction types and their mappings to IA rule dimensions.

Interaction	Rule Type	Description
sort	Order	Sort items by a field. Creates an ordering preference.
move	Order	Reorder an item within a group. Creates a relative ranking rule.
field-star	Order	Star/prioritize a field. Creates a field importance rule.
delete	Partition	Remove item from view. Updates partition membership.
card-move	Partition	Move item between groups. Updates source/target partitions.
lasso-group	Partition	Create new group via lasso selection. Creates a new partition.
field-remove	Hierarchy	Remove a field from an item. Creates a hierarchy-excludes rule.
field-add	Hierarchy	Add a new field to items. Creates a hierarchy-includes rule; the LLM fills values.
apply-annotation	Vocabulary	Apply a term annotation to selected text. Creates a vocabulary definition.
annotate	(logged only)	Open annotation popover. No rule created until applied.

A.3 User Study Details

A.3.1 Procedure

Table A.5: User Study Procedure. Each study session lasted approximately 120 minutes in total.

Step (min.)		Activity
1	(20)	Introduction on research and tutorial for interface
2	(25)	Task Session (S1)
3	(15)	Debrief Session (questionnaire and brief debrief interview)
4	(25)	Task Session (S2)
5	(15)	Debrief Session (questionnaire and brief debrief interview)
6	(20)	Task Session (S3)
7	(20)	Final Interview

A.3.2 Session Task

The two assigned tasks for first two sessions (S1 and S2) represented common information-intensive scenarios:

> **Task 1** You are considering your next step—applying to graduate programs or exploring job opportunities. Research and compare at least 3–4 options across dimensions like research fit or job role, location, funding or salary, and long-term alignment. By the end of the session, have a shortlist with a clear sense of which options you'd prioritize and why.

> **Task 2** You are organizing an outdoor picnic for your lab or team of about 15 people. Plan everything out—food and drinks, activities, supplies, a day timeline, and a budget breakdown across people. By the end of the session, have a concrete plan including a shopping/preparation list, a rough activity timeline, and a budget breakdown.

A.3.3 Post-Session Survey

The post-session survey was administered after each of the two sessions (Session 1 and Session 2) using the same instrument. It consisted of two parts: the NASA Task Load Index (NASA-TLX) and a set of custom items assessing structural alignment.

Table A.6: NASA-TLX Items (administered after each session, 7-point scale: Very Low – Very High)

Dimension	Item
Mental Demand	How much mental and perceptual activity was required?
Physical Demand	How much physical activity was required?
Temporal Demand	How much time pressure did you feel due to the pace of the task?
Performance	How successful were you in accomplishing the task goals?
Effort	How hard did you have to work to attain your level of performance?
Frustration	How irritated, stressed, and annoyed vs. content were you?

Table A.7: Custom Structural Alignment Items (administered after each session, 5-point scale: 1 – 5)

ID	Item
C1	The output was structured the way I wanted.
C2	I felt in control of how the output was organized.
C3	The output felt tailored to my preferences.
C4	I had to work hard to get the structure I wanted.
C5	The system understood what I needed.

A.3.4 Post-Comparison and Final Survey

After completing both sessions, participants answered a post-comparison question and then engaged in a freeform session with the full system, after which a final survey was administered.

Table A.8: Post-Comparison and Final Survey Items

Section	Item	Type
Post-Comparison	Which condition felt more aligned with your preferences?	Choice
	Why?	Open
Final Survey	What was the task you did?	Open
	The system's outputs evolved to better match my preferences over time.	5-point
	I felt the system learned how I wanted information organized.	5-point
	I would use this system for real information tasks.	5-point

A.3.5 Tasks Selected for S3 (Freeform)

Table A.9: S3 Self-Selected Tasks

Participant	Task
P1	Planning a trip to Italy, St. Tropez, and Spain
P2	Finding a gift for a partner
P3	Shopping for clothes
P4	Planning a trip to Hong Kong
P5	Learning plan for beamforming technology
P6	Planning a trip to Barcelona
P7	Creating a pescetarian diet plan
P8	Planning a fixed weekly schedule for 4 months
P9	Trip planning
P10	Planning weekend activities in Seoul
P11	Planning a business trip to Barcelona
P12	Planning a 7-day trip to Barcelona

A.3.6 Baseline and System Prompts

The baseline and Maru conditions shared the same data acquisition, rendering pipeline, and LLM model. The baseline passed user interactions as a flat timestamped log, Maru encoded the same interactions as typed IA rules. Table A.10 summarizes the differences across pipeline stages, and the prompt excerpts below illustrate the contrast.

Both conditions received identical user interactions. The baseline planner prompt included them as:

```
[user query]

## User's prior interactions
[10:32:15] sorted by price
[10:33:01] deleted "Hotel C"
[10:34:22] moved "Hostel A" to "Budget"
```

The Maru planner prompt included the same interactions as structured rules:

```
The user has configured the following IA preferences:

Ordering criteria:
1. price: User prioritizes cost-effectiveness

Partitions:
- Budget: Hostel A, Hostel B
- Luxury: Hotel D

Hierarchy relationships:
- "item" --excludes--> "Hotel C"

IMPORTANT: Use these consistent field names: price, rating, location
```

Stage	Baseline	Maru
Rule detection	Skipped	LLM detects IA rules from user query
Rule retrieval	Skipped	LLM retrieves relevant rules by semantic match
Rule scaffolding	Skipped	Detects implied partition/hierarchy from new data
Codebook pipeline	Skipped	Maps user need to display and interaction patterns
Schema hint	None — field names decided freely by LLM	Built from hierarchy rules: consistent field names enforced
IA rules in planner	Empty	Formatted typed rules: orders, vocabularies, partitions, hierarchies
Interaction history	Appended as flat timestamped log	Encoded as typed IA rules
Interaction → rules	Logged as timestamped strings	Creates typed Order/Partition/Hierarchy/Vocabulary rules
Post-generation extraction	Skipped	Extracts hierarchy rules from plan's field structure

Table A.10: Pipeline differences between the baseline and Maru conditions.