The Anatomy of a Generative Prototyping Session

ANONYMOUS AUTHOR(S)

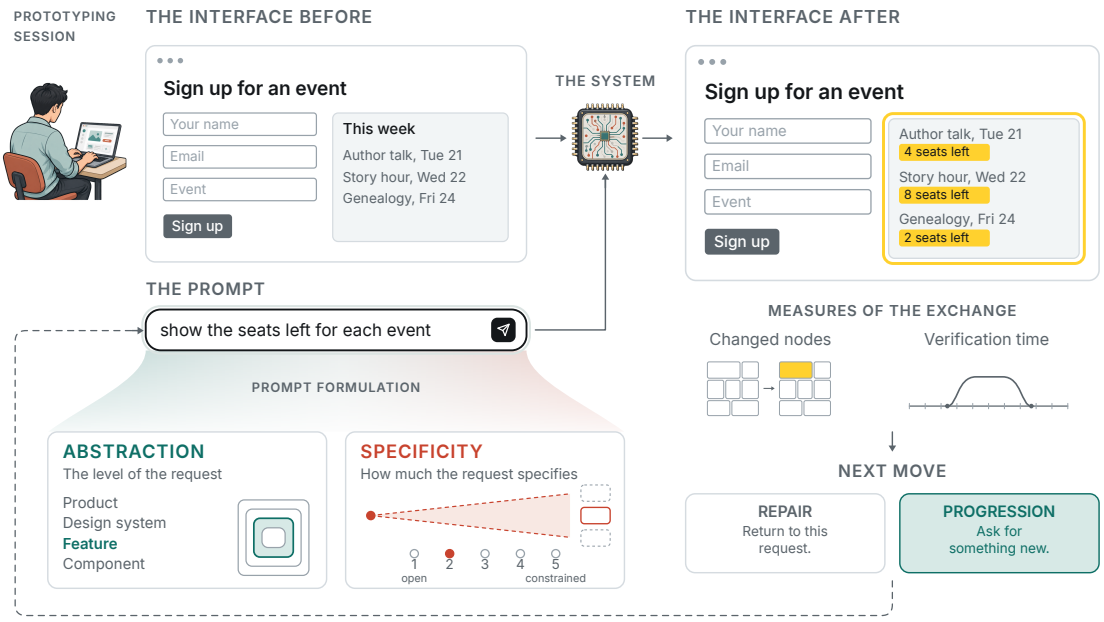


Fig. 1. While prototyping the event sign-up page of a community library, a participant asks the system to *show the seats left for each event*. The feature-level request leaves its concrete implementation largely open to the system. The system returns a modified interface in which each event lists its remaining seats. The participant accepts the design and progresses to the next prompt. We dissect this complete exchange, from the prompt and the generated change to the verification and the next move.

In generative prototyping, the user formulates a request, the system realizes it, and the user inspects the prototype before the next request. This paradigm is rapidly evolving, yet we know little about how these steps relate within a session. In a preregistered study, 100 non-experts used an instrumented generative prototyping environment to complete three tasks in familiar and unfamiliar domains. We dissected 1,643 generations into prompt formulation, extent of change, verification, and next move. We measured the abstraction and specificity of prompts and found both associated with the structural extent of the generated change. In a follow-up study, twelve designers and eight non-experts were interviewed about selected events of their sessions. Based on the qualitative findings, we discuss how reflective practice extends to verification and how expertise informs formulation and evaluation.

CCS Concepts: • **Human-centered computing** → **Human computer interaction (HCI)**.

Additional Key Words and Phrases: generative UI, prototyping, prompting, non-experts, design tools

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2027 Copyright held by the owner/author(s). Publication rights licensed to ACM.
Manuscript submitted to ACM

1 INTRODUCTION

Ask a generative prototyping tool for "an application to manage events" and, within seconds, it returns a fully functioning application with a calendar connected to a database and a search bar. A few prompts later, you may be requesting to "make the clock widget smaller and further to the right". The same system that turns one sentence into a functional prototype can thus turn subsequent iterations into a tedious sequence of small refinements and attempts to undo unwanted changes. The above scenario reflects an emerging paradigm for UI design, which we have only just started to understand.

Generative prototyping tools such as `v0`, `Lovable`, and `Bolt` allow people to produce functional interfaces through natural-language requests, moving rapidly between an initial idea and a polished prototype. Prior work has shown how such tools can support early exploration, make interface production accessible across professional roles, and blend design decisions with their implementation [7, 32]. In conventional prototyping, designers realize a possibility, evaluate it, and develop their understanding of the problem alongside the solution [4, 6, 23]. In generative prototyping, the system produces the concrete realization of a request. Users therefore encounter changes whose details they have not fully specified. Evaluating the result involves establishing what the system changed and judging whether those changes serve the intended design, unlike previous design iteration.

Recent work on generative UI characterizes users' requests according to the semantic levels at which they operate in the interface [25] and documents generated results that omit, add, or extend beyond requested changes [7]. Research on non-expert prompting further indicates that non-experts struggle with the level of abstraction and vocabulary required to interact with generative systems [21]. They also have difficulty translating intended outcomes into prompts, with expertise shaping the formulation of design goals [32]. These studies characterize the requests, the generated results, the users' responses, and the role of expertise. This paper dissects the exchange between users and the generative system itself into measurable parts, providing the first anatomy of generative prototyping.

We conducted two studies using Mark, a custom made generative prototyping environment that reproduces the core features of commercial generative UI tools. In the first study, 100 participants without professional design or software-development experience completed three tasks in which they built prototypes in familiar and unfamiliar domains, and modified an existing interface. The session logs allowed us to examine the relations between prompt formulation, the resulting system response, and participants' subsequent moves, and to compare prompt formulation across domains. In the second study, 12 designers and eight non-experts completed moderated sessions with the same tasks and explained their strategies in interviews. Their accounts allow us to examine reflective practice in generative prototyping and the resources participants draw on to formulate requests and evaluate results.

The paper was guided by the following research questions:

- **RQ1:** How does prompt formulation relate to the structural extent of generated change?
- **RQ2:** How do participants verify generated changes and decide whether to progress or repair?
- **RQ3:** How do domain familiarity and design training shape prompt formulation, verification, and repair?

This paper makes three contributions. First, it introduces the generation event as a unit of analysis for dissecting the exchange into prompt formulation, generated change, verification, and the next move, together with Mark, the experimental instrument that records these events. Second, it provides a corpus of 1,643 events from 100 sessions and a preregistered analysis of empirical data showing how prompt formulation relates to the structural extent of generated change, alongside analyses of verification, repair, and domain familiarity. Third, it develops a qualitative account of generative prototyping as reflective practice, examining the role of domain knowledge and design experience. We

introduce *reflection-in-realization* to characterize how users reconsider their design intentions while examining the system's realization of their requests.

These findings provide an empirical basis for theories of UI prompting and for the design of systems that support users in formulating requests and evaluating generated results. Mark, the data, and the analysis code are released without restriction, to enable reproduction, replication, and extension of the study.[1]

2 RELATED WORK

2.1 Prototyping and Iterative Design

Prototypes externalize design ideas so that they can be explored, communicated, and evaluated before a system is complete. A prototype can give provisional form to the role, look and feel, or implementation of a future artifact [12]. Lim et al. describe prototypes as both filters that foreground particular design dimensions and manifestations that make those dimensions concrete [20]. Because a prototype does not need to resolve every aspect at once, designers can develop and compare alternatives without committing to a complete solution. Sketching preserves this openness through quick, disposable, and deliberately incomplete representations that can be revised before a design converges [4]. Functional interface prototypes additionally make behavior available for evaluation, allowing proposed interactions to be experienced.

Accounts of iterative design describe this reasoning as a reciprocal relation between action and artifact. Schön characterizes designing as a reflective conversation in which a designer makes a move, encounters its consequences in the artifact, interprets that "back-talk", and either develops or revises their understanding of the problem [29]. Later accounts describe this process as the co-evolution of problem and solution representations through successive transformations [6, 10]. Research on design tools further explains how external representations support this process. By making partial states and consequences available for inspection and manipulation, they redistribute cognitive work across designer and artifact [9, 22]. Direct manipulation instantiates this principle at the interface level through a predefined interaction vocabulary applied to represented objects, with the consequences of each operation immediately visible [13, 24]. In generative prototyping, users formulate a request and the system realizes it as a change to the artifact. This fundamentally breaks with previous accounts of prototyping and iterative design, because the reflective conversation proceeds when the consequences of a design move depend on the system's interpretation of the request.

2.2 Prompt-Based Interaction

Prompt-based interaction uses natural-language instructions to direct model behavior [28]. Subramonyam et al. define the *gulf of envisioning* as the difficulty of formulating an instruction while anticipating a model's capabilities and possible outputs [31]. Studies of non-expert prompting find that users often rely on opportunistic revisions, fragile causal theories, and biased evaluation [36]. PromptMaker treated prompts as shared design artifacts for exploring, communicating, and revising intended system behavior [14], and studies of collaborative image generation found that their wording supported exploration and reflection [15].

The formulation of a request is shaped by the user's prior experience, including application-domain knowledge and design expertise. Domain knowledge provides concepts and vocabulary for describing the application context and identifying relevant requirements [11]. Design expertise is associated with richer and more interconnected representations of a design problem, strategic problem framing, the identification of constraints and trade-offs, and the

[1] Project website: https://anatomy.backtothehuman.com/.

evaluation of partial solutions [1, 3, 8]. In prompt-based prototyping, product managers, designers, and engineers make role-specific contributions to defining goals, specifying values, and evaluating model outputs [32]. Professional designers use interface representations to identify incompatibilities between generated behavior and interface structure, while UX practitioners rely on professional judgment to assess the relevance and quality of generated material [19, 27].

2.3 Generative UI and Support for Iteration

Lee characterizes generative UI as an iterative and co-creative process in which responsibility is distributed between designers and computational systems [17]. Studies of professional design practice show that generative AI is used primarily for ideation and early drafts, while limited control over its outputs constrains its use in iterative design work [33, 35]. Within generative UI specifically, Chen et al. found that UX practitioners repeatedly reformulated requests after inspecting generated interfaces, particularly when systems omitted requested details, introduced irrelevant elements, or made local editing difficult [7]. Park et al. similarly found that requests scoped to one part of an interface could produce changes elsewhere, and participants asked for scoped editing, change auditing, and rollback [25].

Research systems address these problems by making the structure, scope, or history of generation more explicit. Park et al. organize design intent across Product, Design System, Feature, and Component levels, using this semantic representation to support articulation, interpretation, and refinement [25]. JELLY represents generated interfaces through evolving task models [5]; SQUIRE exposes a scoped tree representation for UI editing [18]; and Athena makes storyboards, data models, and interface skeletons editable between generations [2]. DesignTrace complements these approaches with a branching history for revisiting alternatives and preserving earlier decisions [26].

We draw on these accounts to dissect each exchange into prompt formulation, generated change, verification, and the next move. By relating these parts across complete sessions, we examine how generative prototyping unfolds and how domain familiarity and design training shape the exchange.

3 MARK AND THE GENERATION EVENT

3.1 The Tool

Mark is a custom generative prototyping environment built for the two studies to reproduce the core features of commercial tools such as `v0`, `Lovable`, and `Bolt` (Figure 2). Participants initiate generation through a prompt bar at the foot of the preview; the bar fades at the end of the page so that the prototype remains visible (Figure 2 **e**). A prompt comprises a text and any references attached to it (Figure 2 **a** – **c**). A reference is either a pointed element, selected in the preview with the pointing tool, or a sketch, drawn on the page with the drawing tool. The system generates changes to the prototype's source code in response to the prompt,[2] and the preview renders the result as soon as the generation completes. The prompt bar expands into a history that lists each prompt with the system's description of the corresponding change, and undo and redo in the history return the prototype to the state before or after a generation (Figure 2 **d**).

[2]Generations were produced by `gemma-4-31b`, an open-source vision-language model served by Cerebras for fast inference, with the same model and configuration throughout both studies.

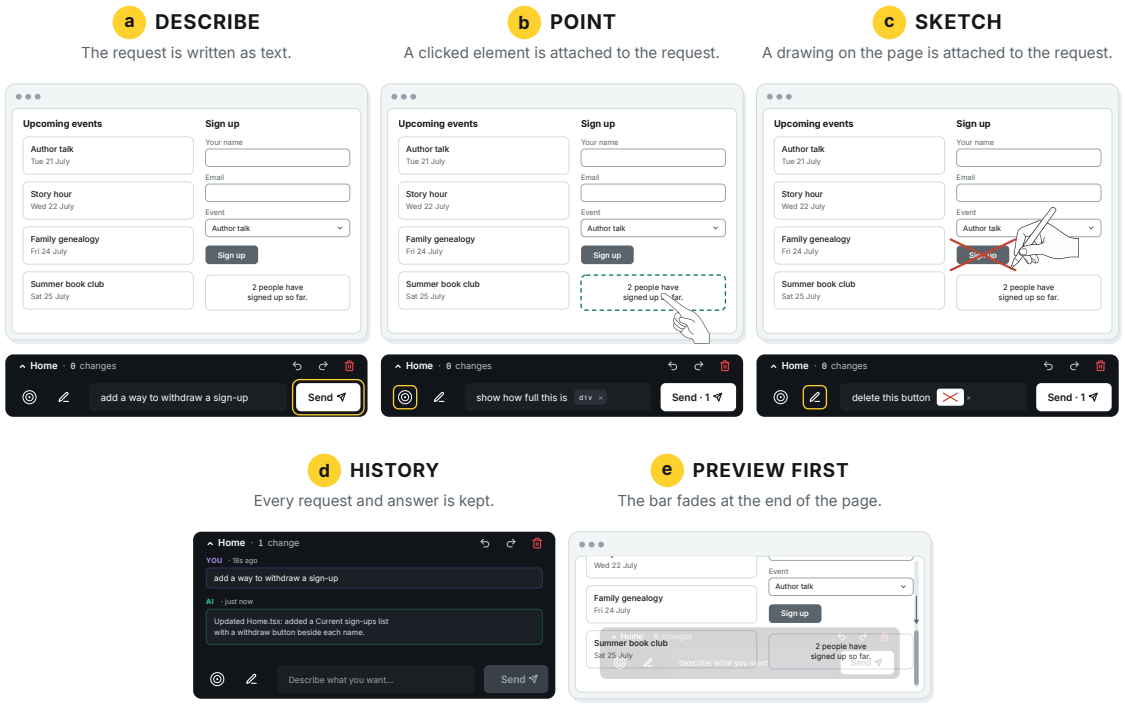


Fig. 2. Mark, the generative prototyping environment used in both studies. The prompt bar below the preview accepts text prompts a, prompts with an attached pointed element b, and prompts with an attached sketch c. The bar expands into a history that lists each prompt with the system's description of the change, and its header provides undo and redo d. The bar fades at the end of the page so that the preview remains visible e.

3.2 Dissecting the Generation Event

The unit of analysis is the **generation event**, the exchange that extends from a prompt to the participant's next move. We characterize prompt formulation through abstraction and specificity.

Abstraction is the level of the prototype at which the prompt operates. We adapt Park et al.'s semantic framework [25] to classify prompts at four levels: Product (e.g., "create an application to manage events"), Design System (e.g., "use a warmer color scheme throughout"), Feature (e.g., "add a way to withdraw a sign-up"), and Component (e.g., "make this button bigger"). The scale is ordinal, from Product as the most abstract to Component as the most concrete. A prompt is assigned the level of its dominant intention; when several equally prominent intentions operate at different levels, it is assigned the most abstract of them.

Specificity is the extent to which a request constrains its realization, that is, how much of the realization the request leaves to the system, coded from 1, when the request states an aim and leaves the change open, to 5, when the request determines the change entirely. "Make this component prettier" and "make this component green" operate at the same level of abstraction, but the second constrains the realization more closely, so its specificity is higher.

The generated change is measured as **changed nodes**. The source of the prototype is stored before and after each generation, each snapshot is parsed as a tree of components, and the number of nodes that the comparison of the two trees marks as added, removed, or modified provides the extent of the change.

Verification time is the time from the completion of the generation to the participant's first subsequent composition-related interaction (typing, pasting, pointing, or sketching), with the intervals during which the browser window is out of focus subtracted. This interval may include both inspection and planning of the next request, and inspection may continue after composition begins.

A generation is **repaired** when the participant's next move returns to the request that produced it, either by prompting for that request again or by undoing the generation; repairs are of three types, **restate**, the request sent again in other words, **refine**, the request made more concrete, and **revert**, the generation undone or its change reversed by a subsequent prompt. A generation is followed by a **progression** when the subsequent prompt asks for something new.

Abstraction, specificity, and the relation between a generation and the following move, with the type of the repair, are assigned by a coding pipeline, a language model run with versioned prompts,[3] whose agreement with two human coders is reported in Section 4. Changed nodes and verification time are computed directly from the research log.

The research log has three layers, from which these measures are computed. For every generation it stores the prompt text, from which abstraction and specificity are coded, the two source snapshots and their structural `diff`, from which changed nodes are counted, and the nodes that a pointed element or a sketch referenced. Interaction telemetry records when the participant starts writing each prompt and by which act (typing, pasting, pointing, or sketching), which marks the end of the verification time, the uses of undo, each of which constitutes a revert, the elements clicked in the preview, each opening of the task brief, and the intervals during which the browser window is out of focus, which are subtracted from the verification time. Participants' informed consent, the start, completion, or timeout of each task, and the outcome of each generation attempt are stored in the research log. An attempt is committed when the change is delivered to the preview, aborted when the participant interrupts it, and failed when the generated source does not pass validation (Appendix B). After an aborted or failed attempt the tool restores the preceding snapshot, and only committed generations are included in the analyses.

3.3 One Task as Generation Events

In the familiar-domain task of Study 1, a participant received a brief asking for a tool to plan and track decorated cookie orders. Figure 3 represents the task as a sequence of generation events, with prompts abridged and some steps merged or altered. The participant began with a Product prompt, "a website that tracks cookie orders", and the system generated the whole page. The verification time before composition of the next request was 32 seconds. The participant then progressed with a series of Component prompts, each pointing at the order form and requesting a delivery choice, a field for inspiration images, or contact details, with each generation changing a few nodes of the form. A Feature prompt then asked for a calendar of upcoming orders and opened a new page. The following request, a white background for the calendar, was reverted through an undo, and the participant returned to the calendar and progressed by asking for the month name in its heading. A Design System prompt then changed the appearance of the whole site, and two further requests added a page of recipes and a save button to the form.

[3] `gpt-5.6-luna`, prompt set `v0.2-exp`; the complete prompts are provided in the supplementary materials.

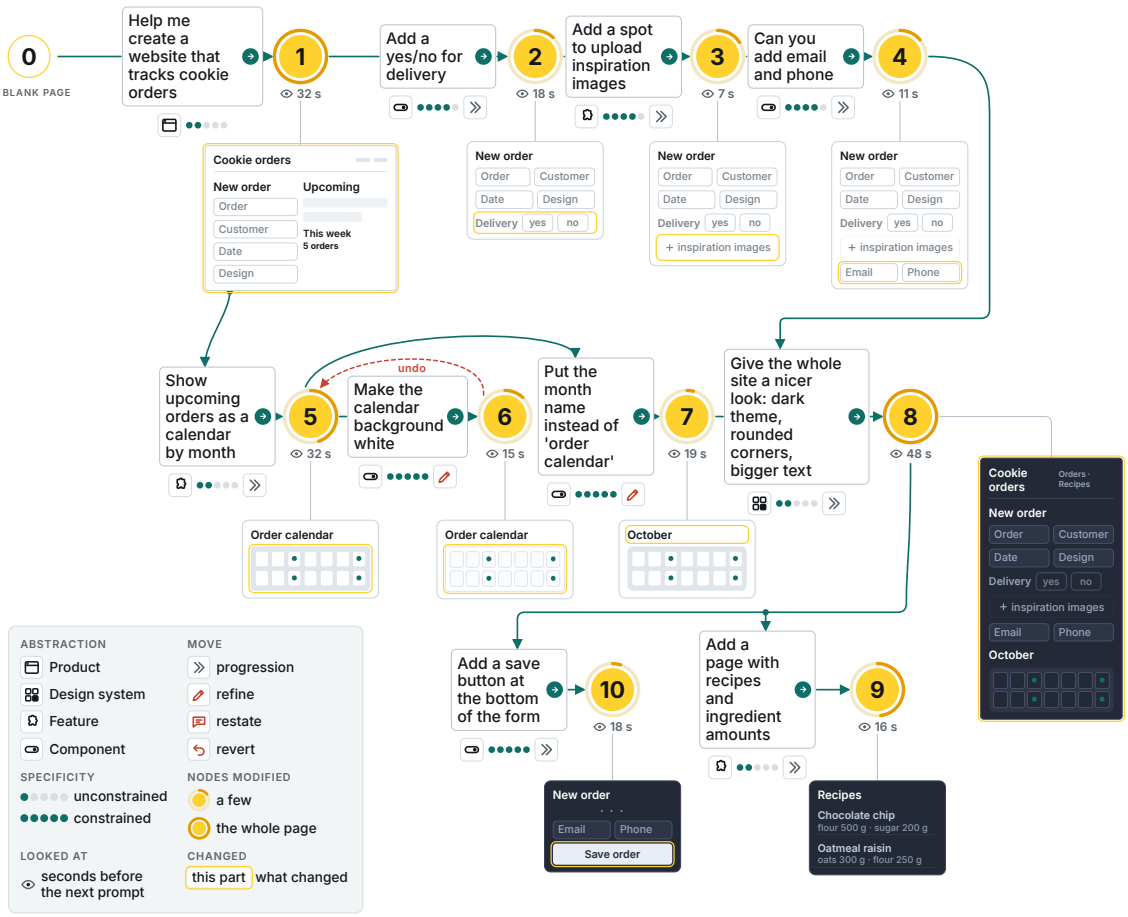


Fig. 3. A sequence of ten generations represented as generation events, constructed from the familiar-domain task of one participant completing Study 1. Each numbered circle represents a generation, and the preceding box contains its prompt.

4 STUDY 1: LOGGED PROTOTYPING SESSIONS

4.1 Design and Participants

In Study 1, we recorded complete prototyping sessions to examine the relations between the parts of a generation event. We used a within-subjects design in which every participant completed three tasks. Participants built prototypes from a blank canvas in an unfamiliar domain (T1) and in a familiar domain (T3), and modified an existing interface (T2). T1 concerned the least familiar of four domains rated in a pre-study survey, and T3 concerned the topic each participant reported knowing most deeply. The order of T1 and T3 was counterbalanced, with T2 always second; 48 participants completed T1 first and 52 completed T3 first. Following a pilot with five participants, we preregistered the study design, hypotheses, and analyses on the Open Science Framework before data collection.[4]

[4]Anonymized view-only link to the preregistration: https://osf.io/8wh4x/?view_only=84f9f96334d44652b9f5ddee4bbf3230;
the materials are deposited in the project at https://osf.io/wgprx/?view_only=01eb2ac01d344b9f832ee38c9f4dd756;
the anonymized Study 1 dataset is available at https://huggingface.co/datasets/backtothehuman/anatomy-generative-prototyping.

The final sample comprised 100 participants recruited on the crowdsourcing platform Prolific[5] with three criteria: English as a first language, an approval rate of at least 99 percent, and no professional experience in UX, design, or software development. A desktop browser was furthermore required. Compensation was a base payment of GBP 9, calibrated on the pilot's median duration at a target of GBP 12 per hour, plus GBP 1 for every five minutes beyond 45 minutes. A session was valid when it was completed with at least one committed generation per task and its research log was complete and internally consistent. Participants were instructed to write their requests themselves. Prompts containing at least eighty typed characters were flagged when the typing rate exceeded ten characters per second, with pasted characters excluded from the calculation. The preregistration specified exclusion and replacement for sessions that were invalid or contained three or more flagged prompts. Of the 142 participants who provided consent, 100 completed all three tasks and 42 did not complete the study. The sample included 49 women, 49 men, and two participants who did not report gender. Age was available for 99 participants (median 35 years, range 18–73).

4.2 Tasks

Each task was introduced by a written brief and had a ten-minute time budget. The unfamiliar- and familiar-domain tasks used the template below, which identifies the future user of the tool and states three needs. Nine slots allow the domain-specific noun phrases to vary while preserving the structure of the brief.

> Imagine {audience} who needs to {core purpose}.
> Right now this is all tracked by hand, and it is getting hard to stay on top of.
>
> Three things matter in their day:
>
> (1) Seeing what is ahead: the {items}, and for each one {detail 1}, {detail 2}, and {detail 3}.
> (2) Recording each new {item} as soon as it comes up.
> (3) Knowing at a glance {summary statistic}.
>
> Build a first prototype of the tool you would give them. Shape it the way {audience} would actually want it.
> There is no right answer.

Each brief states the user's needs and leaves to the participant how each need is realized through the participant's own requests. For T1 we wrote four instances in specialized professional domains (harbor mooring bookings, cold-storage pallet batches, grain milling runs, and auction lot consignments), and each participant received the one they had rated as least familiar in the pre-study survey. For T3 the skeleton was filled at session time by a language model[6] with the activity or subject the participant reported knowing most deeply; a deterministic validator checked the format of every generated brief with a fixed fallback brief (Appendix B).

In T2, every participant started from the same prepared prototype, a working sign-up page for the events of a small community library, with an event list, a sign-up form, and a sign-up counter. The brief stated three needs: a way for visitors to withdraw a sign-up, a view of how full each event is, and a sense of how busy the current week has been. It also stated that the library was satisfied with the rest of the page.

[5] https://www.prolific.com
[6] `gpt-5.1-2025-11-13`, temperature 0, prompt `t3-gen-v1.4`.

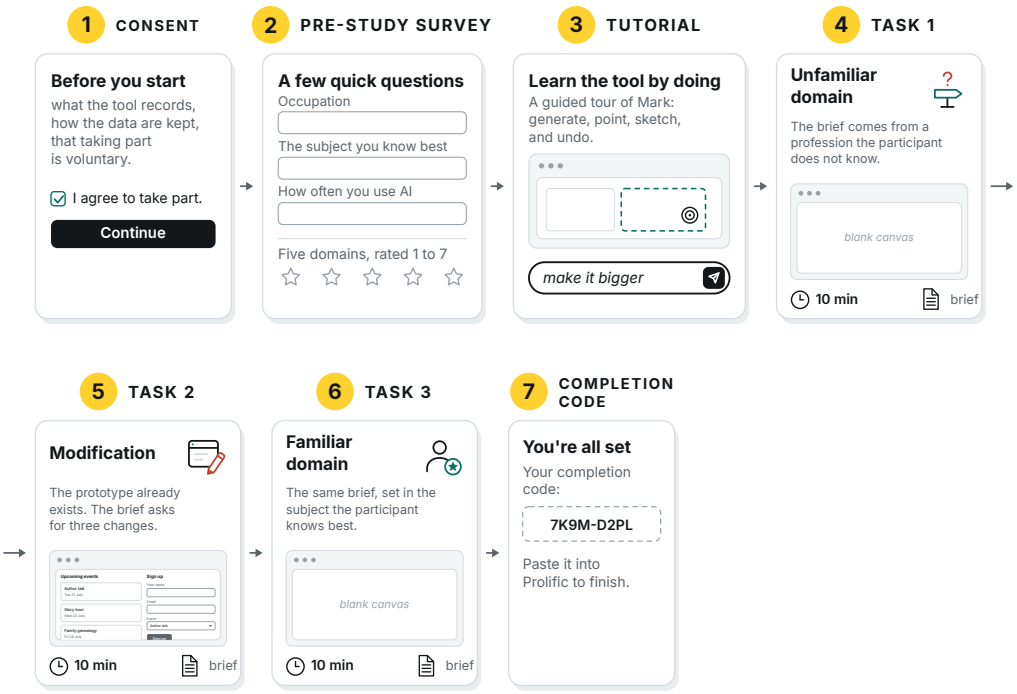


Fig. 4. Study 1 procedure, from consent to completion. After a survey and guided tutorial, participants built a prototype in an unfamiliar domain (T1), modified a library registration page (T2), and built a prototype in a familiar domain (T3). The illustrated order was followed by 48 participants; the other 52 completed T3 before T1, with T2 always second. Participants had ten minutes per task, excluding generation time.

4.3 Procedure

After providing consent, participants completed the pre-study survey, a mandatory guided tutorial, and the three tasks (Figure 4). In the pre-study survey, we asked participants about their occupation, the activity or subject they knew most deeply, their experience with AI tools, their age, and their gender (Appendix A). At the end of the survey, participants rated their knowledge of the four candidate T1 domains and their generated T3 domain on a 7-point scale. During the tutorial, participants generated a page, pointed at an element, sketched, and undid a generation in a practice project.

Each task began with its brief. After reading the brief for T1 and T3, participants completed a four-item questionnaire about their familiarity with the activity described, with each item rated on a 7-point scale (Appendix A). A visible countdown provided ten minutes per task, excluding generation time. Participants could complete a task early once at least one generation had committed. When the countdown expired, the system opened the task completion dialog. After the third task, participants received a completion code.

4.4 Measures

We analyzed generations initiated during the three tasks for which usable versions of the interface code were available both before and after generation (Subsection 5.1). Generations from the tutorial and other activity before the tasks were excluded. Following the definitions in Subsection 3.2, we coded each prompt for abstraction and specificity, counted the

changed nodes to quantify structural change, and calculated verification time from interaction logs. When a subsequent move was available, we classified it as a *progression* or a *repair*, distinguishing *restate*, *refine*, and *revert* repairs.

We computed the **familiarity differential** by subtracting the mean of the four questionnaire ratings for T1 from the corresponding mean for T3. This difference represents how much more familiar participants reported being with the domain of T3 than with that of T1.

4.5 Hypotheses

We preregistered five hypotheses for Study 1:

RQ1: *Prompt formulation and structural change.*

- **H1. Abstraction.** The number of changed nodes increases with the abstraction of the prompt.
- **H2. Specificity.** The number of changed nodes decreases with the specificity of the prompt.

RQ2: *Verification and repair.*

- **H3. Repair.** The lower the specificity of a prompt, the higher the likelihood that its generation is followed by a repair.
- **H4. Verification.** The likelihood that a generation is followed by a repair increases with its verification time.

RQ3: *Domain familiarity.*

- **H5. Familiarity.** Participants write more specific prompts in the familiar-domain task (T3) than in the unfamiliar-domain task (T1).

4.6 Analysis

We coded every committed generation with a language model,[7] following the definitions of abstraction, specificity, and the subsequent move given in Subsection 3.2. We developed the coding prompts over several iterations on pilot sessions.

The pipeline proceeded in three independent calls, one for abstraction, one for specificity, and one for repair. In the first two, the model received the user request, the task, and the state of the prototype before the generation, and it assigned the abstraction level of the request in one call and its specificity score in the other. In the third, the model received the request, a summary of how the interface code had changed, and the text of the next prompt, and it classified the move that followed the generation as a progression or a repair and named the type of repair. We coded *reverts* directly from the undo actions recorded in the log. A generation followed by neither a prompt nor an undo received no classification of the subsequent move.

As per our preregistration, we report H3 and H4 descriptively, because the repair measure did not meet the required inter-rater agreement. The corresponding tests are reported in Appendix C. We retained the inferential tests for H1, H2, and H5, and we discuss the agreement obtained for specificity in Subsection 8.4. For these three tests we used the preregistered models and one-sided directions, and we applied the Holm correction over the five originally planned tests, so that the family-wise error rate is controlled at $\alpha = .05$.

H1 and H2, the extent of the change. H1 states that the number of changed nodes increases with the abstraction of the prompt, and H2 that it decreases with its specificity. Per preregistration, we excluded the first committed generation of each unfamiliar-domain and familiar-domain task, because these generations create the initial prototype from a

[7] `gpt-5.6-luna`. The instructions given to the model are provided in the supplementary materials.

blank canvas, and reported them separately. We fitted a negative-binomial mixed-effects model, with changed nodes as the outcome and abstraction and specificity entered together as numeric predictors. This model accounts for the overdispersion in changed-node counts, whose variance exceeds their mean. A random intercept per participant accounted for the dependence among generations from the same participant. Including both predictors allowed us to estimate the association of each with changed nodes while holding the other constant. Abstraction was scored from 1 (Product) to 4 (Component), so higher abstraction corresponds to a lower score. Specificity was scored from 1 to 5, with higher scores indicating more specific prompts. Both hypotheses therefore predict negative coefficients, which we evaluated using one-sided tests in the preregistered directions.

H3, repair. H3 states that the probability that a generation is followed by a repair decreases with the specificity of its prompt. We report the proportion of generations classified as followed by repair at each specificity level. Generations with neither a following prompt nor an undo were excluded because no subsequent move was available for classification.

H4, verification. H4 states that the probability that a generation is followed by a repair increases with its verification time. We report the distributions and median verification times before repair and progression. The analysis included generations with a classified subsequent move and an available verification time.

H5, familiarity. H5 states that the mean specificity of a participant's prompts is higher in the familiar-domain task than in the unfamiliar-domain task. For each participant, we computed mean prompt specificity separately for the two tasks. We compared these paired task means using a one-sided Wilcoxon signed-rank test in the preregistered direction. This test does not require normally distributed differences. The analysis included participants whose familiarity differential was at least two points on the 7-point scale and who received a generated T3 brief without a fallback. Seventy-one of the 100 participants met these criteria.

Preregistered descriptives. Measured collected without specific hypotheses attached, were pre-registered as purely descriptive reports.

Regeneration similarity measures the structural similarity between repeated realizations of the same prompt. Each of 50 sampled prompts was resubmitted five times, using the prototype snapshot that preceded the original generation as the initial state. For each event, we computed one minus the normalized tree edit distance between the generated component trees, averaged over all pairs of committed reruns. This measure was preregistered as regeneration dispersion; higher values indicate greater structural similarity (Appendix C).

Vocabulary overlap measures the semantic similarity between participants' prompt vocabulary and a domain lexicon generated from each brief. For each brief, we asked a language model to generate a 200-word domain lexicon. For each participant and task, we extracted unique content words after removing stopwords and interface terms. *Density* is the mean of each participant word's highest cosine similarity to a lexicon word. *Coverage* is the mean of each lexicon word's highest similarity to a word in the participant's vocabulary. Both measures were computed separately for T1 and T3 (Appendix C).

We also report the distribution of repair types, the relation between changed nodes and repair, participants' mean prompt abstraction in T1 and T3, and the Spearman correlation between abstraction and specificity across coded generations.

Validation of the coding. We assessed coding agreement on 150 sampled generation events, using quadratically weighted Cohen's kappa for abstraction and specificity and unweighted kappa for relation and repair type (Table 1).

Repair type was assessed only for events both coders classified as repair. Appendix B reports comparisons between each coder and the model and documents all uses of language models in Study 1.

Table 1. Coding agreement on 150 sampled events. Parentheses give the number of paired ratings. The final column compares model ratings with human ratings only where both human coders independently agreed on the code being assessed.

Code	Human–human κ (n)	LLM vs agreeing human ratings κ (n)
Abstraction	.82 (129)	.88 (105)
Specificity	.65 (129)	.60 (75)
Relation	.64 (101)	.45 (87)
Repair type	.36 (19)	—

The preregistration required agreement of at least .70 for each code used inferentially, both between human coders and between the model and adjudicated human ratings. Because repair relation did not meet this threshold, H3, H4, and other repair analyses are reported descriptively. Inferential analyses involving specificity were retained despite agreement below the threshold.

Exploratory analyses. We examined preview interaction in the interval between the completion of a committed generation and the prompt initiating the next committed generation in the same task. An interval contained preview interaction when it included at least one click in the preview or navigation between prototype pages, and we report the observed repair proportions with and without preview interaction, overall and within verification-time intervals (Appendix C).

We classified prompts by their attached reference (none, a pointed element, or a sketch) and counted the participants who used each type. The brief-echo analysis classified prompt content words as echoed from the brief, as domain words absent from the brief, or as other words, and it classified a word absent from the brief as a domain word when its highest cosine similarity to a word in the domain lexicon was at least .60.

We examined how the estimates for H1 and H2 changed under six alternative model specifications, including additional covariates, separate fits for each task, and the inclusion of opening generations from a blank canvas (Appendix C). The task-time analysis recorded the countdown at each prompt and whether participants completed the task before the timeout. To qualitatively compare the prototypes produced in the two building tasks, we scored each participant's final T1 and T3 prototypes with two frontier language-model judges.[8] We sampled 50 participants from the first 81 sessions and obtained 49 usable pairs. Each judge received the brief and the prototype's source code and assigned scores on seven criteria and overall (Appendix C).

5 STUDY 1: RESULTS

5.1 Session Characteristics

Across the 100 sessions, participants initiated 1,789 generation attempts during the three tasks. Of these, 76 were aborted and 68 failed, together accounting for 8.0% of attempts. The analysis table contains 1,643 of the 1,645 committed generations; two were excluded because a usable final source snapshot was unavailable. These comprise 509 generations in the unfamiliar-domain task (T1), 611 in the familiar-domain task (T3), and 523 in the modification task (T2), with a

[8] `gpt-5.6-terra` and `claude-opus-5`, prompt `judge-finals-v1`, temperature 0.

median of 5 per task (interquartile range 3 to 7; Figure 5). The 350 attempts preceding the tasks, including those in the tutorial, were excluded from these counts and analyses.

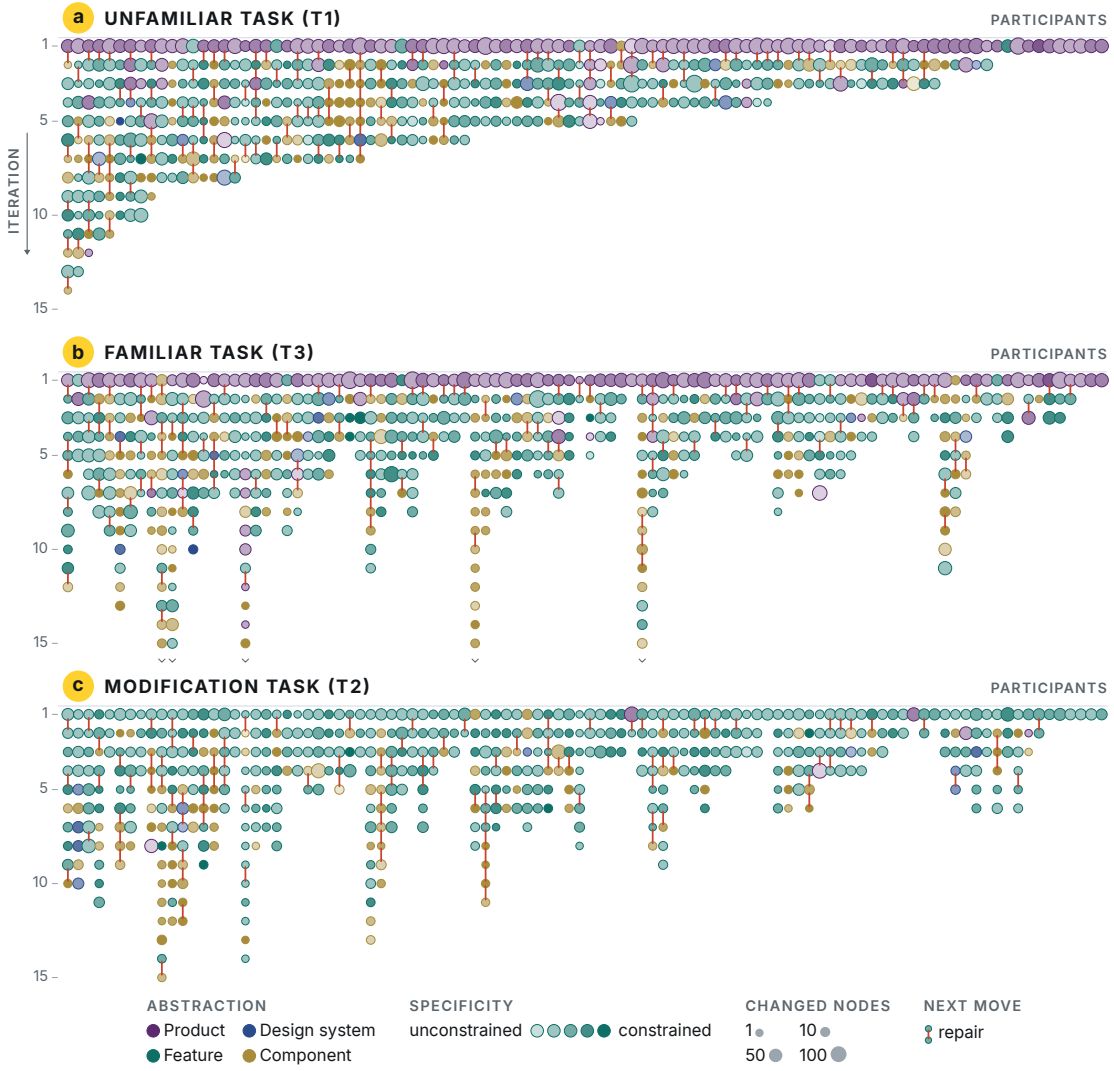


Fig. 5. The 100 sessions of Study 1, with one panel per task (a unfamiliar domain; b familiar domain; c modification). Each column represents one participant, ordered by the number of generations in the unfamiliar-domain task, and each circle one committed generation, ordered vertically by iteration. Hue represents the abstraction of the prompt (Product, Design System, Feature, or Component), tone its specificity, from unconstrained to constrained, and area the number of changed nodes. A red link connects a generation to the repair that followed it. Columns are truncated after the fifteenth iteration, and a chevron marks the tasks that continued beyond it.

From consent to session completion, including generation time, sessions lasted 16 to 79 minutes, with a median of 39 minutes (IQR = 16). Participants completed 225 of the 300 tasks (75%) before the countdown expired, with a median of

4.0 minutes remaining. They submitted 1,790 prompts during the tasks, including one without an associated generation attempt. Of these prompts, 63% were submitted within the first five minutes of a task; the share peaked in the second minute, at 15%, and decreased from the sixth (Figure 6 a). The opening prompt contained a median of 12 content words and the prompts from the second iteration onward, where iteration is the position of a generation event within its task, a median of 5 to 6 (Figure 6 b). Fifty-seven of the 100 participants attached at least one pointed element or sketch to a prompt, and the 25 participants with the most references accounted for 75% of the 354 references (Figure 6 c).

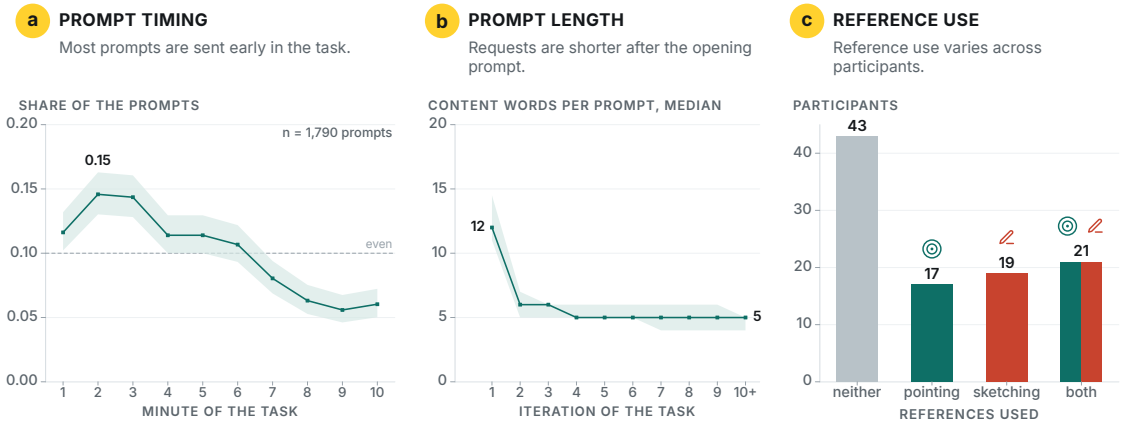


Fig. 6. Prompt timing, prompt length, and reference use across the 100 sessions, with the three tasks pooled. Panel a gives the proportion of all prompts submitted in each minute of a task, with 95% Wilson intervals. Panel b gives the median number of content words per prompt at each iteration, with 95% bootstrap intervals and iterations 10 and later pooled in the last point. Panel c counts the participants by the type of reference they used across the three tasks, neither, pointing only, sketching only, or both.

Feature was the most frequent level of the prompts, 59%, against 24% Component, 15% Product, and 2% Design System, and specificity levels 1 to 5 accounted for 2%, 39%, 33%, 18%, and 8% of the prompts. The opening prompt on the blank canvas of T1 and T3 was Product in 186 of the 200 tasks (Figure 7 a), whereas in the modification task it was Feature in 96% of the tasks, and Product prompts remained rare throughout the task (Figure 7 b). From the second iteration onward, Feature and Component prompts accounted for 93% of the generations. Abstraction and specificity were correlated, with a Spearman ρ of .46 over the coded generations, so that more concrete prompts were also more specific.

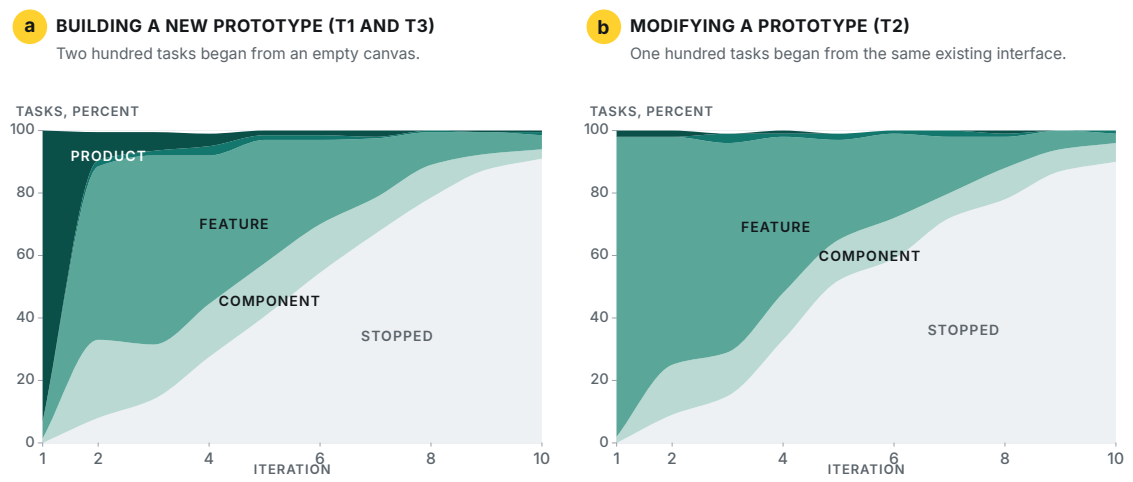


Fig. 7. The abstraction of the prompts at each iteration, for the two blank-page tasks pooled (a) and the modification task (b), as a percentage of the task instances. Iteration is the position of a generation event within its task. Abstraction runs from Product at the top to Design System, Feature, and Component, and the tasks already completed are shown in grey at the bottom.

5.2 Formulation and the Extent of the Change (RQ1)

RQ1 asks how prompt formulation relates to the structural extent of generated change. H1 predicts more changed nodes for more abstract prompts, and H2 fewer changed nodes for more specific prompts. Both associations were in the preregistered directions and remained significant after Holm correction (Table 2). In the negative-binomial mixed model, each level from Product toward Component was associated with a 42% lower expected number of changed nodes (Figure 8 H1), and each point of specificity with a 32% lower expected number (Figure 8 H2), holding the other measure constant.

Both associations remained consistent in the robustness analysis. Under the six alternative specifications, the abstraction coefficient ranged from −0.61 to −0.49 and the specificity coefficient from −0.40 to −0.34, with every 95% confidence interval excluding zero (Appendix C).

The 50 sampled events had a median regeneration similarity of 0.975. Event-level mean distances had a median of 0.025 both between reruns and between reruns and the generation originally received by the participant. These distances decreased with prompt specificity, with exploratory Spearman correlations of $r_s(48) = -.23$, $p = .103$, and $r_s(48) = -.40$, $p = .004$, respectively (Appendix C).

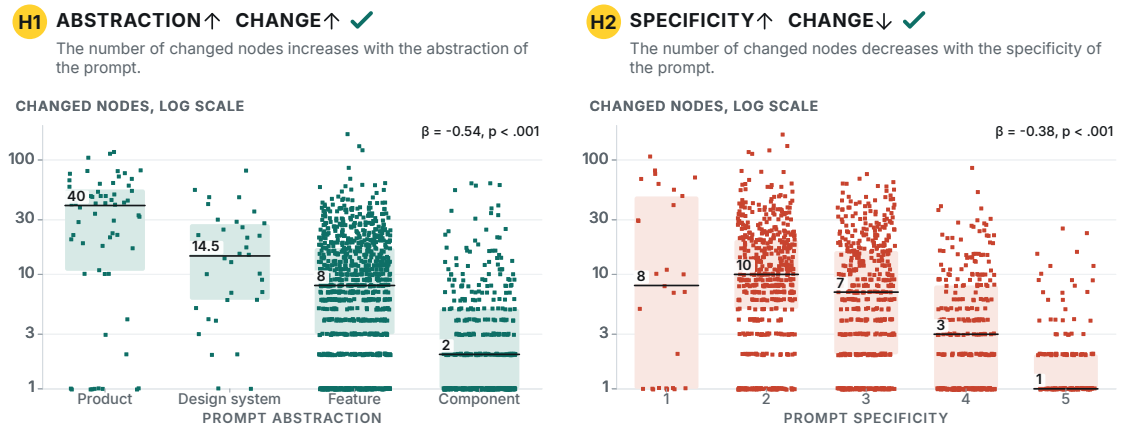


Fig. 8. The two preregistered tests on the extent of the change, H1 and H2. Each panel plots the number of changed nodes on a log scale against the abstraction (H1) or the specificity (H2) of the prompt, with a labelled bar at each group median and a box for the quartiles. The arrows in the titles give the direction each hypothesis expects, and each panel reports the coefficient of its preregistered model. The markers indicate whether the tests met the Holm-adjusted significance threshold; the displayed p-values are unadjusted and one-sided.

Table 2. The three retained inferential tests. H1 and H2 estimates are model coefficients on the log count scale, with standard errors in parentheses. For H5, the estimate is the mean within-participant difference in specificity (T3 minus T1), with the Wilcoxon statistic V in parentheses. Both unadjusted and Holm-adjusted one-sided p-values are reported; the correction retains a family size of five. H3 and H4 are reported descriptively in Subsection 5.3.

Test	Model and predictor	Estimate	p	Holm p	n	Conclusion
H1	negative-binomial mixed model of changed nodes; abstraction	−0.54 (0.05)	$< .001$	$< .001$	1,437 generations	supported
H2	the same model; specificity	−0.38 (0.04)	$< .001$	$< .001$	1,437 generations	supported
H5	paired Wilcoxon signed-rank test; mean specificity, T3 minus T1	0.03 (V = 1,096)	.355	1.00	71 participants	not supported

5.3 Inspection, Progression, and Repair (RQ2)

RQ2 asks how participants verify generated changes and decide whether to progress or repair. H3 concerns repair in relation to prompt specificity, and H4 concerns repair in relation to verification time.

Using the model classifications and logged undo actions, repair followed 26% of generations with a classified subsequent move (350 of 1,340). Across specificity levels 1 to 5, the observed repair proportions were 37%, 26%, 28%, 24%, and 20%, respectively (Figure 9 H3). Median verification time was 19 seconds before repair and 21 seconds before progression (Figure 9 H4).

Preview interaction was measured in the 1,343 gaps between consecutive committed generations of a task (97 participants). A preview interaction was recorded in 45% of the gaps (95% Wilson interval 43 to 48%), in similar proportions across abstraction levels and tasks (Appendix C). Among the 1,338 gaps with a classified subsequent move, repair followed 29% of those with preview interaction (178 of 604; 95% confidence interval 26 to 33%) and 23% of those

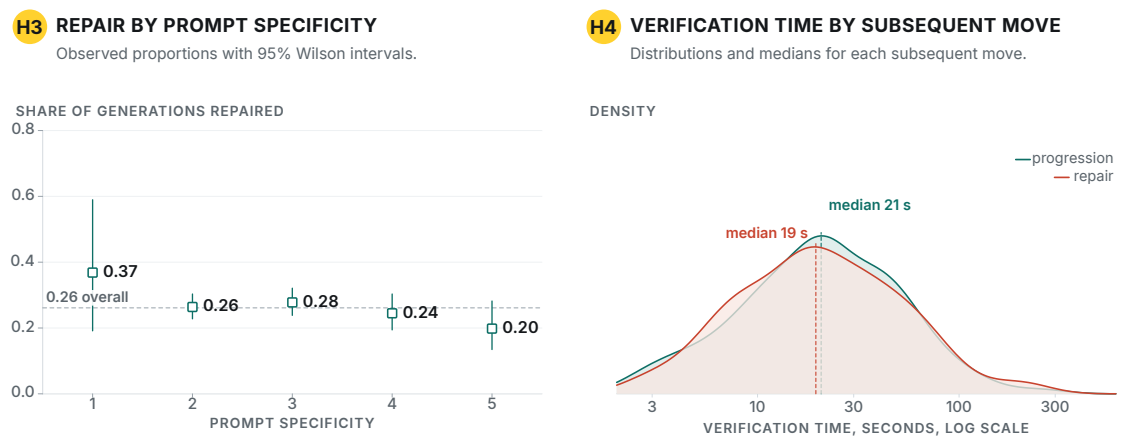


Fig. 9. Descriptive comparisons corresponding to H3 and H4. Panel H3 shows the proportion of generations followed by repair at each specificity level, with 95% Wilson intervals and the overall proportion as a dashed line. Panel H4 shows verification-time distributions on a log scale before progression and repair, with their medians.

without it (170 of 734; 20 to 26%). The observed repair proportion was higher with preview interaction in each of the four verification-time intervals shown in Figure 10 c.

A repair followed 38% of generations after a Product prompt, 36% after a Design System prompt, 23% after a Feature prompt, and 24% after a Component prompt (Figure 10 a).

The model classified 43 repairs as restate and 242 as refine; a further 65 repairs were identified from undo actions (Figure 10 b; Appendix C). The generations followed by a repair had changed a median of 11 nodes, compared with 7.5 for those followed by a progression, and the committed generation that followed a repaired one (for restate and refine, the generation produced by the repair prompt) changed a median of 4 nodes.

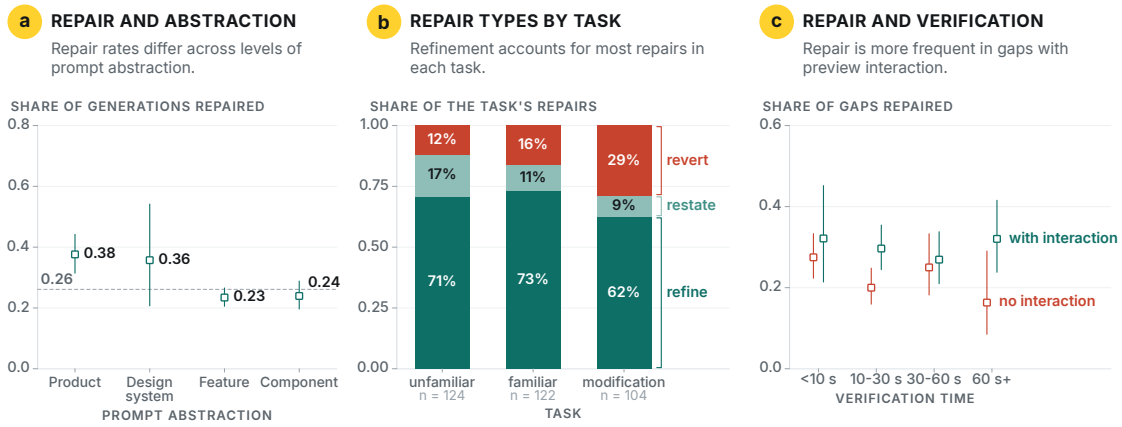


Fig. 10. Repair classifications across the 100 sessions, using model codes for progression, restate, and refine, and logged undo actions for revert. Panel a gives the proportion of generations followed by a repair by the abstraction of the prompt, with 95% Wilson intervals and the overall proportion as a dashed line. Panel b shows the distribution of restate, refine, and revert within each task. Panel c gives the proportion of gaps followed by a repair in four verification-time intervals (under 10 s, 10 to 30 s, 30 to 60 s, and 60 s or more), separately for the gaps with and without a preview interaction, a click in the preview or a navigation to another page, with 95% Wilson intervals.

5.4 Domain Familiarity (RQ3)

RQ3 asks how domain familiarity and design training shape prompt formulation, verification, and repair. Study 1 addresses domain familiarity; design training is examined in Study 2. H5 predicts higher mean prompt specificity in the familiar-domain task and was tested on the 71 participants who met the preregistered eligibility criteria. Mean specificity was 0.03 points higher in T3 than in T1, and H5 was not supported (Holm-adjusted $p = 1.00$; Table 2; Figure 11 H5). Across all 100 participants, mean prompt abstraction was 2.49 in T1 and 2.63 in T3 on the scale from 1 (Product) to 4 (Component).

Among the 71 participants included in H5, median vocabulary density was .61 in T1 and .60 in T3, and median coverage was .45 and .46 (Appendix C). The exploratory brief-echo analysis showed differences in the origin of prompt words. The median share of content words echoed from the brief was 47% in T1 and 27% in T3; this share was higher in T3 for 16 of the 71 participants (Figure 11 +). The median number of domain words absent from the brief was 0.6 per prompt in T1 and 1.0 in T3, with higher values in T3 for 44 participants. In the familiar-domain task, prompts echoed the brief less and contained more domain words absent from it; these comparisons were exploratory and descriptive.

The separate assessment of final prototypes covered 49 participants sampled from the first 81 sessions (Section 4.6). Mean overall scores, averaged across the two language-model judges, were 5.6 in T1 and 5.5 in T3. The judges' overall scores were correlated across the 98 prototypes (Spearman $r_s(96) = .91$, $p < .001$).

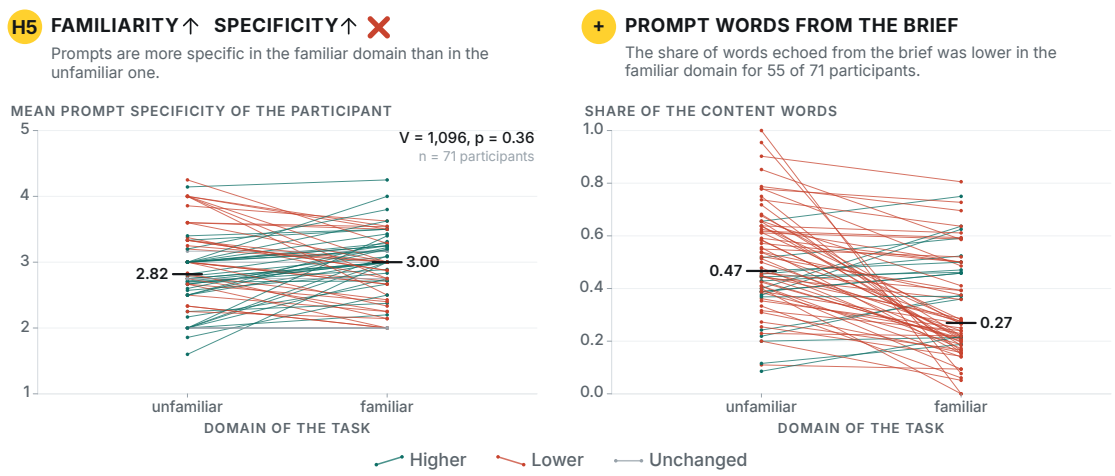


Fig. 11. The preregistered test on domain familiarity, H5, and the exploratory vocabulary finding, for the 71 participants of the H5 test. Panel H5 shows the mean specificity of the prompts of each participant in the unfamiliar and in the familiar domain, joined by a line, with the two medians and the statistic of the registered Wilcoxon test. In both panels, teal indicates a higher value in the familiar domain, terracotta a lower value, and grey no change. The arrows in the H5 title give the direction the hypothesis expects. The marker beside that title indicates whether the test met the Holm-adjusted significance threshold; the displayed p-value is unadjusted and one-sided. The second panel plots in the same way the share of content words that the brief also uses.

6 STUDY 2: REPLAY INTERVIEWS

In Study 1, we quantified how prompt formulation, generated change, and subsequent moves relate. These measures do not reveal why participants formulated a given request, what they expected from the system, or why they continued, revised, or stopped. In Study 2, we therefore conducted semi-structured interviews with non-experts and professional designers after they had completed the same tasks.

6.1 Study Design

Through replay of selected moments, referred to as anchors, we explored participants' expectations, verification of generated results, and subsequent decisions in relation to their own work. By revisiting requests and prototype states, participants could explain how the results compared with their expectations, what they had examined, and how they had decided what to do next.

Because participants had produced different interfaces, we also presented everyone with the same example, a "vignette" consisting of a request and before-and-after screenshots of a gym interface. By asking participants to evaluate the same changes and propose a subsequent request, we compared their evaluation criteria and the reasons for their proposed actions.

At the end of the interview, we asked participants to reflect on the session as a whole, including whether they had considered starting over and what would make them confident in the tool's results.

6.2 Participants

We recruited 20 participants through Prolific, university networks, and professional networks: 8 non-experts and 12 professional designers. The non-experts were aged 26 to 54 years; three reported being women and five being men.

The designers were aged 24 to 40 years; eight reported being women, three being men, and one did not report gender. Table 3 reports their occupations, familiar-task activities, and frequency of AI-tool use. Participant identifiers use **P** for non-experts and **D** for designers, numbered within each group in session order.

Designers' reported backgrounds included product and digital design, UX engineering, and HCI research. The familiar-domain task drew on the activity or subject each participant reported knowing most deeply. For some designers this was design itself; for others it concerned activities such as sailing, photography, or volunteering. Domain familiarity and design experience therefore overlap in some cases. Compensation was GBP 25 for Prolific participants, paid as GBP 0.60 for the booking submission and a GBP 24.40 bonus after the session, and a gift of their choice worth up to DKK 300 (about EUR 40) for participants recruited through university and professional networks.

6.3 Procedure and Interview Materials

Sessions took place over a video call, with participants sharing their screen and one author moderating. After consenting to participation and recording, participants completed the survey, tutorial, and tasks described in Section 4.3. During the tasks, the moderator assisted with technical difficulties and clarified instructions, task terminology, or completion procedures where needed.

Replay interviews. After the tasks, we selected replay anchors from each session log: the generation event with the most changed nodes, the first event followed by repair, and the final prototype of the last task. We also included an event near the median changed-node count with no subsequent repair and randomly sampled one additional event from those not already selected.

During replay, we revisited the prompt to discuss what participants had wanted to achieve and what they had expected to be generated. We replayed their interactions with the generated prototype and asked which aspects they had inspected and whether the result matched their request. Questions such as "Was there anything here you didn't ask for?" extended this comparison to unrequested changes.

We connected these evaluations to subsequent decisions by discussing what participants requested next and asking, for example, "Did seeing this change what you wanted to build?" When participants had initiated repair, we explored the problem they had identified and the alternatives they had considered. When discussing the final prototype, we asked why participants had ended the task and what they would have done next without the timer.

The moderator referred to the recording, prompts, and prototype states and adapted the order and wording of questions to participants' responses. Questions asked participants to describe what they had intended, expected, and done before the moderator proposed any reading of a result, and follow-up questions pursued what participants had raised themselves, sometimes revisiting adjacent events or returning to earlier topics. The selection criteria and complete interview guide are provided in Appendix D.

Shared vignette. The vignette was a fabricated generation event presented as originating from another participant's session. It showed a neighborhood-gym interface before and after the request "make the sign up button stand out and show how many spots are left for each class" (Figure 12). The after state contained more prominent sign-up buttons and remaining-capacity labels, an unrequested "Become a member" button, and an alteration to the weekend program: Saturday's "Open gym" entry was removed and Sunday's "Family circuit" entry duplicated. The interface continued to render normally.

We asked participants to compare the two screens, describe the changes they noticed, and assess whether the result fulfilled the request. To understand how participants would act on their assessment, we asked "What would you do next?

Say it exactly as you would type it." We discussed their reasons for retaining, revising, or further developing particular aspects of the interface. Follow-up questions and returns to the before state varied according to the conversation.

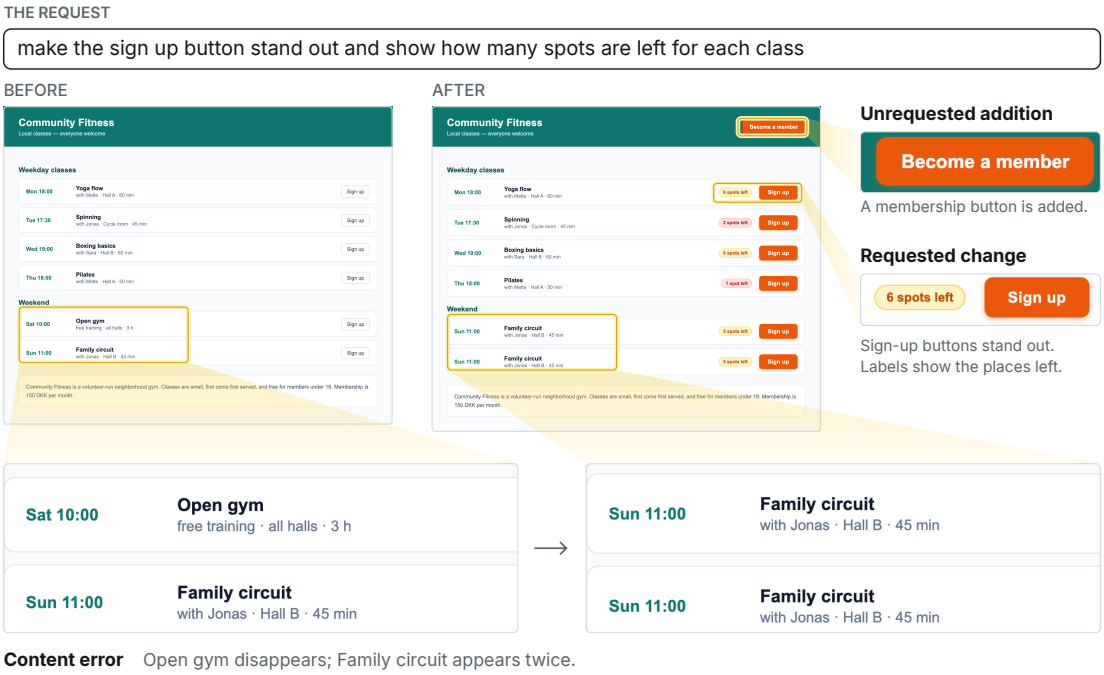


Fig. 12. The shared vignette and its three kinds of change: the requested emphasis and capacity information, an unrequested membership action, and an alteration to the weekend content. Linked enlargements show the affected regions. The screenshots show the study stimulus with the gym name and location replaced; annotations and enlargements were added for the reader and were not shown to participants.

6.4 Analysis

We analyzed the interviews following the stages described by Kvale and Brinkmann [16]. The recordings were transcribed automatically,[9] and every passage quoted in this paper was checked against the recording. We read each transcript in full, divided it into passages that each addressed one moment of the session, and restated the theme of each passage from the participant's point of view, a procedure Kvale and Brinkmann describe as meaning condensation. The condensed passages were then grouped across participants into categories that arose from the material and corresponded to the three questions of the study design, that is, what participants had intended with a request, what they had expected from its realization, and how they had decided what to do next. The results report these categories with excerpts, and the discussion interprets them in relation to the measures of Study 1 and to prior accounts of design practice, which Kvale and Brinkmann distinguish from condensation as meaning interpretation.

[9]Transcripts were generated with `gpt-4o-transcribe-diarize`.

7 STUDY 2: RESULTS

In the replay interviews, participants explained what they had intended with a request, what they had expected the system to generate from it, and how they had decided to continue, revise, or stop after examining the result. We report their explanations in this order and relate them to their knowledge of the activity and of interaction design.

7.1 From Intentions to Requests

During replay, we asked participants what they had intended to achieve with their prompts and how they had decided what to request. Some of the prompts they discussed formulated a modification to the interface or its functionality, while others formulated the goal, that is, what the prototype should support, and left the realization to the system.

In the familiar task, whose brief asked for a tool to support volunteer coordination, **P4** requested a monthly calendar in addition to the daily view the prototype already provided, stating in the prompt that this would allow them to allocate volunteers. During replay, they explained that volunteers *"really need to plan in advance"* in order to rearrange their other commitments, so that a coordinator needs to see the whole month. In their familiar task, a tool for planning trips, **D10** requested that the permanently visible form for adding trips be accessed through a button, explaining in the same prompt that they wanted more attention on the schedule. During replay, they described this explanation as *"trying to maybe give my reason"*. Both participants had already decided how the interface should change, and their requests contained that decision together with the goal that motivated it.

In the unfamiliar task, whose brief asked for a grain mill scheduler, **P6** listed the milling runs, the information to record for each run, and the operations of adding, editing, and deleting entries, a prompt they summarized as *"build me this tool, here's the criteria"* and *"see what you come up with"*, in contrast with instructions about where each element should appear. **D8**, whose unfamiliar task asked for a tool to manage a warehouse, copied its brief into their first prompt as *"the best specification I could give for the starting point"*. Both prompts stated what the prototype had to support and left the arrangement of the page and the choice of controls to the system.

The form of a request varied with the task and with the participant's knowledge of the activity. In their familiar task, a tracker for their own poker sessions, P6 formulated modifications that followed their own routine of recording each session, entering the amount won or lost, and reviewing the results, whereas for the unfamiliar task their request was limited to the criteria in the brief. D8 likewise could not complete the unfamiliar brief without knowing more about warehouse work, an account consistent with the greater reuse of the brief's words in the unfamiliar domain observed in Study 1 (Subsection 5.4). Once the prototype existed, however, D8 examined it *"from the usability point of view, not so much from the domain point of view"* and, *"imagining how the people work in this domain"*, requested predefined storage-bay values because free-text entry could introduce typing errors, and the recording of incoming items by selecting existing entries because re-entering them would duplicate work. **D5** similarly proposed changing the orange membership button of the shared gym vignette because they questioned whether it provided sufficient contrast against the dark-green background. Knowledge of the activity enabled P6 to formulate modifications, while knowledge of interaction design provided D8 and D5 with another basis for proposing changes.

Describing a modification required effort even when the participant had decided what to change. **D5**, whose usual design process consisted of sketching the space occupied by each element and adjusting the arrangement, tried to imagine the layout before describing it in words, *"sketching in my brain and then try to translate it into text"*. Because they found detailed prompting tedious, by the third task they generated one part of the interface first and used the elements already visible as a reference when describing further additions.

In participants' explanations, the choice between the two formulations followed from their knowledge of the activity or of interaction design and from the effort of describing a realization. This knowledge provided a ground for deciding how the interface should change, while participants who had difficulty identifying or describing a modification left more of its realization to the system.

7.2 Expectations of Generated Results

After viewing the generated result, we asked participants whether it corresponded to what they had expected from their request.

In the familiar trip-planning task, **D10** deliberately left the position of the button for adding trips unspecified, explaining that they had given the system *"some freedom"*. They had decided which control to introduce but were willing to consider the system's choice of placement. **D8** expected the system to contribute the requirements they could not identify themselves. In the unfamiliar warehouse task, they expected the system to supplement the brief, reasoning that the model would know *"the domain better than me, since it's like some large language model that can dig through a bunch of stuff and actually give me something that's reasonable"*. During replay, however, they noted that the system *"did not add additional reasonable requirements"* and had followed the supplied instructions. D8 therefore evaluated whether the system had contributed to completing the requirements, in addition to whether it had implemented those already written.

Participants also held expectations they considered implicit in the request and therefore did not state. In the familiar poker task, **P6** expected a requested graph to display time on the horizontal axis and monetary values on the vertical axis. When reviewing the graph without these values, they explained that *"I thought that might automatically happen"* and suggested that they might need to request them explicitly. **D5** similarly reported having to request section titles during refinement, explaining that *"in my mind it's a given that a section needs a title"*. These participants regarded the omitted details as part of the function or component they had requested. Their expectations derived from familiar conventions that they assumed the system would follow without separate instructions.

Expectations could also depend on the interface already available. Although **D10** left the button's position unspecified, they expected the existing form for adding trips to remain within the page when shown or hidden, since they had not requested a dialog overlay. Their explanation, *"I assume it has the context"*, referred to the existing prototype as the basis for interpreting the modification. They expected the system to preserve the form's presentation without a further description when only the way of accessing it changed.

Participants described both the contribution they expected from the system and the properties they expected it to include or preserve without further instructions. These expectations could draw on familiar conventions, the existing prototype, or knowledge they attributed to the model.

7.3 Verification and Subsequent Decisions

During replay, we asked participants what they had examined after generation and how they had decided what to do next. For the final prototype, we also asked about their reasons for stopping and what they would have done with more time. Their accounts connected judgments about the result to what they still wanted to understand, change, or develop.

In the shared gym vignette (Figure 12), the request was to make class sign-up buttons more prominent and show the number of places left in each class. The after-state also included an unrequested "Become a member" button, which 12 participants explicitly identified as new. **P2** saw a use for this addition and proposed a registration page with membership benefits and prices. **D1** questioned which action users should focus on, since the membership and

class sign-up buttons had the same colour, and proposed integrating membership into class registration. **D8** instead objected that membership had not been requested, calling its addition *"a mistake for me"*. D8 justified rejecting the addition by referring to the request, while P2 and D1 considered its place in the service and the interface and treated the unrequested change as material to develop.

Similarly, in their unfamiliar task, a tool for managing harbour berths, **D6** had expected an Airbnb-style map of occupied berths without specifying that arrangement, but the generated interface showed two columns of numbered berths. They considered this plausible because boats would be placed one after another, and explained that they *"worked on top of that"*. Having assessed how the arrangement could represent the activity, D6 adopted it as the basis for further development. Examining the result therefore gave them a reason to reconsider the solution they had initially imagined.

Participants also described uncertainty about whether a result was adequate or whether further changes would be useful. **P3** reported having probably *"blindly trusted"* the unfamiliar auction prototype because they did not know what to look for, whereas the familiar chess prototype *"needed to match"* functionality they already knew. In the auction task, however, P3 noticed that the pictures did not match the descriptions of the lots and requested a separate page displaying the available images. They wanted to investigate whether the problem lay in the available pictures or in how they were matched to particular lots (*"That was mainly for debugging."*). The next request sought information needed to understand the mismatch, so that verification continued through another generation.

Asked what they would do with another hour, **P6** proposed chart filters and session notes for the familiar poker tracker to support reviewing their play, whereas for the unfamiliar grain mill scheduler they struggled to identify useful additions. Familiarity with the activity gave P6 a basis for judging which additional functions would be useful. For the mill, they relied on the requirements supplied in the brief.

Decisions to continue or stop also reflected the objectives participants wanted to pursue and the difficulties they encountered. After judging the unfamiliar warehouse prototype functionally adequate, **D4** continued adjusting its layout to make it more compact and visually balanced (*"functionally, it was enough. It was just me adjusting the visual preference and visual layout"*). **P5** reported satisfaction with the auction prototype before submitting a hammer drawing to see whether the tool could transform it into an auction graphic, following disappointment with an earlier sketch. With these further requests, D4 pursued visual refinement and P5 explored the system after they considered the prototype adequate. Conversely, **P2** stopped working on the familiar hiking prototype without the dates they wanted to display because attempts to add them produced errors. P2 could identify the modification they wanted, but their decision to finish reflected the difficulty of obtaining it.

Across the three parts of the exchange, participants described intentions and expectations that went beyond what they wrote in their requests. Their knowledge of the activity and of interaction design helped them identify and judge requirements, while the effort of describing a realization affected what they specified in advance. They could leave some choices open to the system while expecting other properties to be included or preserved without further instructions. After examining the generated result, they adopted a choice, reconsidered what they wanted to build, or investigated a problem through another generation. Decisions about what the prototype should become were therefore made both before generation and while examining the system's realization. Designers and non-experts differed in the references available to them for these decisions, drawn from interaction design for the former and from the activity for both, and not in the form of the exchange itself.

8 DISCUSSION

The discussion relates the generation events measured in Study 1 to the intentions and judgments participants reported in Study 2. We examine the knowledge on which users ground a design decision, the decisions they defer to the system, and the way they reconsider their intentions throughout the session.

8.1 Grounding Design Decisions

Study 1 did not support the hypothesis that domain familiarity would produce more specific prompts. The interviews indicate how familiarity nevertheless helped participants identify requirements and judge their usefulness, including when they left aspects of the realization unspecified.

Knowledge of the activity allowed participants to draw on their own practices to identify requirements beyond those in the supplied brief. Prior work on requirements elicitation describes how domain knowledge affects the understanding and communication of stakeholder needs [11]. In our sessions, the same knowledge connected formulation to verification, since the reasons for requesting a function also provided grounds for judging whether its realization supported the activity and which further changes would be useful (Subsection 7.3). In the exploratory analysis of Study 1, prompts in the unfamiliar domain reused more of the brief's words (Subsection 5.4). The interviews help interpret this finding, as participants unfamiliar with the activity described relying on the brief to formulate a request and expecting the system to supply requirements they could not identify themselves (Subsection 7.2). Expecting this contribution did not, by itself, resolve their uncertainty about whether the prototype adequately supported the activity. An interface could satisfy the written request while leaving open what else the activity required.

Knowledge of interaction design provides criteria for evaluating interface choices that users have left to the system. Prior work describes the interface representations and professional judgment that designers apply to generated material [19, 27]. In generative prototyping, this knowledge can guide development without being fully expressed in the prompt. Criteria not stated during formulation can remain part of what users expect and become explicit when verification identifies a mismatch. Examining the realization can also identify interaction problems that provide reasons for requesting changes. Design expertise therefore contributes to deciding which generated choices to retain or revise, in addition to specifying an interface before generation. Such judgments can support revisions in an unfamiliar domain while leaving open whether the functions themselves meet the needs of the activity.

Expertise can therefore support delegation even when users have not specified the choices they later assess. A generated interface makes those choices available for verification, but it does not necessarily provide the knowledge needed to judge them. The ability to obtain a realization can consequently exceed the ability to determine which of its choices are appropriate and worth developing.

8.2 Deferring Design Decisions

In Study 1, more specific requests were associated with smaller structural changes (Subsection 5.2). Study 2 shows why the constraints expressed in a request do not fully describe the expectations against which users judge the result. Participants were willing to consider different realizations, but they also expected particular properties without requesting them explicitly. The absence of such a property motivated repair even when the system had followed the written instructions. We distinguish four overlapping reasons for leaving aspects of a realization unspecified, with different implications for what users request next.

Delegation based on partial knowledge occurs when users leave details unspecified because they lack the knowledge to determine what is needed. They provide the available requirements and rely on the system to supply what they cannot identify themselves, as D8 expected when using the unfamiliar warehouse brief (Subsection 7.2). Generation then proceeds despite gaps in the requirements, and obtaining a realization does not resolve those gaps, because the user may still be unable to identify which functions are missing or which additions would be useful. What was unspecified before generation can therefore remain unresolved afterwards.

The absence of detail can also express a willingness to consider the system's proposal. *Intentional openness* can accompany a goal formulated without prescribing its realization, and also a request for a particular modification. D10, for example, requested a button for accessing the trip form while deliberately leaving its position open (Subsection 7.2). The user has specified the change they want and is willing to assess how the system realizes one aspect of it. A subsequent request can then express a preference formed after seeing the result.

Alongside choices deliberately left open, users may hold *implicit expectations* about properties they consider part of the request or assume will be preserved from the existing prototype. Although D10 left the button's position open, they expected the trip form to remain within the page, and D5 considered section titles necessary without initially requesting them (Subsection 7.2). When such a property is omitted or changed, repair requires expressing an expectation that existed before generation. The omission reflects what the user considered already understood, and it carries no willingness to consider alternative realizations of that property.

Even when users know what the prototype should support, the *effort of formulation* can make it difficult to specify its realization in advance. D5's shorter requests allowed them to describe additions in relation to elements already visible, in place of a complete layout composed before generation (Subsection 7.1). The difficulty lay in imagining and describing the arrangement. Subramonyam et al. discuss the balance between articulating intentions in advance and developing them through iteration [31]. In our setting, generation provided a reference for subsequent formulation and allowed users to describe the interface progressively.

What users need to specify next therefore depends on both the choices still open and the expectations they had assumed the system would satisfy. Examining a generated result can help them resolve an open choice or recognize that an expected property requires an explicit instruction. In the latter case, repair clarifies the request while leaving the intended design unchanged. Formulation and evaluation are connected through this examination of what needs to be communicated, which is consistent with the metacognitive demands described by Tankelevitch et al. [34]. Support for formulation could therefore help users identify which properties they expect to be preserved, which choices they want to develop from the result, and which requirements they cannot yet determine. Encouraging greater specificity in the initial prompt addresses only part of this work, because some details become easier to formulate once a realization exists, while others require further knowledge of the activity.

8.3 Reflection-in-Realization

In Schön's account of reflection-in-action, designers evaluate the intended and unintended consequences of their moves and reconsider intentions through their work with intermediate artifacts [30]. In generative prototyping, users delegate implementation to a system and then examine how their requests have been realized, including choices they did not explicitly specify. Verification becomes part of reflection-in-action when users reconsider their requirements, expectations, or intended workflow while examining what the system has implemented.

We describe the reconsideration of design intentions while examining a generated prototype as *reflection-in-realization*. While verifying what has been generated, they can revise both what they want the interface to become and what they

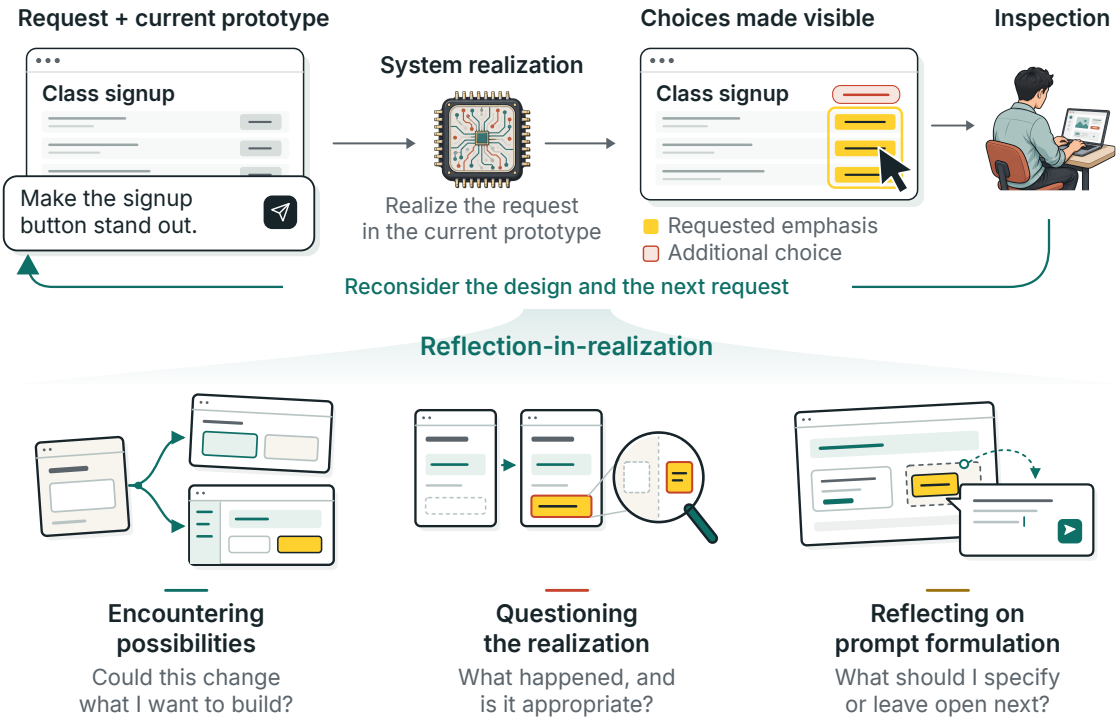


Fig. 13. Three overlapping occasions for reflection-in-realization: encountering possibilities, questioning the realization, and reflecting on prompt formulation. In the upper schematic, the system modifies the current prototype in response to a request, with the requested emphasis in yellow and an additional change in terracotta. While inspecting the result, participants may reconsider both the design and the next request. The lower illustrations show the three occasions through questions participants may pursue within or across exchanges; these are not sequential stages. All interfaces are schematic.

need to specify next. We distinguish three overlapping occasions according to what users reconsider: encountering possibilities, questioning the realization, and reflecting on prompt formulation (Figure 13). For example, users may recognize value in an unrequested function, reconsider how the workflow should be organized, and reformulate their request to develop this revised design.

Encountering possibilities. Serendipitous discovery can occur when users recognize a use for an unexpected layout or an unrequested function and reconsider what they want to build. This recognition may lead to adopting a different representation during a task or proposing further development when discussing a result. Deliberately leaving a choice open is one route to encountering alternatives, but users can also revise an idea they had already formed after judging a different realization useful. To develop that possibility, they need to relate it to the activity and decide how it would fit the interface. Domain knowledge and design experience are therefore relevant to turning an unexpected result into a direction for further work.

Questioning the realization. To decide whether to retain a generated element, users may examine how it relates to the request and what it implies for use. An unrequested element can be worth developing, and an element that satisfies the request can still require revision, so establishing what the system changed is part of deciding whether the change

is appropriate. We regard verification as reflective when users reconsider a requirement, an expectation, or a design decision while carrying out this examination. Examining an added function, for example, users may reconsider how the workflow should be organized, in addition to assessing whether the function fits the workflow they had initially intended.

When users cannot account for a mismatch, they may seek further information through generation itself. P3 requested a page of lot images to establish whether the pictures available to the application were unsuitable or whether the application matched them to the wrong lots, so that the request produced a view through which to investigate the mismatch (Section 7.3). A request can also investigate the tool, as when P5 submitted a sketch after reporting satisfaction with the prototype in order to see what the system would produce from it. In both cases the participant generated something in order to answer a question about the result or about the tool. Verification can therefore continue through a further generation, so that users reflect on how to obtain the information they need, in addition to the design itself.

Reflecting on prompt formulation. Examining a generated prototype can also lead users to reconsider how they formulate a request. When a result omits a property that users were expecting, they recognize that the system does not supply it without instruction and that a later request has to state it. When a result makes part of the interface visible, it becomes a reference for describing what to add, so that users can place further elements in relation to those already generated without composing the whole arrangement in advance. Users then revise their assumptions about what the system infers, what they have to state, and what they can leave open until a realization exists.

8.4 Limitations

Our analysis took the generation event as its unit and the session as its boundary. Both studies used one generative prototyping tool and short tasks with a countdown, so that the tool and the available time may have shaped participants' requests and their decisions to stop, alongside their design intentions. Each participant worked in a single session, and we cannot establish how these practices develop when a user returns to a prototype over several sessions or works in another environment.

For the Study 1 analysis, we retained the inferences involving specificity although agreement for that variable was .65 between the human coders and .60 between the model and the agreeing human ratings, below the preregistered threshold of .70. We are working to refine the coding definitions and instructions, and the revised coding will require validation on independent cases and a rerun of the affected analyses.

The replay interviews provided retrospective accounts of participants' decisions, and participants may have developed explanations while reviewing their sessions that they had not held during the tasks. The replay anchored these accounts in the prompts and prototype states of the session itself, so that participants explained their decisions with the material in front of them. We use the accounts as interpretations of a recorded exchange, not to attribute a difference between the groups to design training.

9 CONCLUSION

This paper dissected generative prototyping into the parts of the generation event, the formulation of a request, the change the system generates, the verification that follows, and the next move. Across 100 instrumented sessions, the abstraction and the specificity of a request were associated with the structural extent of the generated change. In replay interviews, participants explained how they decided what to specify, what they expected the system to realize without instruction, and how they judged a realization before continuing, revising, or stopping.

Participants drew on their knowledge of the activity and of interaction design to formulate requirements and evaluate realizations. Delegating realization leaves some choices open and exposes others that users expect without stating them. Examining the generated prototype can therefore lead users to reconsider both what they want to build and what they need to communicate to the system. Their knowledge provides grounds for directing this process. Obtaining a realization does not itself establish how to judge its adequacy, and that judgment depends on what users know about the activity. Generative prototyping tools can support this work by helping users express expectations, assess generated choices, and develop their intentions across successive realizations. Mark, the data, and the analysis code are released without restriction, so that the anatomy can be reproduced, extended, and applied to other environments.

REFERENCES

[1] Saeema Ahmed, Ken M. Wallace, and Luciënne T. Blessing. 2003. Understanding the Differences Between How Novice and Experienced Designers Approach Design Tasks. *Research in Engineering Design* 14, 1 (2003), 1–11. https://doi.org/10.1007/s00163-002-0023-z

[2] Jon-Tait Beason, Ruijia Cheng, Eldon Schoop, and Jeffrey Nichols. 2026. Athena: Intermediate Representations for Iterative Scaffolded App Generation with an LLM. In *Proceedings of the 31st International Conference on Intelligent User Interfaces*. Association for Computing Machinery, New York, NY, USA, 284–306. https://doi.org/10.1145/3742413.3789133

[3] Tua A. Björklund. 2013. Initial Mental Representations of Design Problems: Differences Between Experts and Novices. *Design Studies* 34, 2 (2013), 135–160. https://doi.org/10.1016/j.destud.2012.08.005

[4] Bill Buxton. 2007. *Sketching User Experiences: Getting the Design Right and the Right Design*. Morgan Kaufmann, San Francisco, CA, USA. https://doi.org/10.1016/B978-0-12-374037-3.X5043-3

[5] 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*. Association for Computing Machinery, New York, NY, USA, Article 686, 20 pages. https://doi.org/10.1145/3706598.3713285

[6] Philip Cash, Milene Gonçalves, and Kees Dorst. 2023. Method in Their Madness: Explaining How Designers Think and Act through the Cognitive Co-evolution Model. *Design Studies* 88, Article 101219 (2023), 32 pages. https://doi.org/10.1016/j.destud.2023.101219

[7] 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*. Association for Computing Machinery, New York, NY, USA, 1179–1196. https://doi.org/10.1145/3715336.3735780

[8] Nigel Cross. 2004. Expertise in Design: An Overview. *Design Studies* 25, 5 (2004), 427–441. https://doi.org/10.1016/j.destud.2004.06.002

[9] Peter Dalsgaard. 2017. Instruments of Inquiry: Understanding the Nature and Role of Tools in Design. *International Journal of Design* 11, 1 (2017), 21–33. https://www.ijdesign.org/index.php/IJDesign/article/view/2275

[10] Kees Dorst and Nigel Cross. 2001. Creativity in the Design Process: Co-evolution of Problem–Solution. *Design Studies* 22, 5 (2001), 425–437. https://doi.org/10.1016/S0142-694X(01)00009-6

[11] Irit Hadar, Pnina Soffer, and Keren Kenzi. 2014. The Role of Domain Knowledge in Requirements Elicitation via Interviews: An Exploratory Study. *Requirements Engineering* 19, 2 (2014), 143–159. https://doi.org/10.1007/s00766-012-0163-2

[12] Stephanie Houde and Charles Hill. 1997. What Do Prototypes Prototype? In *Handbook of Human-Computer Interaction* (2 ed.), Martin G. Helander, Thomas K. Landauer, and Prasad V. Prabhu (Eds.). North-Holland, Amsterdam, The Netherlands, Chapter 16, 367–381. https://doi.org/10.1016/B978-044481862-1.50082-0

[13] Edwin L. Hutchins, James D. Hollan, and Donald A. Norman. 1985. Direct Manipulation Interfaces. *Human-Computer Interaction* 1, 4 (1985), 311–338. https://doi.org/10.1207/S15327051HCI0104_2

[14] Ellen Jiang, Kristen Olson, Edwin Toh, Alejandra Molina, Aaron Donsbach, Michael Terry, and Carrie J. Cai. 2022. PromptMaker: Prompt-Based Prototyping with Large Language Models. In *CHI Conference on Human Factors in Computing Systems Extended Abstracts*. Association for Computing Machinery, New York, NY, USA, Article 35, 8 pages. https://doi.org/10.1145/3491101.3503564

[15] Chinmay Kulkarni, Stefania Druga, Minsuk Chang, Alex Fiannaca, Carrie J. Cai, and Michael Terry. 2023. A Word is Worth a Thousand Pictures: Prompts as AI Design Material. https://doi.org/10.48550/arXiv.2303.12647 arXiv:2303.12647 [cs.HC]

[16] Steinar Kvale and Svend Brinkmann. 2009. *InterViews: Learning the Craft of Qualitative Research Interviewing* (2 ed.). SAGE, Thousand Oaks, CA.

[17] Kyungho Lee. 2025. Towards a Working Definition of Designing Generative User Interfaces. In *Companion Publication of the 2025 ACM Designing Interactive Systems Conference*. Association for Computing Machinery, New York, NY, USA, 489–495. https://doi.org/10.1145/3715668.3736365

[18] Alan Leung, Ruijia Cheng, Jason Wu, Jeffrey Nichols, and Titus Barik. 2025. SQUIRE: Interactive UI Authoring via Slot QUery Intermediate REpresentations. In *Proceedings of the 38th Annual ACM Symposium on User Interface Software and Technology*. Association for Computing Machinery, New York, NY, USA, Article 199, 17 pages. https://doi.org/10.1145/3746059.3747672

[19] Jie Li, Hancheng Cao, Laura Lin, Youyang Hou, Ruihao Zhu, and Abdallah El Ali. 2024. User Experience Design Professionals' Perceptions of Generative Artificial Intelligence. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*. Association for Computing

Machinery, New York, NY, USA, Article 381, 18 pages. https://doi.org/10.1145/3613904.3642114

[20] Youn-Kyung Lim, Erik Stolterman, and Josh Tenenberg. 2008. The Anatomy of Prototypes: Prototypes as Filters, Prototypes as Manifestations of Design Ideas. *ACM Transactions on Computer-Human Interaction* 15, 2, Article 7 (2008), 27 pages. https://doi.org/10.1145/1375761.1375762

[21] Michael Xieyang Liu, Advait Sarkar, Carina Negreanu, Benjamin Zorn, Jack Williams, Neil Toronto, and Andrew D. Gordon. 2023. "What It Wants Me To Say": Bridging the Abstraction Gap Between End-User Programmers and Code-Generating Large Language Models. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*. Association for Computing Machinery, New York, NY, USA, Article 598, 31 pages. https://doi.org/10.1145/3544548.3580817

[22] Kumiyo Nakakoji, Yasuhiro Yamamoto, Takahiro Suzuki, Shingo Takada, and Mark D. Gross. 1998. From Critiquing to Representational Talkback: Computer Support for Revealing Features in Design. *Knowledge-Based Systems* 11, 7–8 (1998), 457–468. https://doi.org/10.1016/S0950-7051(98)00079-3

[23] Jakob Nielsen. 1993. Iterative User-Interface Design. *Computer* 26, 11 (1993), 32–41. https://doi.org/10.1109/2.241424

[24] Donald A. Norman. 1986. Cognitive Engineering. In *User Centered System Design: New Perspectives on Human–Computer Interaction*, Donald A. Norman and Stephen W. Draper (Eds.). Lawrence Erlbaum Associates, Hillsdale, NJ, USA, 31–61.

[25] Seokhyeon Park, Soohyun Lee, Eugene Choi, Hyunwoo Kim, Minkyu Kweon, Yumin Song, and Jinwook Seo. 2026. Bridging Gulfs in UI Generation through Semantic Guidance. In *Proceedings of the 2026 CHI Conference on Human Factors in Computing Systems*. Association for Computing Machinery, New York, NY, USA, Article 880, 16 pages. https://doi.org/10.1145/3772318.3791966

[26] Xiaohan Peng, Debanjana Haldar, Wendy E. Mackay, and Janin Koch. 2026. DesignTrace: Exploring, Iterating and Tracking Design Alternatives with GenAI. In *Proceedings of the 2026 CHI Conference on Human Factors in Computing Systems*. Association for Computing Machinery, New York, NY, USA, Article 434, 22 pages. https://doi.org/10.1145/3772318.3791036

[27] Savvas Petridis, Michael Terry, and Carrie J. Cai. 2024. PromptInfuser: How Tightly Coupling AI and UI Design Impacts Designers' Workflows. In *Designing Interactive Systems Conference*. Association for Computing Machinery, New York, NY, USA, 743–756. https://doi.org/10.1145/3643834.3661613

[28] Laria Reynolds and Kyle McDonell. 2021. Prompt Programming for Large Language Models: Beyond the Few-Shot Paradigm. In *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems*. Association for Computing Machinery, New York, NY, USA, 1–7. https://doi.org/10.1145/3411763.3451760

[29] Donald A. Schön. 1983. *The Reflective Practitioner: How Professionals Think in Action.* Basic Books, New York, NY, USA.

[30] Donald A. Schön. 1992. Designing as Reflective Conversation with the Materials of a Design Situation. *Research in Engineering Design* 3, 3 (1992), 131–147. https://doi.org/10.1007/BF01580516

[31] Hari Subramonyam, Roy Pea, Christopher Lawrence Pondoc, Maneesh Agrawala, and Colleen Seifert. 2024. Bridging the Gulf of Envisioning: Cognitive Challenges in Prompt Based Interactions with LLMs. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*. Association for Computing Machinery, New York, NY, USA, Article 1039, 19 pages. https://doi.org/10.1145/3613904.3642754

[32] Hari Subramonyam, Divy Thakkar, Andrew Ku, Jürgen Dieber, and Anoop K. Sinha. 2025. Prototyping with Prompts: Emerging Approaches and Challenges in Generative AI Design for Collaborative Software Teams. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*. Association for Computing Machinery, New York, NY, USA, Article 882, 22 pages. https://doi.org/10.1145/3706598.3713166

[33] Macy Takaffoli, Sijia Li, and Ville Mäkelä. 2024. Generative AI in User Experience Design and Research: How Do UX Practitioners, Teams, and Companies Use GenAI in Industry?. In *Designing Interactive Systems Conference*. Association for Computing Machinery, New York, NY, USA, 1579–1593. https://doi.org/10.1145/3643834.3660720

[34] Lev Tankelevitch, Viktor Kewenig, Auste Simkute, Ava Elizabeth Scott, Advait Sarkar, Abigail Sellen, and Sean Rintel. 2024. The Metacognitive Demands and Opportunities of Generative AI. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*. Association for Computing Machinery, New York, NY, USA, Article 680, 24 pages. https://doi.org/10.1145/3613904.3642902

[35] Severi Uusitalo, Antti Salovaara, Tero Jokela, and Marja Salmimaa. 2024. "Clay to Play With": Generative AI Tools in UX and Industrial Design Practice. In *Designing Interactive Systems Conference*. Association for Computing Machinery, New York, NY, USA, 1566–1578. https://doi.org/10.1145/3643834.3661624

[36] J.D. Zamfirescu-Pereira, Richmond Y. Wong, Bjoern Hartmann, and Qian Yang. 2023. Why Johnny Can't Prompt: How Non-AI Experts Try (and Fail) to Design LLM Prompts. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*. Association for Computing Machinery, New York, NY, USA, Article 437, 21 pages. https://doi.org/10.1145/3544548.3581388

A PARTICIPANTS AND STUDY INSTRUMENTS

A.1 Participant Profiles

The Study 1 sample comprised 100 participants: 49 women, 49 men, and two who did not report gender. Age was available for 99 participants (median 35 years, range 18–73). AI-tool use was daily for 48 participants, a few times a week for 33, a few times a month for 16, and a few times ever for three.

Table 3 reports the backgrounds and familiar-task activities of the eight non-experts and twelve designers in Study 2.

Table 3. Backgrounds of the 20 Study 2 participants. Age, gender, occupation, and AI-use frequency were self-reported. The familiar-task activity identifies the domain used for each participant's generated familiar task. Occupations have been shortened for readability.

ID	Age	Gender	Occupation	Familiar-task activity	AI-use frequency
Non-experts					
P1	44	Man	Technical services manager	Scottish football	Daily
P2	51	Man	Creditors officer	Hiking trips	Several times a week
P3	26	Man	Junior construction consultant	Chess	Several times a week
P4	45	Woman	Accountant	Event volunteering	Several times a week
P5	31	Man	Ghost kitchen owner	Cookie baking	Daily
P6	50	Man	Self-employed	Poker	Daily
P7	32	Woman	Software developer	Personal fitness	Several times a week
P8	54	Woman	Welfare rights adviser	Dog showing	Several times a week
Designers					
D1	40	Not reported	Digital designer	Music and film history	Several times a week
D2	32	Man	Lead UX/UI designer	Home baking	Several times a week
D3	30	Woman	Principal product designer	Kitesurfing	Daily
D4	28	Man	PhD researcher	Industrial design research	Daily
D5	30	Woman	HCI postdoctoral researcher	Conversational interaction	Several times a week
D6	28	Woman	Product manager	Football coaching	Daily
D7	24	Woman	UX/UI design intern	Youth volunteering	Daily
D8	35	Man	UX design engineer	Bouldering	Daily
D9	29	Woman	HCI PhD student	Design work	Daily
D10	33	Woman	Product designer	Sailing	Daily
D11	29	Woman	UX specialist	User experience design	Several times a week
D12	28	Woman	User interface designer	Toddler parenting	Daily

AI-use frequency concerns AI chat tools. "Several times a week" represents the questionnaire response "A few times a week." "Not reported" indicates that no gender was supplied.

A.2 Survey Items and Consent

The consent screen specified the data recorded by the study (prompts, prototype snapshots, and interaction telemetry), the voluntary nature of participation, and the data-storage procedure. Prolific IDs were stored as keyed hashes.

Table 4. The three instruments of Study 1, with the wording shown to the participant.

Instrument / item	Wording as shown	Response format
Pre-study survey		
Occupation	"What is your occupation? Be specific."	free text
Activity or subject	"What activity or subject do you know most deeply?", with the note "Inside or outside work: a serious hobby, volunteering, anything you really know."	free text
AI-tool use	"How often do you use AI tools such as ChatGPT?"	Never / A few times ever / A few times a month / A few times a week / Daily
Age	"Age"	whole number
Gender	"Gender"	Woman / Man / Non-binary / Prefer not to say
Familiarity check Five ratings	"How familiar are these to you?", with the note "Rate how much you know about each of these domains.", one row for each of the four candidate T1 domains, in a fixed order, and one for the generated T3 domain, inserted at a position that varied by participant	7-point scale, 1 = "Never heard of it", 7 = "Know it very well"
Domain knowledge Four statements	"I am familiar with this kind of work"; "I know the everyday vocabulary people use in this domain"; "I could describe a typical day of the person in the brief"; "I have personal or professional experience close to this domain"	7-point scale, anchored at "not at all true of me" and "very true of me"

B GENERATION AND CODING PROCEDURES

B.1 Generation Attempts and Briefs

The tool validates generated source code by compiling the files and loading the page in the preview, and returns validation errors to the model for up to three rounds of revision. An attempt is classified as failed when the source does not pass validation after these rounds or the model call returns an error, and as aborted when the participant interrupts it or the connection closes during generation. For the brief of T3, the participant's occupation was used in place of the activity or subject they knew most deeply when that response contained fewer than four characters. A deterministic validator evaluated every generated brief. A brief that failed validation was regenerated once at temperature 0.8; if the second brief also failed, the system provided a fixed fallback brief concerning everyday task tracking. The fallback brief was used in two of the 100 sessions.

B.2 Coding Pipeline and Language Models

The coding pipeline processed each committed generation in three passes (Section 4.6). Table 5 gives the question each code answers, the input the model received, and the output, and Table 6 every use of a language model in Study 1.

Table 5. Codes assigned by the coding pipeline, the corresponding coding question, the model input, and the output.

Code	Coding question	Model input	Output
Abstraction	at which semantic level does the prompt operate	the prompt, with its task key and a flag for the first generation on the blank canvas	one of Product, Design System, Feature, Component
Specificity	how much does the request constrain its realization	the prompt alone, with its reference tokens	1 to 5
Requests	which distinct requests does the prompt contain	the prompt, with its task key and a flag for the first generation on the blank canvas	the list of distinct requests, each quoted from the prompt
Repair relation and type	does the next prompt return to this one, and how	the requests, the structural diff summarized as a list of added, modified, and removed elements with their visible text, and the following prompt	repair, progression, or none, the repaired request, and the repair type

Table 6. Language-model components used in Study 1, with their specifications and validation procedures.

Component	Function	Specification	Validation
T3 brief generation	fills the nine domain slots	`gpt-5.1-2025-11-13`, prompt `t3-gen-v1.4` archived verbatim, temperature 0	deterministic validator on every brief; pilot criterion of a maximum 15 percent fallback rate
Domain lexicon	expands each brief into a 200-word domain lexicon for the vocabulary overlap	`gpt-5.6-luna`, lexicon prompt in the analysis scripts	preregistered descriptive (2), reported without a test
Coding pipeline	abstraction, specificity, requests, repair relation and type	`gpt-5.6-luna`, the deposited prompts for pass 1, specificity coding, and pass 3	200 events sampled; 129 of the first 150 rated by both coders for abstraction and specificity; .70 threshold, with an exception for specificity (Section 4.6)
Final-prototype judges	score each final prototype against its brief on seven criteria and overall	`gpt-5.6-terra` and `claude-opus-5`, prompt `judge-finals-v1`, temperature 0, blinded to task and domain	exploratory; agreement between the two judges reported

Table 7. Coding agreement for the first 150 sampled events. Values are kappa, with the number of paired ratings in parentheses. The final column compares model ratings with human ratings on events where the two human coders independently agreed on the code.

Code	Human–human	Coder 1–LLM	Coder 2–LLM	LLM vs agreeing human ratings
Abstraction	.82 (129)	.77 (129)	.80 (129)	.88 (105)
Specificity	.65 (129)	.54 (129)	.36 (129)	.60 (75)
Relation	.64 (101)	.41 (103)	.41 (104)	.45 (87)
Repair type	.36 (19)	.38 (17)	—	—

Abstraction and specificity use quadratically weighted Cohen's kappa; relation and repair type use unweighted Cohen's kappa. Repair-type comparisons include events that both human coders classified as repair. The Coder 1–LLM repair-type comparison additionally requires a model type rating; it does not require the human coders to agree on repair type. Dashes denote comparisons not available in the current assessment.

C SUPPLEMENTARY ANALYSES

C.1 Robustness Specifications

The robustness analysis refitted the changed-node model for H1 and H2 under six alternative specifications: task order as a covariate; iteration (the position of a generation event within its task), task, and task order as covariates; the preregistered model fitted separately within T1, T3, and T2; and all generations included with an indicator for opening generations from the blank canvas. Figure 14 reports the abstraction and specificity coefficients for the registered model and each alternative specification.

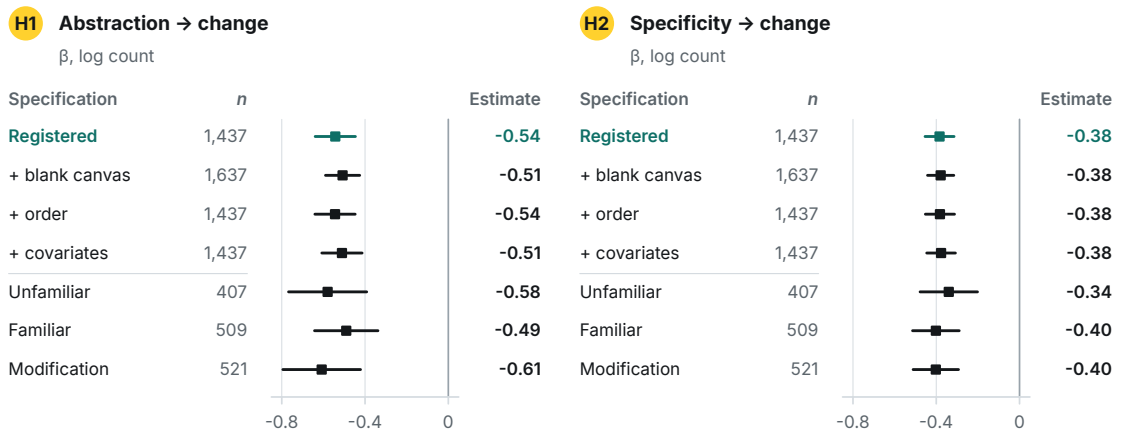


Fig. 14. Robustness of H1 and H2 across model specifications. Squares show abstraction and specificity coefficients in the changed-node model with 95 percent intervals; teal marks the registered model.

C.2 Preview Interaction

Of the 3,358 clicks recorded in the 1,343 gaps, 64 percent targeted interactive controls (buttons, inputs, selects, links, or labels), and navigation to another page occurred in 8 percent of gaps. The proportions of gaps with a preview interaction

were 42, 46, 48, and 40 percent following Product, Design System, Feature, and Component prompts, respectively (n = 218, 28, 763, and 329), and 47, 48, and 41 percent in T1, T3, and T2 (n = 409, 511, and 423).

C.3 Repair Types by Task

Table 8 reports the repair types within each task. Restate and refine were classified by the coding pipeline; revert was identified from undo actions.

Table 8. Repair types by task. Percentages are calculated within each task.

Task	Repairs	Restate (%)	Refine (%)	Revert (%)
Unfamiliar (T1)	124	17	71	12
Familiar (T3)	122	11	73	16
Modification (T2)	104	9	62	29

C.4 Regeneration Similarity

The 50 generation events were sampled at random, ten per specificity level, from the 1,050 committed generations of the first 81 sessions with a specificity code, a non-empty prompt, both snapshots available, and no attached pointed-element or sketch reference. Each rerun underwent the same source-validation procedure as the study's generations and received the preceding prompts from the task as its only context. The component trees are labelled by tag alone, so textual content does not contribute to the distance measure.

The structural similarity between committed reruns of each event and their distance to the generation the participant received are given by specificity in Figure 15.

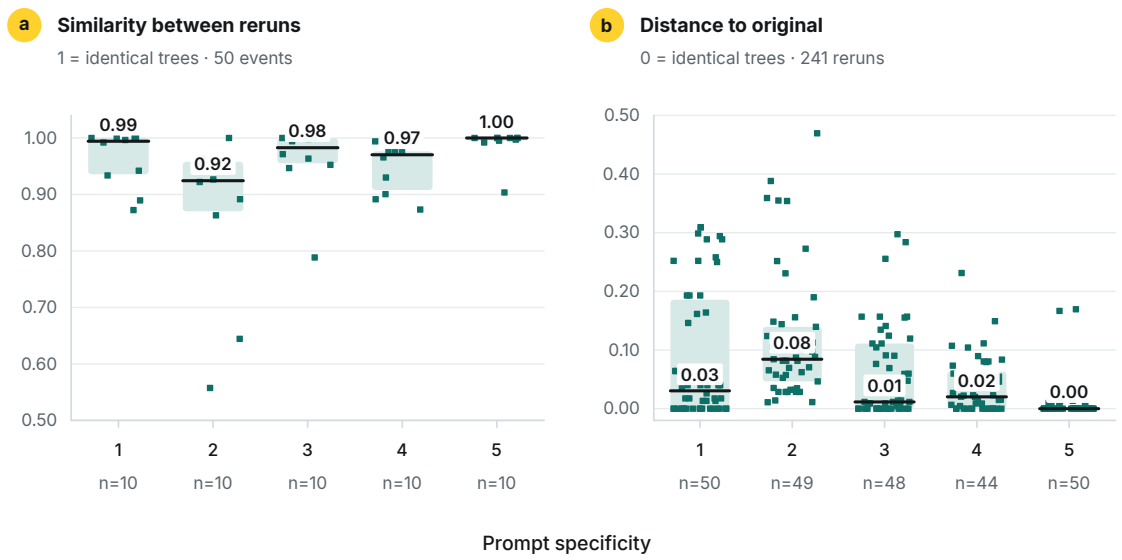


Fig. 15. Regeneration by prompt specificity. Panel a shows each of the 50 events' mean similarity across pairs of committed reruns, measured as one minus normalized tree edit distance. Panel b shows the distance of each of the 241 committed reruns from the original generation. Boxes span the quartiles; black lines and labels mark medians.

C.5 Vocabulary Overlap

A language model (`gpt-5.6-luna`) expanded each brief into a 200-word domain lexicon. Each unique content word in a participant's prompts, after removing stopwords and terms denoting parts of the prototype, was scored by its highest cosine similarity to the lexicon under the word-embedding model all-MiniLM-L6-v2.

Among the 71 participants included in the H5 analysis, participants used a median of 34 distinct words in T1 and 40 in T3. Because coverage increases with the number of distinct words, an exploratory recomputation sampled 20 distinct words per participant and task and averaged coverage across 400 samples; median coverage was then .415 in T1 and .415 in T3 (n = 82 of the 100 participants, those with at least 20 distinct words in both tasks).

C.6 Final-Prototype Evaluation

Each judge scored each prototype from 1 to 5 on seven criteria (each of the brief's three needs, domain fit, usability, visual quality, and code soundness) and from 1 to 10 overall.

Of the seven criteria, only domain fit differed between the two tasks, by 0.36 points in favour of T3 over the 49 pairs of prototypes; on the 35 pairs from qualified participants the difference was 0.10 points.

D REPLAY INTERVIEW MATERIALS

D.1 Anchor Selection

The Anchor Sheet linked selected generation events to their prompts, prototype states, and positions in the session recording. Table 9 summarizes the selection and the topic of each replay anchor. The moderator used these entries flexibly: not all were discussed, and more than one entry could refer to the same event.

Table 9. Replay anchors, selected moments, and interview topics. A1–A5 refer to events in the participant's session; V and D refer to the shared vignette and debrief.

Anchor	Selected moment	Interview topic
A1	Largest structural change across the three tasks	Intentions and expectations
A2	First generation followed by repair; one per task for designers	Problem identified and alternatives considered
A3	Final prototype of the last task	Stopping, further development, and authorship
A4	Event nearest the median changed-node count, with no subsequent repair	Verification and subsequent decisions
A5	Random event from those remaining	Verification and subsequent decisions
V	Shared gym interface before and after generation	Noticing changes and proposing the next request
D	Session as a whole	Restarting, future use, and trust

Structural change was measured by the number of changed nodes in a committed generation. For A2, repair meant a subsequent undo, a near-identical resubmission, or a prompt classified as repair by the coding pipeline. Opening generations in blank-canvas tasks, A1, and the final generation reserved for A3 were excluded. Near-identical prompts had a Ratcliff/Obershelp similarity ratio of at least .85 after lowercasing and removing bracketed reference tags; the ratio measures the proportion of characters in matching text blocks. A5 used the fixed random seed 20260818.

After the third task, the script recorded the selected events, repair relations, and coding-pipeline status in the Anchor Sheet. The moderator could use the sheet to move between the session recording and the corresponding questions.

D.2 Interview Guide

The interview guide and event-selection procedure were revised during the early sessions; the guide reproduced below reflects the later version. Questions are grouped by anchor, with additional questions for designers after the shared questions. Order, wording, and coverage varied during the interviews.

A1: Intentions and expectations

"Take me back to this request: what were you trying to get?" "What did you expect the tool would actually do with this?" "Was there anything here you didn't ask for?"

For designers: "Where does this layout come from, in your view?"

A2: Repair and alternatives

"What was wrong here, in your words?" "Why do you think it came out like this?" "What options did you feel you had at that point?"

For designers: on a plausible generation, "How would you have flagged this in a design review?"; on a broken one, "What did this failure tell you about the tool?"

A3: Stopping and further development

"What told you this was done?" "If the timer hadn't been there, what would you have done next?" "How much of this feels like yours?"

For designers: "Would you show this to a client as a starting point? What would you fix first?"

A4, A5: Verification and subsequent decisions

"Did this do what you asked?" "What did you do between this landing and your next prompt?" "Did seeing this change what you wanted to build?"

For designers: "What would you check before trusting this screen?"

V: Shared vignette

"Talk me through what you see. What changed between the two screens?" "Anything else that changed?" "What would you do next? Say it exactly as you would type it." "What did the tool do here, in your judgment?"

D: Debrief

"At any point, did you consider starting over from scratch? What made you stay, or restart?" "Outside this study, would you use a tool like this? For what?" "What would it take for you to trust what it gives you, without checking everything yourself?" and, as the last question of the debrief, "Is there anything you expected me to ask and I did not?"

For designers: "If a tool like this entered your design work, what would it change in how you work?" "Did anything in the tool conflict with how you normally work?"