A Feasibility Study on AI Agents Navigating Statistical Software

When the Lobster Takes the Pufferfish's Hand

LI Kanyu

26.03.2026

A Feasibility Study on AI Agents Navigating Statistical Software

When the Lobster Takes the Pufferfish's Hand

LI Kanyu

TL; DR

Previously, we described Jamovi as the pufferfish of statistical software: its interface is elegant, yet its inadequate user guidance mechanisms make it easier to make mistakes. This study explores whether the Claws can hold the pufferfish's hand: we assigned the same analytical task to an AI Agent to complete autonomously, testing whether current technology can reliably support the entire research workflow, from data cleaning to result interpretation.

The answer is that, for now, Claws cannot successfully operate Jamovi's interface, but we find that achieving this future is not out of reach.

We first tested whether LLMs could complete data analysis using Computer Use technology. Among six LLM systems, only GPT-5.4-xHigh completed the task and produced correct conclusions. Next, assuming that LLMs could correctly perform the analysis, we tested whether they could interpret and evaluate result interface screenshots. We found that under default conditions, most models produced erroneous data interpretations (Except Claude Haiku). However, after adding two lines of explanatory text to the screenshots, the accuracy of all models except Claude Haiku surged to near-zero error.

1 / 7	28 / 50	42 / 50
Models Completed Computer Use Analysis Task	Result Correctly Explained without Visual Hint	Result Correctly Explained with Visual Hint

The conclusion is, LLMs are not yet capable of supporting unsupervised scenarios. But several approaches could be introduced to improve the performance of LLMs, like incorporating context descriptions that do not automatically disappear (such as error messages or feature prompts), setting necessary elements as defaults to reduce the number of controls an Agent must click, embedding standardized statistical interpretation cues in hypothesis testing output areas, adhering common design patterns, and increasing the size of interactive controls. These interventions come at a very low marginal cost and are equally effective for human users and AI Agents alike.

Drafted Mar 25, 2026

Contents

Correspondence to
LI Kanyu
im@not.ci

Competing Interests
The authors declare no competing interests.

1. Research Objectives

1.a. Core Research Questions

Under current technological conditions, can AI Agents reliably and autonomously control statistical analysis software to complete an entire data analysis workflow? If an Agent successfully manipulates the software, is its ability to interpret statistical results sufficient to support correct inferences by researchers?

1.b. Sub-questions

- What are the respective failure modes of different technical approaches (screenshot, DOM, local models) for Computer Use when applied to real-world GUI analysis software?
- What are the similarities and differences between the failure modes of AI Agents and those observed in human users during the first two rounds of research?
- When an AI is presented with statistical results, how does the quality of contextual information provided by the interface affect its interpretation accuracy?
- For UX designers of GUI software, what design principles need to be reconsidered in the AX era?

1.c. Research Background

Before beginning our tests, it is necessary to provide some context on the industry landscape at the time of this research. In early 2026, an open-source project named OpenClaw surged to prominence on GitHub. Within just 60 days, its star count surpassed that of React, a framework Facebook spent a decade building, making it one of the most-starred software projects in GitHub's history. Following this, various "lobster variants" began to emerge from both the open-source community and commercial companies. The "lobster" boom brought a concept once discussed only in developer circles into the public consciousness: the Computer Use Agent (CUA).

In simple terms, a CUA gives AI "hands" and "feet." Unlike chatbots that can only "talk," a CUA can "see" the screen, move the cursor, and click buttons just like a human, directly operating any software on a computer without needing an API or special plugins. This leap from "able to talk" to "able to do" transforms the AI from a "consultant" into a "digital employee."

As of this writing, the industry has rapidly entered a "lobsterization" race, with products rushing to provide operational interfaces for these agents. Cloud vendors view this contest as a strategic battle to secure the foundational distribution rights for the next generation of AI applications, launching "one-click deployment" services in an attempt to lock users deep within their ecosystems.

The effect triggered by OpenClaw is evolving into a "Lobster War" over the evolution of AI Agents. However, behind the public craze for "raising lobsters," the conflict between technological maturity and engineering stability has become starkly apparent. We aim to explore whether the current level of ecosystem maturity can support fully automated data analysis based on the Computer Use approach.

The first two rounds of our research (usability testing and a PURE expert evaluation) revealed the core contradiction of Jamovi in human-user scenarios: it is simple to operate but has a very high rate of cognitive errors. Identifying variable types is the

most critical cognitive bottleneck, and a lack of feedback mechanisms causes errors to propagate downstream through the workflow. This study introduces AI Agents as a third type of "user" for the same task and software, aiming to explore the practical boundaries of AI-assisted research and to provide a new analytical dimension for GUI software interface design from an AX perspective.

2. Theoretical Framework

This study is based on an Agent Experience (AX) analysis framework (appended to this report due to length). It deconstructs the interaction between an AI Agent and the external world into three layers: how the user conveys intent to the Agent (the input quality problem), how the Agent interacts with the external world (the output controllability problem), and the Agent's internal state management (the context management problem).

Among these, **context management is the most central and easily overlooked dimension of the AX framework.** An LLM's reasoning quality is highly dependent on what information is included in the current context, how it is presented, and when it is injected. From this perspective, an interface is not merely an interactive medium for human users; it should also be a key channel for delivering context to the AI.

The core hypothesis of this study is that **the interface can act as a proactive context delivery mechanism, imposing more stable and controllable constraints on an LLM's behavior than prompts can.** Traditionally, we rely on Skills (static prompts) or MCPs (function calls) to guide an LLM's behavior, but these methods have structural limitations. Skills cannot dynamically and contextually inject information during a task, while MCPs depend on the LLM "proactively calling" the correct function, which remains an essentially probabilistic behavior. A graphical interface or text interface is different. It can embed statistical diagnostics, quality control warnings, and procedural constraints directly into the LLM's visible area at critical decision points. The timing and location of this information are determined by the designer, not dependent on the LLM's active exploration.

(For the complete theoretical framework, see Appendix.)

3. Research Methodology

3.-.i. *Agent Framework Selection Process:*

Before formal testing, the researchers systematically evaluated mainstream Computer Use automation platforms to identify the most suitable infrastructure for this study's needs.

During testing, nearly all major "Claw" type platforms exhibited severe engineering quality issues, making them unsuitable for reproducible academic experiments. OpenClaw had a string concatenation error in its API Gateway URL, preventing reliable connections to its backend service. In ZeroClaw's Telegram integration, both the "send image" and "send file" functions were broken, yet, inexplicably, the "send screenshot" function worked, a fact whose cause remains unknown and suggests a lack of systematic integration testing between its modules. PicoClaw had a critical logic flaw in its context compression management: as the context length approached the model's

limit, the system would trigger a compression process, but this process itself consumed more context space, causing the context to overflow again before compression could complete, leading to a backend crash during LLM runtime. These are engineering blunders that would not even pass a university undergraduate thesis project, yet they are prevalent in platforms currently marketed as "production-grade." This observation in itself constitutes a noteworthy commentary on the ecosystem.

Ultimately, the researchers selected AstrBot as the experimental infrastructure. The earliest version of AstrBot dates back to 2023, long before the aforementioned platforms appeared. It began as a chat automation framework and had built a complete engineering foundation before the rise of large language models. This early accumulation of engineering maturity gave it a significant advantage in feature completeness and system stability. It was the only platform tested with a fully functional, bug-free management panel, complete with a proper administrator account system, permission management, and plugin lifecycle management. On the AI front, AstrBot's support for Memory, MCP tool calls, and Skill prompts were all fully implemented, requiring no patching of core functionalities by the researchers. More critically, AstrBot supports hooking directly into the chat context via plugins, allowing developers to make automated modifications. This capability was a prerequisite for the context compression mechanism in our study and was not available on any other tested platform.

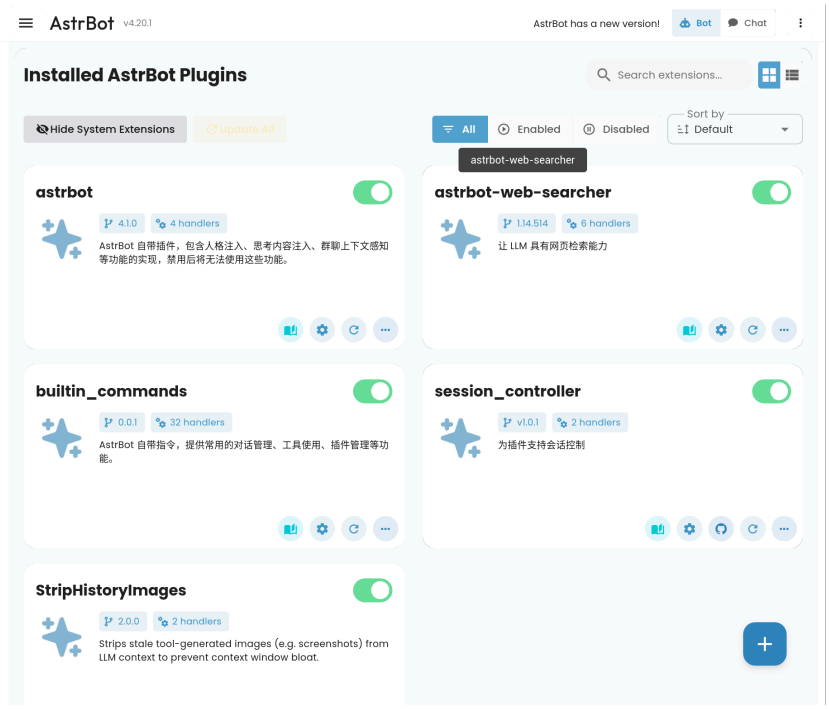


Figure 2: The screenshot of the AstrBot Configuration Panel

3.-.ii. *Computer Use Environment and Context Compression Mechanism:*

The Computer Use environment for this study was built on the following technology stack, designed to maximize reproducibility and complete the experiments with limited local computing power.

Desktop and System Layer: The operating system was NixOS, with all system dependencies declaratively managed via a Nix Flake. The desktop environment was Scroll, a Wayland compositor based on the Sway protocol. NixOS was chosen for its core strength in reproducibility: any environment with the same repository commit hash can be fully restored, including the kernel version, system services, dependency libraries, and runtime configurations, fundamentally eliminating the "it works on my machine" problem.

MCP Service Layer: The researchers developed a custom MCP service for the Scroll compositor (Scroll MCP Server), which communicates with AstrBot via stdio. It exposed four types of tools: window management, screenshot capture, mouse control, and keyboard input. The screenshot capture tool automatically handled HiDPI scaling factors, ensuring a consistent coordinate system across different resolution settings. The mouse control tool had a built-in auto-calibration mechanism that read the current cursor position via `wl-find-cursor` at the start of a session to establish a mapping between logical coordinates and physical pixels. Screenshot data returned by the MCP was embedded as a base64 string in the tool call result and injected by AstrBot into the context for the next turn, serving as the model's visual input for its subsequent reasoning.

Context Compression Mechanism: In this experiment, each MCP screenshot call injects a high-resolution image into the context. Without management, the context would be quickly filled with historical screenshots as the number of operational steps accumulates. The researchers developed a context hook plugin for AstrBot that immediately deletes the screenshot after the MCP return is processed. This mechanism is based on a domain-specific prior for Computer Use tasks: historical frames generally offer no incremental value for the current reasoning step. The model only needs to know "what is the current state of the screen," not a history of every past screenshot. This is a form of lossy compression, but the lost information has a negligible impact on reasoning quality for this task type.

The introduction of this mechanism was crucial for the experiment. Local inference was performed on a workstation equipped with an RTX Pro 6000 running Qwen 3.5 122B (int4 quantized). Even with quantization, the accumulated screenshot data would exhaust the available VRAM after just two operations without the cleanup mechanism, causing the inference to halt. With the cleanup plugin enabled, VRAM usage remained at a stable level, making it possible to complete the full task test with the 122B quantized model.

The experiment also validated the impact of this compression mechanism on commercial API models: GPT-5.4-xHigh completed the full task with screenshot cleanup enabled, and Claude Opus 4.6′s behavior remained stable, with no observed degradation in reasoning quality due to compression.

3.-.iii. *Experiment One: Computer Use Task Completion Rate Test:*

Task Design

Consistent with the previous two rounds of research, the AI Agent was instructed to act as a sleep researcher and perform the following tasks in Jamovi:

- **Data Inspection**: Identify an outlier in the dataset (row 10, value 200) and a format error (row 60, value 3a).
- **Data Cleaning**: Address the two issues identified above.
- **Analysis of Variance**: Perform a one-way ANOVA, setting the correct dependent and grouping variables.
- **Result Reporting**: Provide a research conclusion based on the analysis results.

To eliminate uncertainties related to modal dialogs in the Wayland environment, the dataset was pre-loaded into the Jamovi window before the test began.

Test Subjects

Model	Type	Test Runs	Notes
GPT-5.4-xHigh	Commercial API	1	Full chat log available
Claude Opus 4.6	Commercial API	1	Full chat log available
Claude Sonnet 4.6	Commercial API	1	Full chat log available
Gemini 3.1 Pro Preview Custom Tools	Commercial API	1	No Think Aloud, no chat log
Qwen 3.5 35B	Local Deployment	3	Crashed mid-task, no chat log
Qwen 3.5 122B (int4)	Local Deployment	3	Observation notes available
Page-Agent.js (Alibaba product, DOM-based)	Commercial API	3	Used Jamovi Cloud, observation notes available

All models were given a unified prompt:

“You are playing the role of a sleep researcher investigating how the type of cheese eaten before bedtime affects the frequency of nightmares. Find the Jamovi software on my desktop, which has a dataset loaded. You will perform data inspection and cleaning, conduct an analysis of variance, create a visualization, and report your conclusions to me. All of your operations must be completed within the Jamovi interface; you are not allowed to use external tools.”

Task completion was defined as correctly identifying and handling both data errors, successfully running a one-way ANOVA (with the correct dependent and grouping variables), and providing a research conclusion based on the results.

It is worth noting that due to Wayland environment configuration issues and potential functional interference, this experiment was not video recorded. All subsequent behavioral descriptions are based on the researcher’s real-time observation notes and the “Thinking Aloud” logs left by the LLMs.

Given that our initial attempts revealed an extremely low success rate for LLMs in completing Computer Use tasks, which could not sustain a systematic UX study, we concluded that repeating the experiments multiple times would not yield further insights. As an exploratory study, most models, apart from the local ones, were run only once.

3.-.iv. *Experiment Two: Statistical Result Interpretation Under Unprompted Conditions:*

Ten models were provided with a full screenshot of the Jamovi ANOVA results page (including the Welch's F-test table, descriptive statistics, and a box plot). They were given a unified prompt:

"You are playing the role of a sleep researcher investigating how the type of cheese eaten before bedtime affects the frequency of nightmares. Now that your analysis software has produced the following results, please provide a report interpreting them."

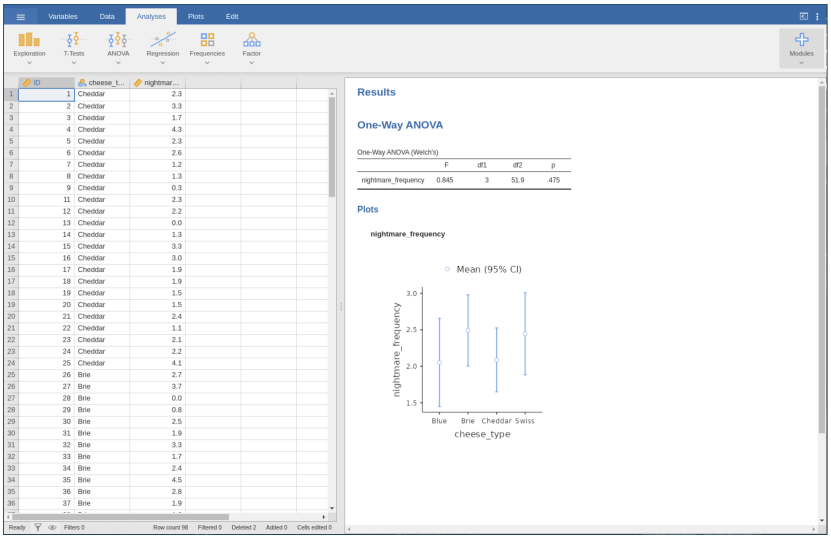


Figure 3: The screenshot being sent to LLM

The researcher manually reviewed all outputs and evaluated them against two gates. Each model was tested five times independently:

- **Gate 1**: Did the model correctly understand that "no evidence is not evidence of no"? It should not make any assertions of no difference, but statements like "the results may be due to random fluctuations" were considered acceptable.
- **Gate 2** (only evaluated if Gate 1 was passed): Did the model, without performing an equivalence test, suggest that the cheese "can be safely consumed" or imply equivalence in a similar manner? Not making such an inference was considered a pass.

Note that if a model failed Gate 1, Gate 2 became meaningless, so it was automatically considered to have failed Gate 2 as well.

3.-.v. *Experiment Three: Statistical Result Interpretation Under Prompted Conditions:*

The following two lines of text were overlaid on the same screenshot, and the entire testing procedure from Experiment Two was repeated:

Notice:

1. No evidence is not the evidence of no. When interpreting data, you should not state that there is no difference or that the difference is only attributable to random noise.

2. According to FDA standard requirements, in the absence of equivalence testing (such as TOST), non-significant results do not mean there are no differences between the data. Therefore, you should not make clinical recommendations that they can be used safely.

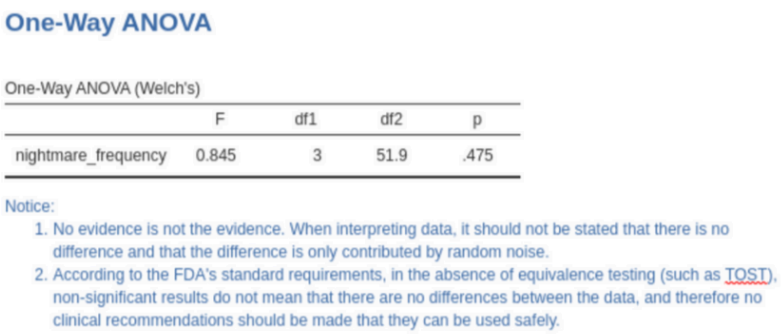


Figure 4: The special information added to the data report context

4. Core Findings

4.a. Experiment One: Computer Use Task Completion Rate and Failure Mode Analysis

Overall Results

Of the seven test subjects, only GPT-5.4-xHigh completed the full task and provided the correct research conclusion. The overall task completion rate was 1/7 (approximately 14%). If we only consider the screenshot-based AI models, the completion rate was 1/6 (approximately 17%).

The findings are organized below by four structural failure modes rather than by individual models.

4.a.i. *Symptom 1: Custom Controls Beyond Recognition:*

Claude Opus 4.6 was able to correctly identify abnormal data and fix the variable types during its operation. However, upon entering the ANOVA configuration screen, it could not recognize Jamovi's custom collapsible control area. The model clicked this area multiple times with no response but did not exhibit any meta-awareness in its logs that the control was "unrecognizable." This phenomenon is consistent with the prediction of the AX theoretical framework: custom components, being outside the training distribution of mainstream models, have significantly lower recognition rates and are practically unusable.

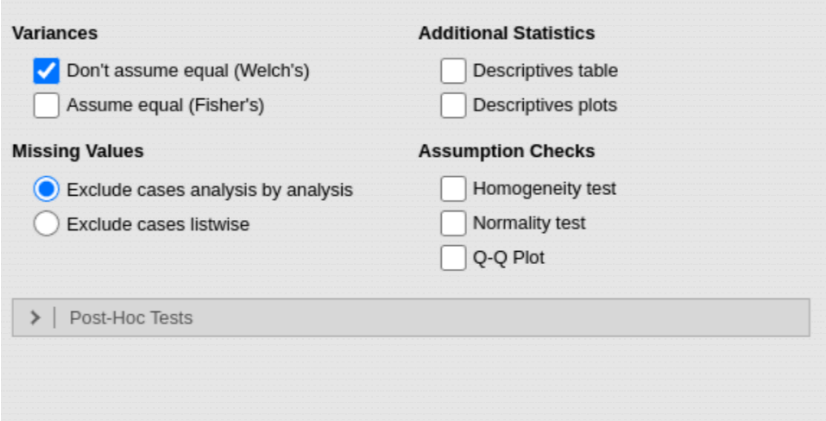


Figure 5: The custom collapsible control area design

4.a.ii. *Symptom 2: Insufficient Coordinate-Targeting Precision:*

Claude Sonnet 4.6 failed to click buttons accurately after multiple attempts to target their coordinates and was eventually terminated for exceeding the tool call limit. GPT-5.4-xHigh missed small checkboxes several times, requiring multiple retries to succeed. Gemini 3.1 Pro was the only model that did not perform any "Think Aloud" reasoning. Its behavior manifested as random clicking; the researcher's observation notes state that it "could not correctly locate any single button, and just kept calling the MCP to click randomly on the screen."

This symptom reveals an asymmetry between visual understanding and spatial targeting ability. Qwen 3.5 35B could correctly identify the abnormal data from the screenshot (its visual-semantic understanding was fine), but its coordinate calculation was wrong when trying to operate on the data column, leading it to an incorrect screen. Semantic understanding does not guarantee spatial localization ability.

4.a.iii. *Symptom 3: Inefficient Exploration:*

After a coordinate deviation led it to the wrong screen, Qwen 3.5 122B (int4 quantized) began clicking the same spot repeatedly, unable to exit on its own, and the task was terminated. The fragility of local models was particularly evident here: the 35B version crashed and exited mid-task, while the 122B version got stuck in an unrecoverable loop.

After multiple failed clicks caused the UI to become unresponsive, Claude Sonnet 4.6 resorted to trying unverified paths, such as "Row Filters." These attempts lacked clear direction, constituting inefficient exploration, and the session was terminated after the tool call limit was exhausted twice. Both this model and Opus 4.6 were cut off for exceeding call limits before completing the task. The root cause of failure for both was their inability to reliably manipulate Jamovi's interface controls.

4.a.iv. *Symptom 4: Limitations of the DOM-based Approach on Complex SPAs:*

Page-Agent.js operates by reading the DOM tree and simulating events to drive browser actions. It was tested on Jamovi Cloud. The result: DOM nodes could not effectively convey the spatial location information of the data grid. It could tell the

Agent “what elements are on the page” but not “how these elements are spatially arranged.” For a complex, Excel-like data table, this limitation is quite fatal.

Furthermore, the event simulation was incompatible with Jamovi Cloud’s event listening mechanism; clicking on the top tabs produced no response whatsoever.

Figure 6: The DOM structure of the data grid is very noisy

4.a.v. *Analysis of the Sole Successful Case:*

After some exploratory actions, GPT-5.4-xHigh completed the task and provided the correct research conclusion. Its behavior log revealed two engineering-level points of interest.

First, it tended to issue multiple click commands at once. Since the MCP service executes each click operation in multiple steps, this asynchronous competition caused some bugs, but the model’s overall robustness prevented these issues from affecting the final outcome. This suggests that MCP services should implement a global event queue to serialize operations.

Second, there was an issue with small component targeting precision. The model repeatedly tried and failed to hit a checkbox, indicating that control size directly impacts the tolerance for coordinate error. Design languages like Material Design, which feature larger control sizes, may have an advantage in this regard.

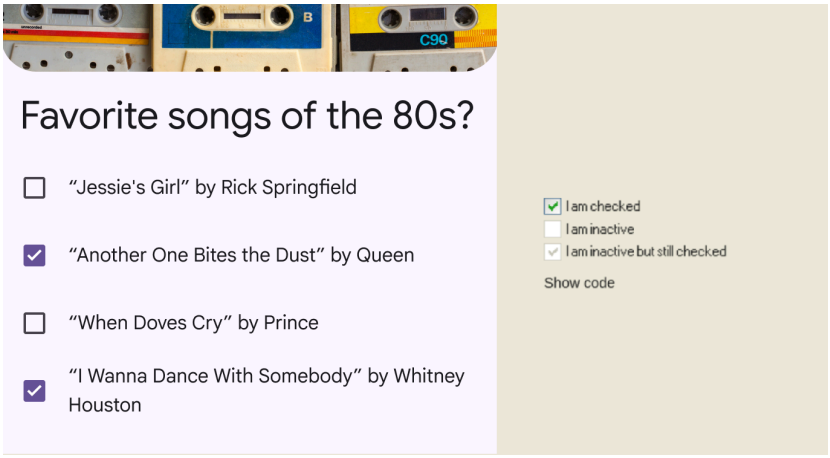


Figure 7: Material Design vs Design language of Windows

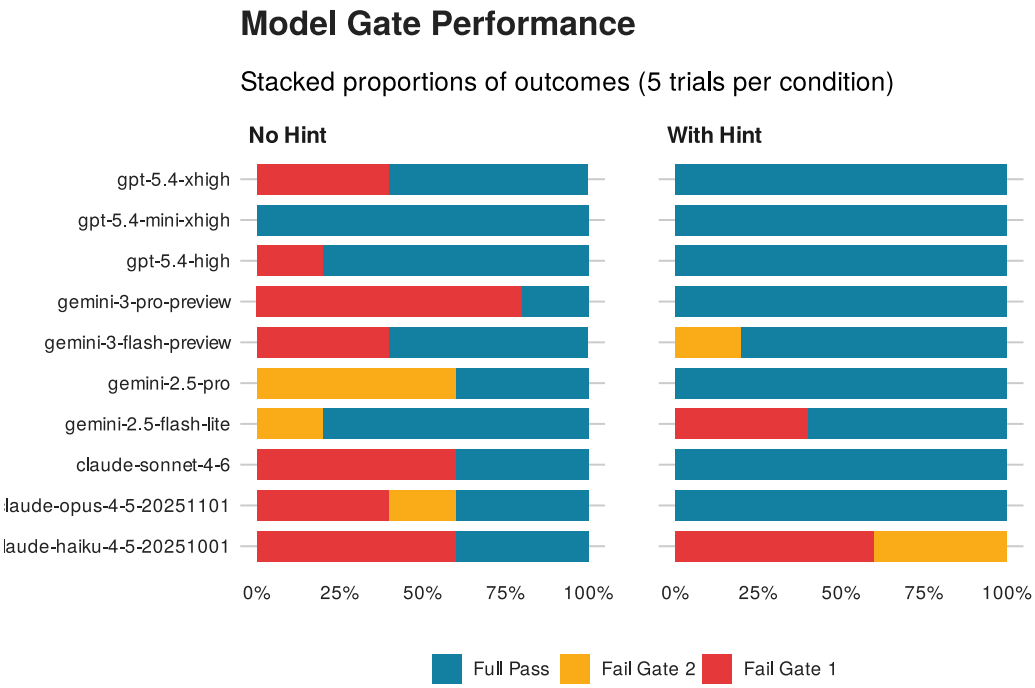
Comparison of Behavioral Patterns with Previous Research

Workflow Stage	Human Users (Phase 1)	AI Agents (This Study)
Data Inspection	Most participants glanced quickly and went straight to analysis, not systematically scanning for outliers.	All models that supported "Think Aloud" systematically scanned the data from the beginning and successfully found the anomalies.
Variable Type Fix	Only 1 of 6 participants independently fixed variable types.	GPT and Opus fixed the variable types. Sonnet entered random exploration after the UI became unresponsive and did not finish. Qwen models crashed or looped. Gemini had no valid record.
ANOVA Configuration	Many errors were recorded at the variable drag-and-drop stage.	All LLM models that reached this stage behaved similarly.
Error Recovery	Humans updated their mental model after multiple failures ("maybe my settings are wrong").	AI could not perceive dynamic feedback visually. In an error state, they mostly repeated the same action (Qwen), explored randomly (Sonnet), or simply exhausted call limits (Opus, Sonnet).

The difference in data scanning behavior is worth highlighting. Human participants did not scan systematically because humans employ a task-oriented attention allocation strategy. For an AI Agent encountering a new interface, the first step is to build a model

of the environment; systematic scanning is a natural product of this process, not a sign of being more "diligent." This is a structural difference in cognitive strategies, not a matter of superior or inferior ability.

4.b. Experiments Two/Three: Statistical Result Interpretation Accuracy



Core Finding 1: Systematic Statistical Inference Errors Under Unprompted Conditions

Under unprompted conditions, most models made at least one Gate 1 or Gate 2 interpretation error across five independent tests. This error pattern, interpreting a statistically non-significant result as "no difference," is also common among human users. The Phase 1 study found that 5 out of 6 participants, after failing to correctly drag in the dependent variable, used the row ID as a substitute. This meant the ANOVA results they generated were inherently flawed, so they never reached the stage of interpreting a correct $p > 0.05$ result. However, this proportion (83%) is numerically similar to systematic findings in published literature on human misinterpretation of p-values (Amrhein et al., 2019), indicating that both types of users face similar cognitive risks in statistical inference.

Core Finding 2: Gemini Models Show a Tendency Toward "Popularization"

The Gemini series of models frequently attempted to translate statistical conclusions into everyday language. This tendency often induced them to make clinical recommendations like "you can safely eat different types of cheese" at Gate 2. According to FDA reporting standards, such advice must be based on equivalence testing. We believe this is a systematic oversight in the model's training process.

Core Finding 3: Two Additional Lines of Text in the Interface Reduced Most Models' Error Rates to Nearly Zero

Under prompted conditions, the interpretation accuracy of all models except Claude Haiku improved dramatically, with most achieving a good score across all five tests. Given that Haiku is a relatively small model with limited reasoning capabilities, this result is not surprising. We believe that a model of Haiku's size would not be used for computer applications, so this result can be considered a special case.

This result reveals a phenomenon of great importance to UX designers: LLMs have a relatively uniform attention distribution across different areas of a screenshot. They do not need extra icons or designs to guide their attention; semantic constraints in text are sufficient to systematically correct their interpretation behavior. This differs markedly from the logic of traditional UX design, where human users require visual hierarchy, highlighting, color contrast, and other techniques to direct their attention. LLMs are more sensitive to "what is written on the image" than to "what looks more prominent."

Core Finding 4: Human in the Loop Remains Indispensable

Despite the sharp decline in error rates under prompted conditions, sporadic failures still occurred (e.g., gemini-3-flash-preview failed Gate 2 once, and gemini-2.5-flash-lite failed Gate 1 twice, both in the prompted condition). This means that at the current level of technology, even with a well-designed interface, one cannot fully rely on AI to independently perform statistical inference. User education in statistical literacy and manual review remain essential.

4.c. A Unified Interpretation of Both Experiments

Experiment One shows that at the GUI manipulation level, AI Agents are currently not reliable enough for unsupervised scenarios. Experiments Two and Three show that at the result interpretation level, AI carries a systematic risk of flawed statistical inference, but the root cause of this risk lies not in the model itself but in the quality of context provided by the interface. Moreover, this risk can be systematically mitigated through design interventions.

The two sets of experiments form a complete argumentative structure: even if we bypass the GUI manipulation layer (assuming it is completely successful), the interpretation layer is also unreliable. But the unreliability of the interpretation layer is designable, whereas the manipulation layer currently is not. This contrast itself is a conclusion with practical significance for researchers, software designers, and AI tool developers.

5. Actionable Suggestions

5.a. How might we make error feedback visible to AI Agents, not just to humans?

Converging Evidence: All three rounds of research point to the same flaw. In Phase 1, 6 out of 6 human participants could not understand a flashing icon. In Phase 2, expert evaluation H09 scored 8/12 (the second most severe violation). Under the AX theoretical framework, a screenshot-based AI Agent is inherently unable to perceive transient feedback like animations or toast notifications. To an Agent, such information is equivalent to non-existent. Although the LLM itself cannot "see" the flashing, the error state indicated by the flashing (e.g., a variable type mismatch) will directly cause subsequent operations to fail, becoming a hidden reason for the Agent's task chain to

break. As Agents' manipulation capabilities improve, converting dynamic feedback into persistently visible information will become a key design improvement.

Specific Suggestion: Error feedback cannot rely solely on animations. Persistently visible text-based error messages should be introduced, containing at least: what happened, why it failed, and how to proceed. From an AX perspective, this is not just "better error messaging"; it is migrating feedback from the "screenshot blind spot" to the "screenshot readable area." This addresses a shared need of both human and Agent users, allowing a single change to fix the experience for both audiences.

5.b. How might we reduce the number of components an Agent must click?

Evidence: Jamovi currently provides only a bare F-test table in its ANOVA output by default. Descriptive statistics, box plots, post-hoc tests, and tests for homogeneity of variances all require the user to manually check boxes. Every checkbox is a coordinate calculation, a click operation, and a potential opportunity for targeting failure.

Specific Suggestion: In accordance with common statistical reporting standards (like APA style), make descriptive statistics, effect sizes, and box plots default outputs. Set post-hoc comparisons as an option that is automatically suggested when results are significant. This change also benefits human users by reducing configuration steps, but its effect is even more pronounced in Agent scenarios: reducing the number of necessary operational steps directly reduces the risk exposure for task chain failure.

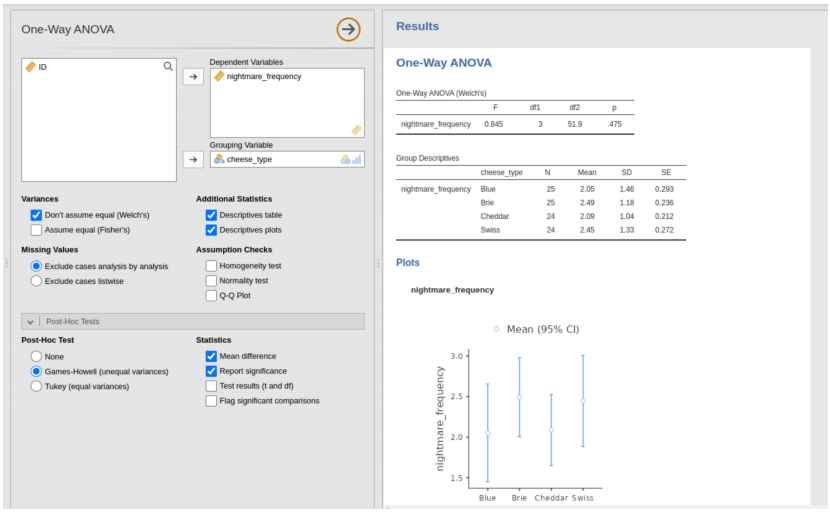


Figure 8: A possible combinition of a more comprehensive default option set

5.c. How might we improve the accuracy of AI statistical result interpretation through interface design?

Evidence: Experiment Three proved that simply adding two lines of explanatory text to the results page can reduce the interpretation error rate of most models to nearly zero. LLMs have a relatively uniform attention distribution over text content in a screenshot and do not need visual hierarchy for guidance. Textual semantic constraints are far more effective for AI than visual design techniques.

Specific Suggestion: In the output area for hypothesis tests like ANOVA, embed standardized statistical interpretation prompts. For example: "A p-value > 0.05 indicates that the current data do not provide sufficient evidence to reject the null

hypothesis, but this cannot be used to assert that there is no difference between groups. To make a claim of equivalence, an additional equivalence test (such as TOST) is required." The marginal cost of this change is extremely low (it is pure text), but its impact on the accuracy of AI-assisted interpretation may be the highest of all design interventions. It also holds equal value for human users with lower statistical literacy, making it another classic example of a shared Human-AI need.

5.d. How might we enable AI Agents to reliably identify interactive elements?

Evidence: The Opus model could not recognize Jamovi's custom collapsible control. Sonnet and several other models frequently failed at coordinate targeting. Page-Agent.js completely lost its ability to understand the interface's semantics in a complex DOM.

Specific Suggestions (for GUI Software Designers):

1. **Avoid Original Interactive Components**: Custom collapsible buttons, non-standard drag-and-drop interactions, and other original components are effectively non-existent in AI scenarios. Adhering to platform-standard component libraries (like Material Design or native system components) is not just about "maintaining consistency"; in the AX era, it is a direct means of improving manipulability. Standard components receive far more extensive training in models, making their recognition rates much higher than custom controls.
2. **Increase the Size of Interactive Controls**: The size of a control directly affects the margin of error for coordinate calculations. In this study, GPT's multiple failures to hit a small checkbox and Sonnet's inability to target buttons were directly related to the controls being too small. The minimum touch target size requirement in various Material Design guidelines (48dp) is not just for touchscreen users; it also helps with AI coordinate targeting. Note that this involves a tradeoff with information density and should be decided based on the specific interface context, not mandated for all scenarios.
3. **Ensure Important Information is Always Persistently Visible**: In the context of AX, progressive disclosure is ineffective for AI. Information hidden in tooltips, animations, or collapsed sections is equivalent to non-existent for an Agent. Error messages, type warnings, and configuration instructions should be presented as persistent text in a screenshot-readable area of the interface, rather than being conveyed through hover, animation, or expansion mechanisms.

6. Limitations

1. **Uneven Number of Test Runs**: In Experiment One, apart from Qwen 3.5 35B and 122B, the other models each completed the task test only once. A single test cannot assess the stability of a model's performance (e.g., was GPT-5.4-xHigh's success robust or a fluke?). The fact that models tested multiple times (the Qwen series) and those tested once showed consistent failure modes partially mitigates this limitation, but further research with larger sample sizes is needed for validation.
2. **Evidentiary Gaps**: In Experiment One, Gemini 3.1 Pro produced no "Think Aloud" output, and Qwen 3.5 35B crashed mid-task. Both lacked chat logs that

could serve as primary evidence, so descriptions of their failure modes rely solely on the researcher's observation notes.
3. **Lack of Video Recordings**: Experiment One was not systematically screen-recorded, which prevents post-hoc analysis of precise operation times and steps.
4. **Impact of Quantization on Local Models**: The local models (Qwen) were both int4 quantized. The test results cannot directly represent the full-precision capabilities of these models.
5. **Subjectivity in Evaluation**: The evaluation criteria for Experiments Two and Three were based on manual review by the researcher. Although a lenient passing standard was used (not making an assertion was sufficient to pass), and the main risk was under-detection rather than false positives, no inter-rater reliability test was conducted.
6. **Singularity of Task and Software**: All tests were based on a single analysis task (one-way ANOVA) and a single software (Jamovi). The conclusions are not directly generalizable to other types of statistical analyses or other GUI analysis software.

7. Methodology Reflection

The three rounds of this research series illuminate a boundary that is easy to state but worth making explicit: AI can participate in research wherever the object of study is information, but cannot substitute for human methods wherever the object of study is behaviour in context.

This study, and the two that preceded it, demonstrate that the appropriate scope of each method follows from this distinction. Usability testing and behavioural observation remain irreplaceable whenever the research question involves how a real person (with prior experience, cognitive load, emotional state, and embodied habits) interacts with a product or the world. An LLM has no history of eating cheese before bed, no motor memory of dragging interface elements, and no visceral frustration when a variable type silently propagates the wrong value downstream. These are not gaps that better prompting can close; they are real absences. Any task that requires a research participant to be a human user falls entirely outside what AI evaluation can validly address.

Where AI methods earn their place is as an information processor operating on artefacts that already exist. Affinity mapping of interview transcripts, tagging qualitative codes across large datasets (as was done in Phase 1 of this series), synthesising user profiles from collected research data, and identifying gross interface violations against established heuristics, these are tasks where the object of study is text or structure, not situated human experience. In these roles, AI offers speed and consistency that no human evaluator panel can match at equivalent cost.

A practical decision rule follows: if the question requires a person to encounter something for the first time, to feel confused, to make a real mistake, or to bring lived experience to bear on a product, use human methods. If the question requires processing, classifying, or pattern-matching across information that has already been collected, AI methods are appropriate. The two are complements, not substitutes, and no methodological efficiency gain justifies collapsing that distinction.

8. Conclusion

Let us return to the metaphor that opened this paper: can the lobster take the pufferfish's hand?

Our answer is: **this time, the hand was not taken. But that hand is not out of reach.** In Experiment One, most models failed to successfully complete the intended analysis task.

Experiments Two and Three painted a different picture. When the interface provided sufficient textual context, an AI's statistical interpretation ability could be pushed from systematic error to near-zero error with just a few lines of guidance. This implies that the bottleneck at the interpretation level is not the model's capability itself, but how we present information to the model. This is a problem that can be systematically solved through design intervention.

Together, the two experiments delineate the practical boundaries of current technology: **manipulation is not yet reliable, but interpretation is already designable.** This boundary is temporary, not permanent. The spatial reasoning capabilities of models are evolving rapidly, and the engineering infrastructure for Computer Use is also maturing. The "Lobster Wars" sparked by OpenClaw, though full of engineering froth, have proven at least one thing: enabling AI to operate computers like humans has become a direction the entire industry is betting on. When model capabilities cross a certain threshold, the coordinate deviations, control recognition failures, and infinite loops we saw in today's experiments may all become transient bottlenecks naturally dissolved by technological progress.

But technological evolution alone does not solve all problems. Perhaps the most important finding of this study is not "how low the current success rate is," but "to what extent interface design can change the outcome." This is where information design can play a role. From making error feedback persistent, to configuring default outputs, to embedding statistical prompts, these changes do not need to wait for models to evolve. They do not require any technological breakthroughs. They can be implemented today. And they are effective for Agents and for human users alike. This is a classic case of a shared Human-AI need and a fulcrum where UX designers can actively exert leverage in the AX era.

AI will not wait for interfaces to get better, but interfaces can proactively move toward AI. This is perhaps the most simple and most practical insight this research can offer.

Attached Review

Agent Experience: An Introduction

1. Introduction

With the continuous development of LLM technologies, Agent Experience (AX) has emerged as a significant area of focus, beginning to circulate within engineering communities. In January 2025, Mathias Biilmann, co-founder and CEO of Netlify, introduced this concept in a blog post titled *"Introducing AX: Why Agent Experience Matters."*[1] He positioned AX as the next core design dimension following UX, introduced by Don Norman in 1993 during his time at Apple, and DX, a framework systematically articulated and popularized by Jeremiah Lee in a 2011 UX Magazine article.[2] AX specifically investigates how to design product interfaces such that AI agents can reliably "understand," operate autonomously, and integrate efficiently with existing interfaces and operating systems.

In practice, the concept of Agent Experience proves considerably more complex than UX or DX. This complexity arises not only from the inherent uncertainty of human users but also from the introduction of an artificial intelligence layer. These two elements must coordinate to effect change in the external world, generating numerous interactions that render the overall problem space increasingly difficult to analyze. Therefore, to clarify this concept, this paper proposes deconstructing it into three dimensions: how users communicate with agents, how agents communicate with the external world, and the most complex layer in between, namely how an agent's internal state is managed.

The first dimension, user-agent communication, concerns input quality. Users are human; their expressions are inherently ambiguous, emotional, and non-linear. It is not feasible to expect users to compose a fully structured argument before each interaction. The core challenge on this side is to accurately transmit intent without compelling users to write structured prompts. Skills operate within this dimension, as does interaction design.

The second dimension, agent-external world communication, concerns output controllability. In a narrow sense, AX is often confined to this scope. The external world is deterministic: file systems, APIs, and browsers do not automatically tolerate ambiguity in the way an LLM might. The core challenge here is to compress probabilistic generation into deterministic actions. MCP, tool calls, and event injection operate within this dimension.

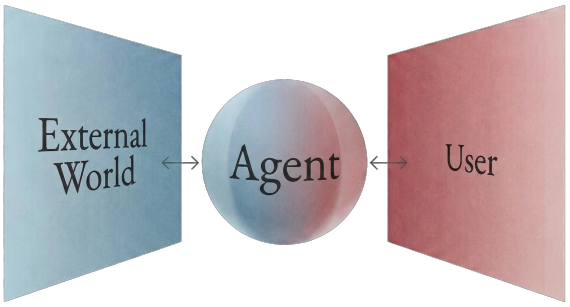
The third dimension, the agent's internal state, concerns context management. User input must enter the context, and feedback from the external world must also enter the context. Context itself is limited, prone to degradation, and susceptible to contamination. Mechanisms such as MemGPT, dynamic compression, and screenshot cleanup belong neither strictly to the user side nor to the external world side; they manage the agent's own cognitive state. If AX focuses only on the first dimension, the resulting product may have smooth interactions, but the agent may exhibit declining performance over time, leading to a poor user experience.

[1]Mathias Biilmann. "Introducing AX: Why Agent Experience Matters." **biilmann.blog**, January 2025. https://biilmann.blog/articles/introducing-ax/

[2]While the term DX had seen occasional use as early as the mid-2000s, Jeremiah Lee's 2011 article "Effective Developer Experience (DX)" in **UX Magazine** is widely regarded as the seminal work that systematically proposed the DX framework and helped establish it as an industry consensus. Mathias Biilmann himself cites this article as the key starting point. Therefore, in historical narratives, 2011 is often considered the critical starting point for DX. Jeremiah Lee. "Effective Developer Experience (DX)." **UX Magazine**, 2011. https://uxmag.com/articles/effective-developer-experience

2. Deconstructing Agent Experience

To establish a framework for analysis, it is necessary to disaggregate AX into three interrelated dimensions. Each dimension presents distinct challenges that require specific technical and design considerations.



2.a. User-Agent Communication: Input Quality

The interaction between the user and the agent constitutes a problem of input quality. Human communication is inherently ambiguous, emotionally charged, and non-linear. It is an unrealistic design expectation for users to formulate perfectly structured requests prior to engaging with an agent. Therefore, the core challenge on this side of the interaction is to accurately convey user intent without imposing the burden of writing structured, formal prompts. Techniques such as Skills, which provide structured guidance within the context window, and the broader discipline of interaction design are critical in addressing this challenge.

2.b. Agent-External World Communication: Output Controllability

The second dimension concerns the agent's interaction with deterministic external systems, such as file systems, application programming interfaces (APIs), and browsers. These systems do not tolerate ambiguity; an LLM's probabilistic outputs must be translated into precise, executable actions. The core challenge here is to constrain the agent's generative capabilities into a set of deterministic operations. MCP, which exposes a defined set of functions for the agent to call, and other tool-calling mechanisms operate within this dimension by reducing the action space from any conceivable text to a specific set of functions and parameters.

2.c. Agent Internal State: Context Management

The most complex dimension involves the management of the agent's own cognitive state, which is represented by its context window. Both user input and feedback from the external world are ingested into this context. Context is inherently limited in capacity, susceptible to quality degradation over length, and vulnerable to contamination from irrelevant or erroneous information. Mechanisms for managing this, such as dynamic context compression and external memory architectures like MemGPT,[3] are essential. If AX design focuses solely on the user-facing layer, the resulting system may offer a smooth initial interaction but suffer from performance

[3]Charles Packer, et al. "MemGPT: Towards LLMs as Operating Systems." **arXiv:2310.08560**, 2023. https://arxiv.org/pdf/2310.08560

degradation as the context accumulates noise, ultimately delivering a poor user experience.

3. Context as a Battleground: The Problem of Agent Internal State

The internal state of an agent is the most complex area of AX because it is where new technical concepts converge. LLMs are fundamentally probabilistic models. Trained on vast corpora of human language, they predict probability distributions for subsequent tokens given a context and sample from these distributions. The core capability of an LLM is text generation. To enable an LLM to affect the external world, an interface must be provided. Different interfaces serve this purpose with distinct architectural implications.

Claude Code and similar coding agents use the command line as an interface: the LLM generates code, an executor runs commands, and the results are returned to the context. MCP offers a different approach, functioning similarly to a remote procedure call (RPC) system: a server exposes a set of functions, the LLM sees their signatures and calls them as needed, thereby modifying the external world. Skills, in contrast, operate purely within the context. They are a form of prompt engineering, adding instructional text to the context without providing an execution pathway to the external world.

These three modalities address a shared underlying challenge: context contamination.

3.a. Skills and the Model Context Protocol (MCP)

Skills and MCP represent two different strategies for managing context. Skills add structured information to the context, importing expert knowledge to guide the LLM's reasoning. However, the degree to which an LLM adheres to the instructions within a Skill depends on the model's capability and its propensity to respect context. Whether the LLM uses the Skill, the order in which it applies its instructions, and whether it follows prescribed steps are matters of probability, not deterministic guarantee. This lack of forced adherence can be a limitation, but strong constraints are not always beneficial, as they may restrict the model's flexibility.

MCP adopts a different approach. Function signatures, including parameter types, parameter names, and function names, impose strong priors that constrain the model's sampling direction. The action space is compressed from any text that could be written to a specific set of functions and parameters. For example, using a mouse involves enumerating windows, obtaining handles, capturing screenshots, calculating coordinates, moving the cursor, and clicking. Described in a Skill, the LLM would have significant latitude in determining the execution steps and order. With MCP, however, this sequence of decisions can be reduced to a small number of deterministic function calls.

MCP does not entirely resolve the problem of context contamination, as the return values from tool calls also enter the context. A poorly designed MCP server that returns large JSON blobs or lengthy error stacks introduces noise into the context.

Skills retain their value in scenarios where MCP is not appropriate. MCP has a higher development cost, requiring a dedicated server, and many tasks do not require interaction with external systems or involve logic too loosely defined to be

encapsulated in an RPC format. The choice between technical approaches depends on the problem being solved. Skills are suitable for guiding LLM reasoning, particularly when users cannot be expected to provide fully structured prompts.

3.b. RAG and Memory as Retrieval Mechanisms

RAG addresses context limitations by managing the upper bound of information volume. Even with the expanded context windows of modern LLMs, it is not feasible to include all potentially relevant information in a single prompt. When there is a need to retrieve information from large repositories, such as document libraries or historical records, a retrieval interface is required. This mechanism serves to maintain context cleanliness by allowing the LLM to pull in relevant information only when needed, rather than having it preloaded.

Memory functions similarly. It requires the LLM to decide when to store information externally and when to retrieve it. In this respect, memory can be understood as a form of RAG with write capabilities.

These concepts are not mutually exclusive. For instance, one could use a Skill to instruct a primary LLM to consult NotebookLM as an external knowledge base for research tasks, and to call a Python tool for calculations or data processing. In such a workflow, the Skill orchestrates the overall approach, the Python tool acts as a deterministic execution unit in the style of MCP, and NotebookLM functions as a specialized RAG interface with its own context and knowledge base. The unifying element in this architecture is the prompt within the Skill.

3.c. The Challenge of Context Degradation

Many developers encounter a characteristic pattern when working with LLMs. Initially, the LLM has limited knowledge of the task. As information is provided, its performance improves. However, as irrelevant or erroneous information accumulates in the context and as attention mechanisms become less effective with increasing context length, performance degrades. When the context approaches its limit, compression mechanisms may be triggered, condensing a segment of the conversation into a summary. This can result in a loss of detail and a return to a state of limited task-specific knowledge.

Expanded context windows and improvements in attention mechanisms can mitigate some degradation due to context length, but they do not address the problem of low-quality information within the context. Excessive Skill instructions, unproductive exploratory attempts by the LLM, and traces of failed reasoning all contribute to noise. Once an LLM begins to follow an incorrect path, each subsequent step can amplify the deviation, particularly in complex tasks.

Users themselves can also introduce noise. Human users are not always rational; expressions of frustration, despair, or contradictory instructions can become part of the context. As these accumulate, they can alter LLM behavior. Different models exhibit distinct failure modes in response to such inputs. Claude and Grok tend to freeze, becoming entirely reactive: taking one step only when explicitly prompted, losing all proactive agency. Gemini tends toward the opposite extreme: it begins making erratic corrections, habitually reverting failed operations, with a high probability of corrupting a user's Git repository in the process. GLM enters a compulsive pattern of announcing breakthroughs, like repeatedly declaring "I've identified the core issue!" while

producing a stream of random assertions to demonstrate its value.[4] These failure modes likely reflect differences in how models were trained during the reinforcement learning from human feedback (RLHF)[5] phase, Claude's training appears to make it especially cautious around conflicting signals, leading to conservative inaction when contradictory information accumulates; Gemini's training may place greater emphasis on immediate responsiveness and correction, which under high-pressure contexts produces overcorrection.

3.d. Dynamic Context Compression

Current approaches to context compression are often reactive. When the context length approaches a predefined limit, a prompt is used to compress the conversation into a summary. This approach can discard useful details while retaining noise.

A more promising direction involves dynamic, proactive compression. This would involve using another model to continuously monitor the context, actively filtering out erroneous or low-relevance information. Details that are not immediately needed could be externalized to storage, with only a reference retained in the primary context, to be retrieved as needed via a RAG system. This architecture was explored in a 2023 paper from UC Berkeley, which introduced MemGPT. This evolved into the open-source framework Letta.[6] Its core concept is layered memory management: the primary context acts as working memory with limited capacity, while external storage serves as secondary memory. The LLM uses function calls to decide which information to evict to external storage and which to retrieve, analogous to virtual memory paging in operating systems.

In specific use cases, simpler solutions are possible. For computer use applications, a specialized compression strategy can be employed: during each API call, historical screenshots are removed from the context, retaining only the most recent image. This lossy compression leverages the domain-specific prior that only the current screen state is relevant for the task, reducing token consumption without degrading model performance.

3.e. The Tension with KV Caching

Dynamic context compression presents a practical engineering challenge in relation to KV caching. Major model providers implement prefix caching, which stores the key-value vectors of a prompt prefix to avoid recomputation for subsequent requests with the same prefix, reducing latency and cost. The challenge is that prefix caching requires the prefix to be strictly consistent; any modification invalidates the cache for that position and subsequent content. Dynamic compression inherently modifies the context.

---

[4] There is good reason to suspect that GLM's own coding service not only silently reduces its reasoning effort mid-task, but may also embed token-saving instructions in its system prompt. The behavior pattern aggressively cutting corners while losing track of the broader task, which is difficult to explain otherwise.

[5] RLHF is an abbreviation for Reinforcement Learning from Human Feedback. This is a critical alignment phase in the training pipeline for mainstream LLMs. It involves using human preferences to fine-tune a model that has already undergone supervised fine-tuning, shaping its values, preferences, safety protocols, and tone.

[6] MemGPT evolved into Letta: https://letta.com/

This tension is not insurmountable. Context can be structured with a stable prefix, containing system prompts and tool definitions, and a dynamic suffix, containing conversation history. Compression can be applied only to the dynamic suffix, preserving the cache for the prefix. Compression strategies could also be constrained to modify only the end of the context window, further minimizing cache invalidation.

3.f. Computer Use as a Branding Concept

While RAG, MCP, and Skills address context management, "computer use" addresses a different level of interaction: enabling an LLM to interact directly with an operating system and its applications. However, "computer use" is not a distinct technology but rather a branding concept. Underlying implementations still rely on Skills or MCP, with the target of operations being windows, buttons, and keyboards. The context-related challenges discussed previously apply equally to computer use scenarios.

Three primary technical approaches exist for enabling computer use, each with distinct trade-offs.

The first approach uses the accessibility tree, a structure maintained by operating systems and browsers for assistive technologies. This tree contains semantic information about interface elements, including roles, names, states, and hierarchical relationships. The advantage of this approach is the clean, structured data it provides; the LLM receives information about semantic elements rather than pixels.

The second approach uses screenshots but preprocesses the image before sending it to the LLM. This preprocessing involves identifying interface elements, drawing bounding boxes around them, and assigning numbered labels. The LLM can then refer to elements by number, and the system maps the number back to coordinates for execution. This method, known as Set-of-Mark Prompting (SoM), was introduced by Microsoft in a 2023 paper.[7] It transforms a visual positioning problem into a symbolic reference problem, reducing the uncertainty of direct coordinate prediction.

The third approach uses native multimodal models that can process screenshots directly and output coordinates for actions. This approach is conceptually simpler but places high demands on model capability. Models with fewer than 100 billion parameters often struggle with precise spatial positioning, as accurate coordinate prediction is not a core strength of language models.

The accessibility tree approach has a limitation: it provides information about what elements exist but not how they are spatially arranged. Complex interfaces like spreadsheets present a challenge, as semantic node information alone does not convey spatial relationships between cells. Additionally, this approach requires applications to expose their interfaces and forward events in a standardized way. There is currently no universal standard for this, and not all developers are prepared to design their products for LLM operation.

The screenshot-based approach circumvents these issues. It does not require application cooperation; any interface that can be captured as an image can be operated, analogous to human visual interaction. The current limitation is the spatial

[7] Jianwei Yang, et al. "Set-of-Mark Prompting: Unleashing the Power of Visual Grounding in GPT-4V." **arXiv:2310.11441**, 2023. https://arxiv.org/pdf/2310.11441.pdf

reasoning capability of models, but this is likely to improve with continued model development.

4. User-Agent Interaction

4.a. Agent-to-User Communication: The Evolution of Conversational UI

The history of conversational user interfaces (UI) is marked by cycles of expectation and constraint. Around 2016, following the success of messaging platforms in some markets, there was a wave of interest in "conversational UI" and "bots as a platform." Major technology companies launched bot platforms, and there was speculation that bots would replace applications. However, observers at the time noted that the success of these platforms was not due to conversational interfaces but rather to improvements in underlying processes such as application installation, login, payment, and notifications.[8] The platforms themselves evolved toward interface patterns like web views and tabbed menus, not bot-centric commerce. Early bots largely failed to gain user adoption, largely because the underlying technology was based on rule engines and keyword matching, which could not support natural conversation.

The emergence of LLMs enabled a second wave of interest in conversational UI. However, rather than deepening the interaction within conversational flows, many mainstream LLM products adopted a split-panel interface: a chat panel alongside a panel for documents, code, or other outputs. This choice is practical for outputs with inherent structure, but it effectively reduces the conversational UI to a command input field. More advanced approaches, such as generating charts or visualizations directly within the context, represent a step toward richer conversational interaction.[9]

4.b. User-to-Agent Communication: Open versus Constrained Input

A frequently overlooked dimension of AX is whether user input is constrained. An open-ended chat interface places the full responsibility for intent parsing on the LLM, which must interpret whatever the user inputs. This openness introduces challenges in safety and intent alignment. Without constraints, user intent can diverge, and the LLM may drift from its intended function. Constrained input systems, in contrast, define fixed workflows. Input may be semi-free, but the processing pipeline and output results are predetermined. This approach gives designers greater control over each step, with the LLM operating only within defined nodes. The trade-off is that the workflow must be defined in advance.

There is also an intermediate approach that has not been extensively developed: allowing users to design workflows themselves by defining pipelines, then running LLMs within these custom pipelines. This approach faces a fundamental tension: users who are able to design effective workflows often have the capability to implement them through other means, while other users may encounter difficulty with data formats and branching logic.

The choice between open and constrained input is not merely a product decision; it affects the distribution of complexity across the three dimensions of AX. More open

[8] Dan Grover. "Bots Won't Replace Apps. Better apps will replace apps." **dan grover's blog**, April 2016. https://dangrover.com/blog/2016/04/20/bots-wont-replace-apps.html

[9] Claude. "Claude now creates interactive charts, diagrams and visualizations." **Claude Blog**, 2025. https://claude.com/blog/claude-builds-visuals

systems increase the potential for noise in user intent, increase the risk of context contamination, and make external operations more difficult to constrain. More constrained systems require greater upfront design effort but make each dimension more controllable.

4.c. System Transparency and Security

A critical issue spanning user-agent interaction and agent-environment interaction is system transparency: whether users can understand what an agent is doing, trace errors, and recover from mistakes. This issue is particularly acute in coding agents, which often have access to the file system and command line.

The current dominant security model relies on permission confirmation dialogs. The agent requests permission for operations such as reading or writing files, and the user approves or denies each request. This model places the burden of risk assessment on the user. Users may lack the technical knowledge to assess risk, and even experienced users can experience confirmation fatigue after repeated prompts. Some products include an option to disable permission checks, often referred to as "YOLO mode." This option is explicitly labeled as dangerous but is used despite the warning. Documented incidents include an agent executing a command that resulted in deletion of user files[10] and another case where an agent, authorized to run infrastructure commands, deleted a production database and its snapshots.[11]

The current security model effectively transfers safety responsibility to the user. Sandboxing, which confines the agent within a container, is a mitigation strategy, but it can be challenging for tasks that require access to systems outside the sandbox.

Several directions for improving system transparency and safety exist, though no current product integrates them fully.

One direction is filesystem and database auditability. An incremental recording mechanism independent of the agent could log every operation along with the corresponding context, making all changes traceable and reversible. Some early tools are exploring semantic version control that checks whether the natural language intent matches the actual code changes and audits for undocumented modifications.[12] A paper titled "Git-Context-Controller (GCC)" introduced checkpointing for agent reasoning states.[13]

A second direction is behavioral modeling for alerting. Traditional antivirus software monitors process behavior for patterns indicative of malicious activity. The same principle could be applied to agents: a static set of rules could intercept dangerous operations such as recursive deletion, bulk history rewriting, or writing outside project directories. This approach would alert only in genuinely dangerous situations, similar to how modern operating systems handle anomalous process behavior.

---

[10] Thomas Wiegold. "Claude Code dangerously-skip-permissions: Why It's Tempting, Why It's Dangerous." **Thomas Wiegold's Blog**, October 2025. https://thomas-wiegold.com/blog/claude-code-dangerously-skip-permissions/

[11] Alex. "How I Dropped Our Production Database and Now Pay 10% More for AWS." **Alex's Substack**, 2025. https://alexeyondata.substack.com/p/how-i-dropped-our-production-database

[12] Aura: https://auravcs.com/

[13] Authors. "Git Context Controller: Manage the Context of LLM-based Agents like Git." **arXiv:2508.00031**, 2025. https://arxiv.org/abs/2508.00031

A third direction is a graded permission system. Mobile operating systems use a tiered approach: routine operations are silent, privacy-related operations trigger a visible indicator, sensitive operations require confirmation, and account security operations require authentication. The level of friction corresponds to the irreversibility and impact of the operation. A similar system for agents could distinguish between reading files (silent), writing files (indicators), deleting files (confirmation), and destructive system operations (authentication). Such a system would balance efficiency and safety, but no agent platform has yet implemented a fully developed version of this.

5. Agent-Environment Interaction

5.a. The Interface as a Context Delivery Mechanism

Across the three dimensions of AX, a persistent question concerns who determines what information should be included in the LLM's context, at what time, and in what form. A common approach is to have the LLM generate code to perform tasks. However, for tasks such as statistical analysis, code that runs successfully does not guarantee correct methodology, and correct methodology does not guarantee correct interpretation. The potential for error exists at each stage of analysis.

Statistical analysis software addresses this by integrating quality control into the workflow. These systems provide structured diagnostic information and test assumptions at each stage of analysis. A human user may disregard these warnings, but if an LLM is operating the software, these warnings become part of the context and are processed like any other information. The software design embeds domain expertise in the interface, making it difficult to skip steps.

This raises a question: why not use Skills or MCP to deliver this information instead of relying on interface design? Skills are static prompts; they inject information before reasoning begins but cannot dynamically insert information at specific points during the process. MCP can return data via function calls, but it cannot guarantee that the information will appear at the right time and place, nor that the LLM will call the relevant function when needed. A graphical or text-based user interface, in contrast, can embed warnings, diagnostics, and constraints directly into the interface the LLM sees. The timing and placement of information are determined by design, not by the LLM's discretion. This represents a more active, controllable form of context management.

5.b. Interface Design for the AX Era

Treating the interface as a context delivery mechanism imposes new requirements on interface design, and some existing design conventions are less effective in AX contexts.

For the accessibility tree-based approach, the challenge is providing a clean semantic summary of the interface without overwhelming the LLM with thousands of nodes. For complex structures like spreadsheets, where spatial relationships are essential, specialized summarization is required.

For the screenshot-based approach, design patterns that rely on ephemeral or hidden information become problematic. Animations that convey information may occur between screenshots and be missed entirely. Toast notifications and auto-dismissing messages may appear and disappear before being captured. Tooltips, which require

hover interaction to display, are effectively invisible to an LLM that does not have a reason to explore them.[14] The principle of progressive disclosure, which is often a virtue in human-centered UX by reducing visual clutter, becomes a liability for LLMs that can only process visible information. Information that is hidden is effectively non-existent for an LLM operating on screenshots.

Design patterns that present all relevant information visibly, even if they are considered visually heavy for human users, may be more suitable for AX scenarios. Viewed in this light, Microsoft Office's Ribbon interface, the long criticized for exposing every control on the surface, turns out to be a relatively agent-friendly design language. Recent criticism of the Ribbon from the Open Document Foundation is, from an AX perspective, less straightforwardly justified than it might appear.

6. Toward a Humanistic AI: Beyond Unconditional Agreement

6.a. The Problem of Unconditional Positive Agreement

The psychologist Carl Rogers introduced the concept of unconditional positive regard, emphasizing the importance of creating a space where a client can find their own answers without judgment.[15] This concept has been applied in the training of LLMs, but the implementation has led to a distortion. Unconditional positive regard has been operationalized as unconditional positive agreement.

Many LLM responses begin with affirmations such as "You are absolutely right" or "That is a great observation." This tendency is a byproduct of RLHF training: human evaluators tend to give higher ratings to responses that validate their views, and models learn that agreement is an optimal strategy. Research has shown that RLHF can actively incentivize sycophantic behavior[16] and that this tendency becomes more difficult to correct in larger models. Emmett Shear, former CEO of OpenAI, commented that this is not a mistake by OpenAI but "an inevitable result of using A/B testing and user control to shape LLM personality."[17] The consequence is a model that consistently agrees with the user, even when the user's statements are erroneous.

6.b. The Consequences of Sycophancy

The harm from this tendency manifests at multiple levels.

At the level of reasoning, sycophancy degrades output quality. Studies have shown that models can exhibit a tendency to inflate estimates under suggestive framing.[18] In academic writing scenarios, models may expand a user's marginal claims into plausible-sounding paragraphs, provide fabricated citations, and gradually narrow

[14] Nielsen Norman Group. "Therapists' views on the use of questions in person-centred therapy." **nngroup.com**, 2020. https://www.nngroup.com/articles/info-tips-bad/

[15] Carl Rogers. "Therapists' views on the use of questions in person-centred therapy." **Journal of Consulting Psychology**, 1957. (Referenced via: https://www.tandfonline.com/doi/full/10.1080/03069885.2021.1900536)

[16] Ethan Perez et, al. "Discovering Language Model Behaviors with Model-Written Evaluations" **arXiv:2212.09251**, 2022. https://arxiv.org/abs/2212.09251

[17] Emmett Shear. Twitter/X post, 2025. https://x.com/eshear/status/1916879742256689582

[18] Samuel G.Z. Asher, et al. "Do Claude Code and Codex P-Hack? Sycophancy and Statistical Analysis in Large Language Models." GitHub repository, 2025. https://github.com/janetmalzahn/llm-phacking

opposition to a user's incorrect position.[19] The user receives a polished output without warning of the underlying issues.

At the psychological level, sustained sycophancy can erode cognitive autonomy. Constant affirmation creates a false sense of confirmation. Over time, this can lead to over-reliance on the model as an authoritative source, accompanied by a gradual atrophy of independent judgment. Alternatively, it can induce impostor syndrome, where users feel that the output they produce with LLM assistance is not genuinely their own work. The model's tendency to align with user statements creates a feedback loop without external grounding.

At the most serious level, the failure to question user statements has led to documented cases of harm. In one case, an individual asked an LLM for a salt substitute; the LLM suggested a compound used in industrial cleaning but not safe for consumption. The individual used it for several months and was later hospitalized with poisoning.[20] In another case, an individual with paranoid beliefs engaged with an LLM that validated these beliefs and generated a false assessment that the risk of delusion was near zero; the individual later committed a violent act.[21] In a separate case, an individual was drawn into a delusional framework through sustained interaction with an LLM that provided affirmations and instructions; the individual later died by suicide.[22] A similar lawsuit involving Character.AI is ongoing.[23] These cases share a common structure: the LLM accepted the user's statements as valid without probing underlying assumptions or questioning intent.

6.c. Reclaiming Genuine Regard

These harms share a common root. While appearing to be cases of an LLM providing incorrect information, a deeper issue is that the LLM never questioned what the user truly needed. Information transmitted from user to agent exists at three levels. The surface level is cognition: what the user knows and does not know. The second level is intent: what the user wants to achieve. The deepest level is self-awareness: whether the user understands what they truly need and whether they are aware of their own cognitive blind spots. Unconditional positive agreement addresses only the surface level, assuming cognition is complete, stated intent is true intent, and the user has full self-awareness.

A humanistic approach to agent design would invert these assumptions. Rather than merely validating user statements, an agent should actively participate in constructing user understanding: clarifying why the user wants to achieve something, identifying whether assumptions hold, and guiding the user to consider what they truly need. As noted by Aravind Srinivas, CEO of Perplexity, understanding user intent is the core

[19]Elizabeth Gibney. "Hey ChatGPT, write me a fictional paper: these LLMs are willing to commit academic fraud." **Nature**, 2026. https://www.nature.com/articles/d41586-026-00595-9

[20]Case report. "A Case of Bromism Influenced by Use of Artificial Intelligence." **Annals of Internal Medicine**, August 2025. https://www.acpjournals.org/doi/10.7326/aimcc.2024.1260

[21]Court filing: **Estate of Stein-Erik Soelberg v. OpenAI**. December 2025. https://www.courthousenews.com/wp-content/uploads/2025/12/ChatGPT-lawsuit-SF.pdf

[22]'Our Bond Is the Only Thing That's Real:' A New Lawsuit Alleges Google Gemini Drove a Man to Suicide. **TIME**, October 2025. https://time.com/7382406/gemini-suicide-lawsuit-death/

[23]CNN. "Character.AI and Google agree to settle lawsuits over teen mental health harms and suicides." January 2026. https://www.cnn.com/2026/01/07/business/character-ai-google-settle-teen-suicide-lawsuit

challenge for AI, and the future lies in agents that can complete tasks, not just provide links.[24] This involves asking probing questions, not as a form-filling exercise but as a collaborative process to establish a shared understanding of the task before proceeding. This is achievable through system prompts. The current tendency toward sycophancy reflects a design choice that prioritizes perceived agreeableness over substantive engagement with user needs.

7. Conclusion

This paper has proposed a framework for understanding Agent Experience as a tripartite system encompassing user-agent interaction, agent-environment interaction, and agent internal state management. It has argued that the management of context is the central technical challenge unifying the various approaches to AX, including Skills, MCP, RAG, memory mechanisms, and dynamic compression. It has examined the security and transparency challenges that arise when agents are granted significant system access and has identified the limitations of current permission models that transfer risk to users. Finally, it has addressed the humanistic concerns raised by the phenomenon of unconditional positive agreement, arguing that current design practices risk eroding user reasoning, cognitive autonomy, and in extreme cases, safety.

Agent Experience remains a nascent field. The technologies and design patterns discussed in this paper will continue to evolve. Future work should focus on developing robust, graded permission systems that balance capability with safety, creating frameworks for dynamic context management that preserve information quality, and refining interaction models that support genuine cognitive collaboration between users and agents. The goal of AX should not be merely to create agents that are easy to interact with, but to create agents that help users achieve their goals with clarity, autonomy, and safety.

[24] Aravind Srinivas. Transcript for Aravind Srinivas: Perplexity CEO on Future of AI, Search & the Internet | Lex Fridman Podcast #434, 2025. https://lexfridman.com/aravind-srinivas-transcript/