

TALEA: Connecting Guidance, Demonstration, and Selective Automation in a Spreadsheet Agent

ANONYMOUS AUTHOR(S)

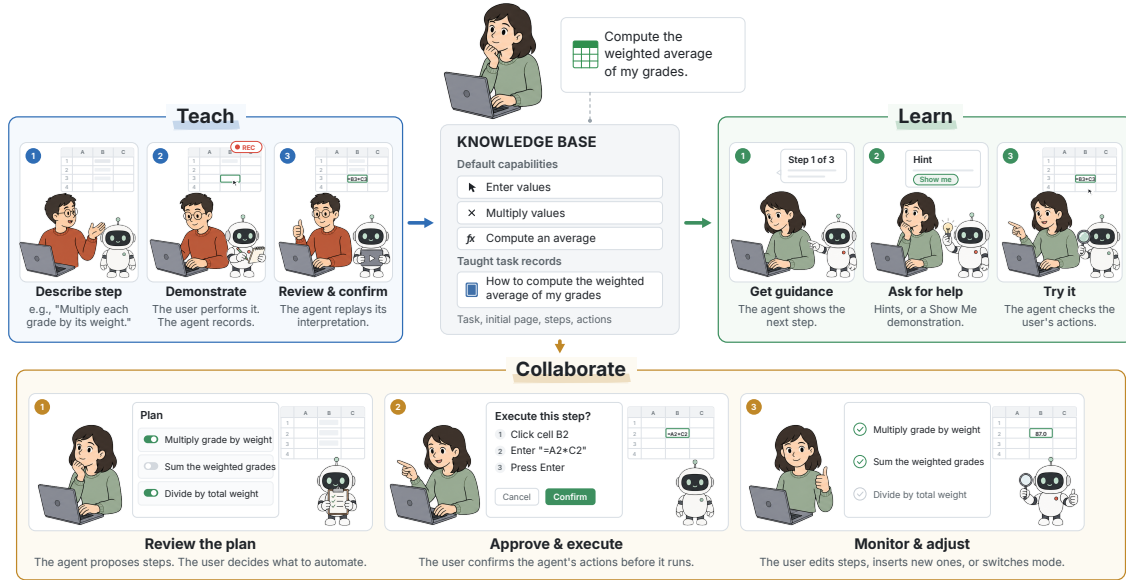


Fig. 1. TALEA's three interaction modalities, illustrated with a common spreadsheet example. In **TEACH**, users describe and demonstrate steps and review the agent's replay. **LEARN** provides progressive guidance, and **COLLABORATE** offers a plan whose steps users can edit and selectively delegate. The arrows show how task records connect the modalities: Teach stores a task description, initial page state, step descriptions, and accepted actions, which Learn and Collaborate retrieve as references for subsequent guidance and automation.

Browser agents can guide a user through a procedure, learn one from the user's demonstration, or execute selected steps on the user's behalf. When one interface supports all three, users must recognize what they should provide, what the agent has interpreted, and what they are approving. We present TALEA, a browser extension connecting **LEARN**, **TEACH**, and **COLLABORATE** through shared task records that carry taught procedures into guidance and automation. Users receive progressive guidance, teach through step descriptions and demonstrations followed by replay and acceptance, and authorize selected actions within editable plans. In an exploratory, facilitated within-subject study, 18 participants used all three modalities on distinct spreadsheet tasks. We report participants' experiences of control and assistance, together with difficulties in articulating procedural knowledge and coordinating execution. We discuss implications for supporting user participation in specifying procedures, reviewing generated interpretations, and sharing execution with an agent.

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

Manuscript submitted to ACM

CCS Concepts: • **Human-centered computing** → **Interaction paradigms; HCI design and evaluation methods**; *Interactive systems and tools*; • **Computing methodologies** → *Natural language processing*.

Additional Key Words and Phrases: web agents, human-agent interaction, interaction modalities, user control, LLM-powered agents, human-AI collaboration

ACM Reference Format:

Anonymous Author(s). 2027. TALEA: Connecting Guidance, Demonstration, and Selective Automation in a Spreadsheet Agent. In *Proceedings of CHI Conference on Human Factors in Computing Systems (CHI '27)*. ACM, New York, NY, USA, 29 pages. <https://doi.org/XXXXXX.XXXXXXX>

1 Introduction

A user who needs the weighted average of a set of grades can ask a browser agent to enter the formula, request instructions and enter the formula themselves, or demonstrate the calculation once so that the agent can perform it on later occasions. The three requests assign the user three different roles, following a procedure the agent supplies, providing the single example from which the agent must generalize, or reviewing a change the agent proposes. Browser agents derive actions from natural language requests and execute them through graphical interfaces [3, 14, 15, 22, 26, 27, 29, 31]. We study them in spreadsheets, where a procedure is assembled from formulas and recurs across documents [13]. Each intermediate value occupies a cell of its own, so a step of a procedure has both a stated form and a visible result. The value alone does not establish correctness, because an incorrect cell reference produces a plausible number and the error is seldom apparent from the result it leaves behind [32]. An agent operating on a spreadsheet therefore has reason to display a proposed change before performing it.

When one interface hosts all three activities, the user's role changes from one moment to the next. The interface may be waiting for a step description, a demonstration, an authorization, or a judgment about a result, and the request that opened the task does not determine which. Bansal et al. treat instructing an agent and verifying what it produces as two distinct demands the user carries through an interaction [4]. An interface that moves among these arrangements within a single task adds a third demand, recognizing which arrangement is in force at the present moment.

Earlier systems have combined some of these activities in a single representation, and they differ in what they give the user to review. CoScripter records browser actions as editable textual instructions that users can run automatically, step through, or interleave with manual action [19]. It replays the script as recorded and derives no interpretation for the user to examine. ALLOY makes such a derivation inspectable by holding the procedure in editable nodes, each carrying a subtask description together with the agent's instructions [20]. Pista presents the actions derived from a request one step at a time and scopes a correction to the step under review [33]. SUGILITE accepts a stated command alongside a demonstration of the same task and executes the task when the command recurs [21], without a per-step review of its interpretation. Teaching by demonstration has also been previewed in a deployed browser agent [30]. Across these research prototypes and deployed products, we are not aware of a system in which the same user states a step in natural language, demonstrates that step, and reviews a replay of the agent's interpretation of both inputs. Nor are we aware of one in which guidance and authorized execution are later generated from what the user accepted in that review. We know of no account of what such a combination asks of users who meet it for the first time.

The combination places two demands on the user, which we distinguish throughout the paper. *Specification* covers dividing a procedure into steps and wording each description so that it corresponds to the actions demonstrated under it. *Coordination* covers recognizing, at each moment, whether the interface expects a description, the actions to be

recorded, or a judgment of what the agent produced. The object of that judgment differs between two situations. When teaching, the user assesses the agent’s interpretation of a procedure they have already performed, and the question is whether the agent read the demonstration and its description as intended. When delegating, the user assesses a proposal about actions not yet performed, and the question is what those actions will do to the page. Our empirical question is how users specify procedural steps and coordinate their actions with the agent during initial use, while this division of labor is still unfamiliar.

We present TALEA, a browser extension for a spreadsheet website that offers the three activities as modalities the user selects (Figure 1). Every request is first checked against the operations configured for the agent and the tasks the user has already taught. A request the agent cannot serve opens a switch to Teach, which carries the request over as the task to be taught. In **LEARN**, the agent proposes the steps of a procedure one at a time for the user to perform. A hints panel lists the actions of the current step, *Show Me* replays them, and non-blocking feedback appears when an action departs from the expected sequence. In **TEACH**, the user writes a description of a step and performs it while the extension records. The agent derives its interpretation of those two inputs and replays it on the table, first restored to the state the demonstration began from, and the user then accepts the replay or supplies a textual correction. Ending the session stores the task description, that initial state, the step descriptions, and the accepted actions as one record, which Learn and Collaborate receive together with the current table. In **COLLABORATE**, the agent proposes an editable plan, and for each step the user decides whether to perform it or to request generated actions. Requested actions are displayed for authorization before any cell changes.

We implemented TALEA for a custom spreadsheet website and conducted an exploratory, facilitated within-subject study with 18 participants, each using all three modalities on three distinct spreadsheet tasks in a single session. We analyzed interaction logs, questionnaires, and interviews to examine how participants divided and worded the taught procedure, when they requested assistance or delegated execution, and how they read the controls for demonstration and automation.

This paper makes four contributions. First, it presents an interaction design that combines a teaching sequence of step description, demonstration, replay, and acceptance with progressive guidance and selective automation. A user who reaches the agent’s limit can therefore teach the missing procedure and reissue the request. Second, it introduces a record mechanism that retains the step as the user authored it, storing the description the user wrote together with the actions the user approved. Third, it provides an empirical account of specification and coordination during 18 participants’ initial use of the three modalities. Fourth, it derives design implications for specifying procedures, reviewing generated interpretations, and sharing execution with an agent.

Participants rated the system favorably, with a mean SUS score of 79.86, and completed 52 of 54 tasks. Eight participants reported difficulty formulating teaching steps, and the descriptions they submitted varied in how they divided the calculation. The facilitator observed that participants read the activity indicator beside a submitted description as a sign of ongoing generation, and they consequently waited during part of the interval in which they were expected to demonstrate. The indicator and the expectation of a demonstration reached the user through the same channel, the panel beside the spreadsheet. Some participants found the step-card controls in Collaborate unclear at first, the automation switch in particular, and most reported that their meaning became clear after use. We distinguish difficulty in formulating step descriptions from uncertainty about when to demonstrate or authorize execution, because the

two admit different remedies. The first calls for showing a step's recorded and generated actions beside its description before the step is stored. The second calls for a separate channel that announces the act expected of the user.¹

2 Related Work

2.1 Specifying and Reviewing Agent Actions

An agent operating an interface on the user's behalf writes into the user's own document, and its actions can depart from what the user intended. Zhang et al. collected users' accounts of browsing agents producing unintended outcomes, among them wrong booking dates and fabricated contact details [39]. Design guidance for generative systems accordingly asks that what a system produces be presented for review [38]. Bansal et al. [4] place verification of agent behavior among the challenges of human-agent communication, alongside communicating goals, capabilities, and outcomes. An interface that divides a task between the two parties must therefore let the user specify that division and indicate which contribution is expected at each moment. Both requirements have counterparts in research on people working together. Clark and Brennan describe grounding as establishing that a contribution has been registered [8], Dourish and Bellotti locate coordinating information in the object both parties act on [10], and Suchman describes a plan as a resource elaborated in action [36].

Amershi et al. [2] address the first by recommending that a system communicate what it can do, and the second by recommending that it show information suited to the current activity. Cila [7] separates the commitments within such an arrangement, among them the autonomy the agent holds and when its actions require approval. Feng et al. treat autonomy in the same separable way, as a level set for an interaction [11].

One arrangement makes the division explicit in a plan whose steps the user reads, edits, and assigns before execution. Cocoa, developed by Feng et al. [12] for researchers in a document editor, runs the steps of a plan in order, and each produces text the user has to judge. Later steps draw on earlier outputs, so an unexamined result propagates into what follows. Users edit the step descriptions, assign each step to themselves or to the agent, and modify an intermediate output before later steps consume it. During execution, the step assigned to the user is highlighted when input is required. Magentic-UI, developed by Mozannar et al. [24] for browser tasks, likewise presents an editable plan, and adds controls that transfer browser operation between the two parties while the task runs.

A second arrangement leaves the division implicit and lets the user redirect an execution in progress. CowPilot, developed by Huq et al. [16] for browser tasks, displays each proposed action beside its highlighted target element before performing it. Users can pause or reject that action, operate the browser themselves, and return control. The manual actions are appended to the history from which the next action is generated.

Collaborate divides the work in the first manner, since the user revises a proposed plan and decides for each step whether to perform it or to request generated actions. As in CowPilot, a step the user performs leaves the sheet in the state the agent generates the next step's actions from (Section 4). An execution already under way cannot be redirected in Collaborate, since the user authorizes a list of actions through a separate confirmation before the agent performs them. In Cocoa, Magentic-UI, and CowPilot the agent proposes the procedure being divided, and the user revises that proposal. The procedures treated below originate with the user.

¹Project website: <https://talea.backtothehuman.com/>.

2.2 Demonstrating, Reviewing, and Reusing Procedures

Programming by demonstration enables users to specify a program through examples of the actions it should perform [9, 18, 25]. Its appeal for people who do not program is that performing the procedure supplies the specification. Its difficulty is that the system must generalize from the one or few executions it was given, and that generalization is not visible in the demonstration. Reviewing a series of such systems, Lau [17] argues that teaching and supervising a demonstrated procedure can cost more than performing the task. The balance depends on whether users can read and correct what was inferred.

Leshed et al. [19] developed CoScripter to record browser actions and render them as natural language instructions that users read and edit directly. The recording, the displayed procedure, and the editable source are therefore one object. Users can run a script automatically, step through it to learn an unfamiliar procedure, and interleave manual steps with automated ones. One object serving those three uses is the precedent TALEA builds on. In a survey of users who had abandoned the tool, three respondents reported coordination problems where manual and automated actions alternated during execution. That difficulty of turn-taking is separate from the difficulty of specifying a procedure.

Recent systems offer two channels for correcting a generated procedure. A user amends the demonstrated example so the system derives the procedure again, or edits the derived representation directly. In MIWA, Chen et al. [6] link the description of each generated step to the webpage elements it acts on. Users execute one step at a time, inspect the results, and correct individual actions in the trace, from which the script is regenerated. In ALLOY, Li et al. [20] represent a demonstrated procedure as a graph whose nodes hold a subtask description with its agent instructions. Users edit those instructions and connections, so revision is carried out in the system's representation, where the cost Lau describes attaches to reading and correcting the agent's text.

Pairing a stated command with a demonstration has a precedent in SUGILITE, which performs a taught task when the accompanying command is issued again [21]. In Teach, TALEA takes a description and a demonstration for each step, and the user reviews the agent's interpretation of that pair as a replay. A correction is written in text that returns to the model with the previous generation. The accepted pair holds what an ALLOY node holds, a subtask description with the actions for it, and TALEA derives it by generation from the user's two inputs.

Each system retains a completed procedure in a form that determines what a later request can draw on. CoScripter keeps recorded scripts in a shared repository from which other people run them [19]. ALLOY adapts a saved workflow to a related request through placeholders the request fills with task-specific values [20]. Magentic-UI synthesizes a plan from the trace of a completed execution and attaches it to a subsequent request [24]. Further work induces reusable workflows from earlier executions for the agent to draw on [23, 37]. The procedure carried forward is in each case derived from an execution. A TALEA task record holds the description the user wrote for each step together with the interpretation accepted for it. When Learn generates guidance or Collaborate generates actions to authorize, the agent receives that record with the table currently displayed.

2.3 Spreadsheet Assistance and End-User Programming

Spreadsheets are an environment where people who do not program already build and run procedures of their own. Those procedures are assembled from formulas and repeat across documents that share a structure and differ in their values [13]. Checking such a procedure involves examining the formula that produced a value, since a mistaken cell reference leaves a plausible number behind [32]. Gulwani et al. [13] developed techniques through which users specify a spreadsheet transformation by supplying examples of the outputs they want, and a corrected output becomes a further

example. Where programming by demonstration takes examples of the actions that produce a result, this line of work takes examples of the result itself.

In the WYSIWYT methodology for end-user software engineering, Rothermel et al. [32] make the individual computed value the unit of validation. Users mark a displayed value as correct, and that mark propagates through the cells the value depends on. The interface reports how much of the computation the accumulated marks have covered. That unit, one computed value the user judges from the screen, carries over to an agent operating the spreadsheet. TALEA records such a judgment when a user accepts a replay, and it provides no equivalent report of coverage. The evidence available for the judgment also differs, since the user examines the actions performed and the values they leave behind.

In Pista, an Excel extension developed by Sabouri et al. [33], reviewing an agent step means inspecting the applied formula, the affected data ranges, and an explanation. Users can request a correction scoped to the step under review. Because a correction can change computational dependencies, Pista keeps the original execution and branches with regenerated subsequent steps for comparison. TALEA shares this per-step review of agent actions inside a spreadsheet. The object under review differs, since Pista presents actions the agent derived from a request, while Teach presents the agent's interpretation of a procedure the user described and demonstrated.

3 Design Rationale

A spreadsheet procedure can be delegated to the agent, performed by the user under the agent's instructions, or demonstrated for later reuse. Each system discussed in Section 2 fixes one part of this arrangement in advance and places the other outside the user's reach. Cocoa states in its plan who performs each step, while the plan itself comes from the agent [12]. CowPilot offers each proposed action for review, while the division of work between the parties stays implicit [16]. Initiative passes between the two parties several times within a single task [1]. We organize the controls of TALEA around who specifies the procedure, who executes its actions, and when the user reviews what the agent has produced.

The agent contributes a product of its own at each exchange, and any divergence between that product and what the user meant reaches the user only where the interface presents it before it takes effect. Each modality fixes an answer to the first two questions and supplies the review that answer calls for (Table 1). Those assignments remain revisable within a task, as when a user following guidance in Learn hands one step to the agent. Teach produces the record that connects the three modalities, which Learn and Collaborate consult when responding to a later request.

Table 1. User input, review, execution, and use of task records in TALEA. A task record contains a task description, initial page state, step descriptions, and accepted actions.

Modality	User input	Review and execution	Task record
LEARN	Task request and spreadsheet actions.	User follows generated instructions and can request hints or an agent demonstration.	Matched record informs guidance and demonstrations.
TEACH	Task and step descriptions, demonstrations, and corrections.	Agent replays generated actions; user accepts each interpretation or requests a correction.	Accepted steps and actions are saved when the user ends the task.
COLLABORATE	Task request, plan edits, and execution choices.	User reviews the plan and proposed actions, then acts manually or authorizes agent execution.	Matched record informs action generation for the current page.

3.1 Specifying and Reviewing a Taught Step

A demonstration of a weighted-average calculation gives the agent a sequence of selections and entered formulas. It leaves open which properties of those actions are essential, since a range may have been chosen because it holds the grades or because it occupies the third column [18]. A description of the same step states the intended operation and leaves the cells and the formula undetermined. In Teach the user supplies both for each step.

From these two inputs the agent generates its interpretation of the step, an ordered sequence of actions on the table. TALEA presents that sequence as a replay on the spreadsheet, touching the cells and entering the values the agent inferred. Earlier systems render such a sequence for reading, as editable instructions, workflow nodes, or formulas with their affected ranges [19, 20, 33]. The table first returns to the state the demonstration began from, so the replay shows the agent’s interpretation alone. Users either accept the replay or provide a textual correction, which returns a regenerated attempt to inspect and accept or correct again.

Acceptance commits the user to the displayed interpretation, and that step with its description is the unit a later request draws on. The judgment covers a replay on the table just worked on, and whether the same actions suit a sheet whose values or cell positions differ stays open.

3.2 Receiving Guidance and Authorizing Execution

Learn and Collaborate both assign execution at the granularity of a single step. A high degree of automation and a high degree of user control are separate dimensions of a design [35], and in both modalities the act that puts an operation into effect is the user’s.

In Learn the interface displays one step at a time. A single step is small enough for the user to check the agent’s proposal before acting. Assistance inside a step escalates in two stages, since *Hints* names the actions the step requires and leaves their performance to the user, while *Show Me*, offered with the last hint, has the agent perform them. A user who cannot carry out a step therefore hands it to the agent and continues with the steps that follow.

Guidance advances as the agent compares the user’s clicks and inputs with the actions it generated for the current step. An action outside that sequence can come from a valid alternative route as well as from an error. Deviations are therefore reported through non-blocking feedback that leaves the spreadsheet available for user action.

A task can mix a repetitive step with one whose formula the user wants to compose, so one decision about automation would suit only part of it. Authorization in Collaborate is therefore given for individual steps. A request in Learn and each step of a Collaborate plan are judged against the operations the agent holds and the tasks this user has taught (Section 4.2). That judgment determines whether the step can be delegated and which stored procedure informs the response. The plan comes from one request and can omit what the user knows, so the user reads it first and can delete, insert, or reword a step. Rewording a step withdraws it from automation, since the judgment of availability was made for the text the plan carried, and the user then performs that step or teaches it.

Actions are requested for a step whose description the user has already approved, and their descriptions appear in the step’s card before any cell changes. A plan can name the right operations while the actions generated for one step address the wrong cells, so the two are reviewed separately. Reviewing the actions is the judgment of a proposed change that Section 1 distinguishes from the judgment of an interpretation during teaching. Cila’s separation of an agent’s autonomy from the moments requiring approval [7] applies here, so TALEA keeps the control that revises what will be done apart from the execution authorization that permits it.

3.3 Retaining Procedures for Guidance and Automation

When the user ends a teaching session, TALEA stores the step descriptions and the accepted actions with the task description and the table as it stood when the demonstration began. The record keeps each stated operation paired with the execution accepted for it, so a later request gives the agent both the intended operation and one worked example of it. That initial state records what the procedure was demonstrated against, so the agent can relate the demonstrated targets to the table the user then has open.

Learn and Collaborate receive the matched record together with the current page, so the agent can use the demonstrated procedure as a reference when it generates actions for the requested task (Figure 1). Some requests match nothing the agent holds, and TALEA answers those with a transition into Teach that carries the request over as the task to be taught.

Review during reuse draws on different material from the acceptance given during teaching, since the user examines whether the generated actions use the intended cells, values, and operations. Section 4 describes the interfaces, prompts, and stored representations through which TALEA provides these arrangements.

4 TALEA

TALEA is a Chrome browser extension operating on a webpage it does not control. Observation and action both pass through scripts injected into the active tab, which read and manipulate the page’s document object model (DOM). An action is therefore identified by the page element it touches.

4.1 Architecture Overview

A recorder observes the user’s clicks and committed inputs, a replayer performs generated actions one at a time, and small components draw feedback into the page. In Learn, the recorder’s events are compared against the sequence the agent expects for the current step. In Teach, the same events form the demonstration from which the agent generates its interpretation.

The extension operates on a custom spreadsheet website we built for the study. A page whose structure we control gives every cell a stable identifier for recording and replay, and it is small enough to supply to a model as context. Users select a modality and enter a request in the panel beside the spreadsheet (Figure 2).

Capability assessment returns a match against a supplied list of entries, and Collaborate planning returns the steps of a task in prose. Learn guidance, and action generation in Teach and Collaborate, must produce formulas, selectors, and ordered action sequences valid for the page in its current state. These two groups of calls to large language models (LLMs) use different configurations. The calls returning matches and prose use GPT-5-mini with low reasoning effort, and the calls producing formulas and actions use GPT-5 with medium reasoning effort. Appendix A reports the exploratory experiments behind the first configuration and the additional manual checks behind the second.

4.2 Shared Knowledge and Capability Assessment

The knowledge base holds the entries every request is judged against, and it contains two kinds. A predefined operation is a short natural language description of something the agent can already do on this spreadsheet. Examples are clicking a cell, writing a value into it, and applying a sum formula to a range. We authored these entries for the study deployment, covering the formulas the spreadsheet website implements and the basic cell interactions the page supports. They carry no stored actions, so a request matching one is served by generating actions from the request and the current table.

The second kind of entry is a task record, written by the user in Teach (Section 4.4). A record holds the task description, the initial DOM of the table, and an ordered list of steps. Each step contains its natural language description and the agent actions accepted during teaching. Each action specifies its type, a click or an input, a selector for the target element, and a value for inputs. Those selectors identify cells of the page as it stood during the demonstration, so the stored initial DOM is supplied alongside the current table. When a request matches a task record, Learn guidance and Collaborate action generation receive that record, with its step descriptions and accepted actions, together with the current table. Task records persist in browser local storage within the user's installation.

For capability assessment, an LLM receives the request with the lists of predefined operations and user-taught tasks, and returns either a matching entry or a judgment that the request is unavailable. The prompt limits matching to the supplied entries, so availability reports what the knowledge base contains as the assessment reads it. A knowledge base holding separate entries for sums, averages, and weighted sums therefore reports a request for a weighted average as unavailable. An assessment can also refuse a request the agent could have served, or match an entry whose generated steps do not suit the sheet in front of the user. An available request produces generated steps in Learn and an editable plan in Collaborate, and an unavailable one opens a dialog offering a switch to Teach (Section 4.7). The panel reports whether the request can be served without naming the entry that matched it.

4.3 LEARN Interface

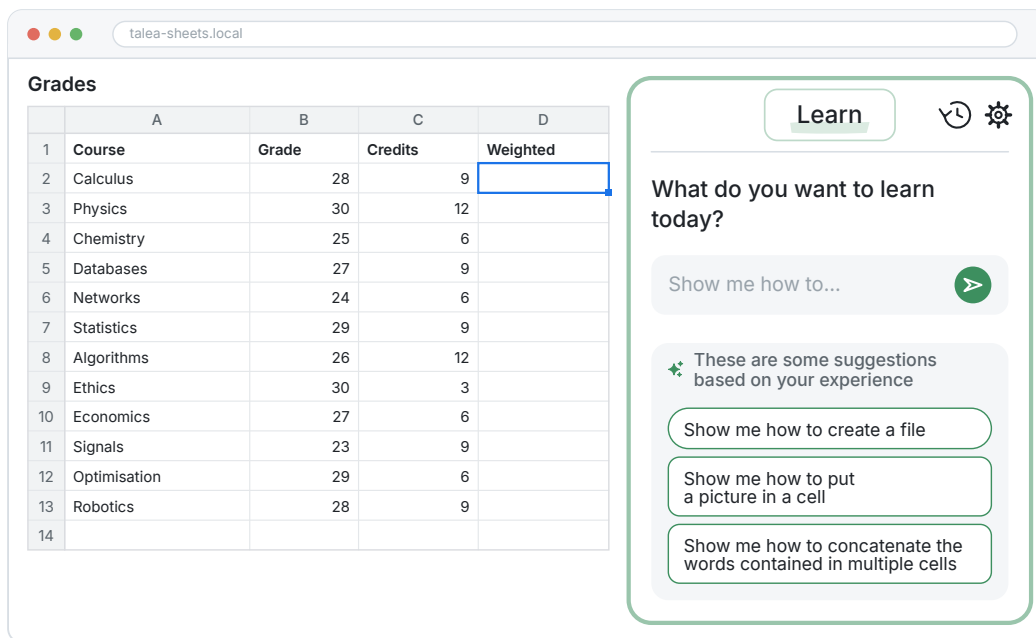


Fig. 2. System interface, with the spreadsheet on the left and extension panel on the right. The panel shows the modality selector (Learn), a task input field, and example requests offered as shortcuts. These examples illustrate the input field and are independent of the knowledge base. The corresponding Teach entry interface lists recently taught tasks.

In Learn, users submit a task description for capability assessment (Figure 3a). A generated step is one instruction naming the cell to act on, with the formula for that cell beneath it (Figure 3b). Each step also carries the clicks and inputs that perform it, and the next becomes visible when the current one is complete.

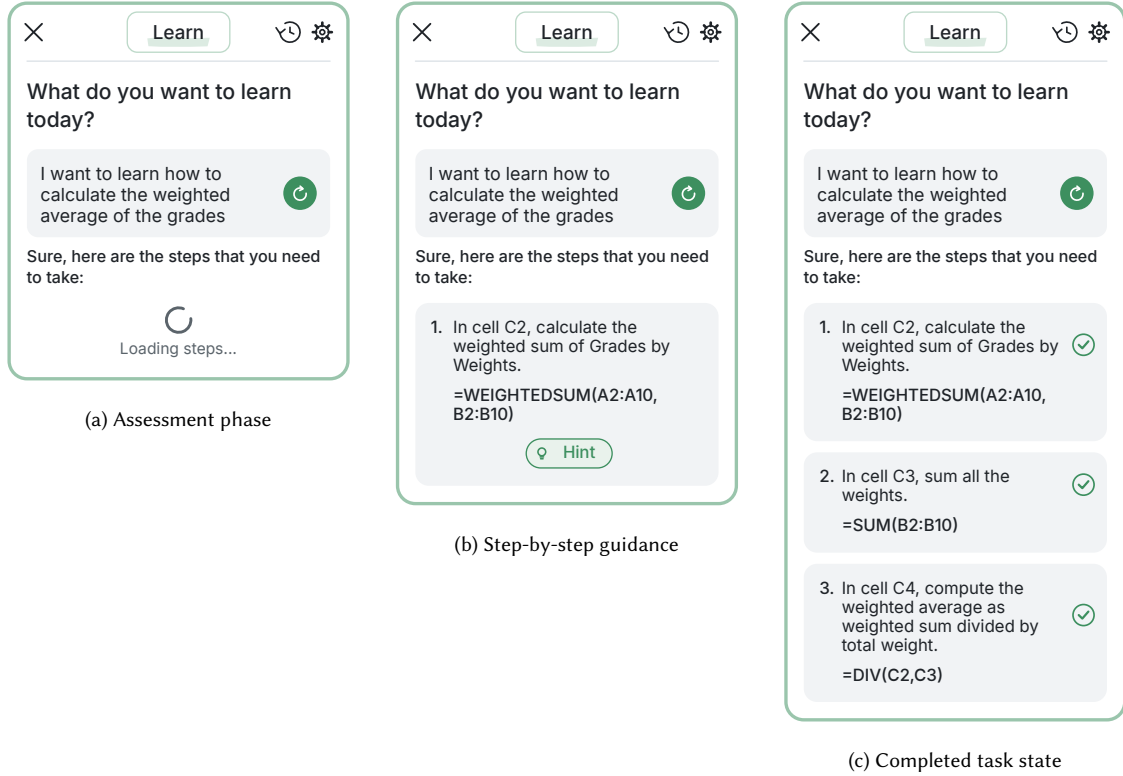


Fig. 3. LEARN modality interface. (a) The request is assessed against listed capabilities. (b) Steps are revealed progressively, with action-level hints available. (c) Completion indicators track progress through the generated sequence.

As users perform a step, the agent compares their clicks and committed inputs against the next action pending in that step's sequence. An input is committed when it is settled with Enter or a change of focus. An input matches when the cell and the committed value are both those the sequence specifies. A result obtained through a different formula, or written into another cell, therefore counts as a deviation. TALEA does not independently validate the result the spreadsheet holds. A mismatch triggers a popup naming the action expected (Figure 6c). The popup does not block the page, so users resynchronize by performing the actions the sequence still expects.

Users can also request action-level guidance through *Hints*, which lists the actions of the current step. *Show Me* appears with the last of those actions and replays them on the sheet with their targets highlighted (Figure 6d–e). The values the replay enters remain in the sheet, and the step is marked complete when the replay finishes, so the user continues from the state the agent produced.

4.4 TEACH Interface

In Teach, users submit a task description, and capability assessment establishes whether the agent already holds the requested procedure (Figure 4). For a previously taught task, a dialog offers to replace the existing record through re-teaching or to return to the entry interface (Figure 7c). For a predefined operation, the dialog offers only a close button, since a record taught under the same description would be matched in place of the operation the agent already performs. A request matching no existing entry proceeds to teaching, and the system saves the initial table DOM before demonstration begins.

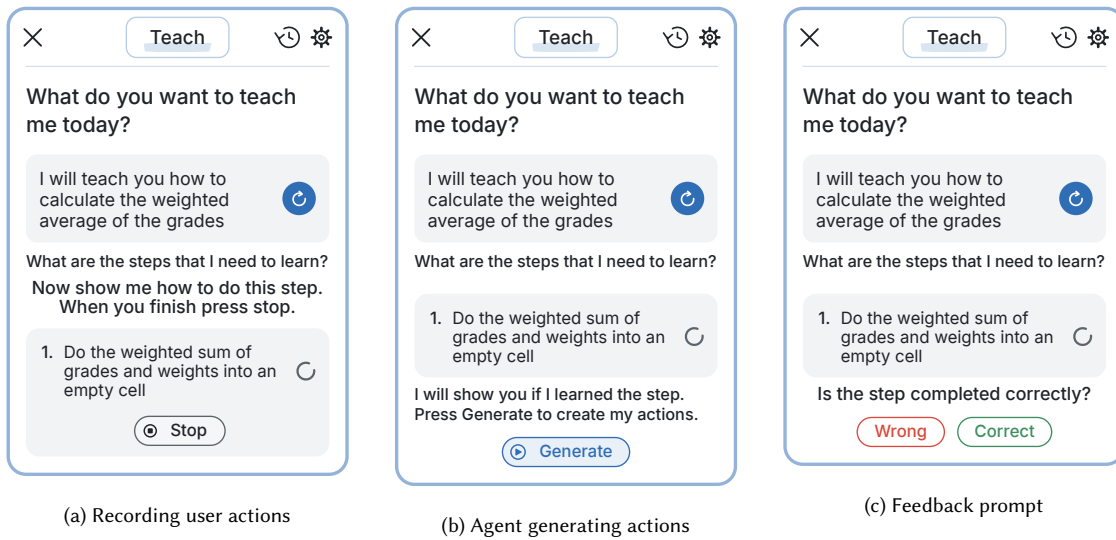


Fig. 4. TEACH modality interface. (a) The user describes a step and then demonstrates its actions while the agent records. (b) The agent generates actions from the step description, demonstration, and page context. (c) After reviewing the replay, the user accepts the result or provides a textual correction.

Users describe each step and then demonstrate it. Submitting the description places it in a card with a *Stop* button and a message asking the user to show how the step is done. An activity indicator beside the card marks the step as in progress while the recorder captures actions on the spreadsheet. After users stop recording and request action generation, the extension restores the table to its state before the demonstration. The LLM receives the task description, step description, table DOM, and recorded demonstration. Users confirm the start of the replay through a dialog in the page, and can stop it while it runs.

After reviewing the replay, users accept the generated actions or provide a textual correction. Acceptance fixes the content of the step, and the accepted actions become its stored representation. Later generation receives them as the reference for the task. The recorded demonstration serves generation only and is not kept in the record. For regeneration, the LLM also receives the correction and the preceding generated actions. A step leaves review only when a replay is accepted, and the interface offers no control for abandoning a step.

Clicking *End Task* saves the record and adds the task to the knowledge base (Section 4.2). The panel returns to the Teach entry interface, where the saved task appears among the recent tasks. The interface does not display the steps and actions the record holds. A record replaced through re-teaching is not retained.

4.5 COLLABORATE Interface

In Collaborate, users submit a goal, such as calculating average weekly spending (Figure 5). One LLM call generates the steps of the task, and a second assesses those steps against the predefined operations and user-taught tasks. A single call performing both tended to decompose an unavailable operation into available ones, so planning and assessment were separated during development. A plan can therefore contain steps the agent cannot perform, which remain in it for the user to perform or to teach.

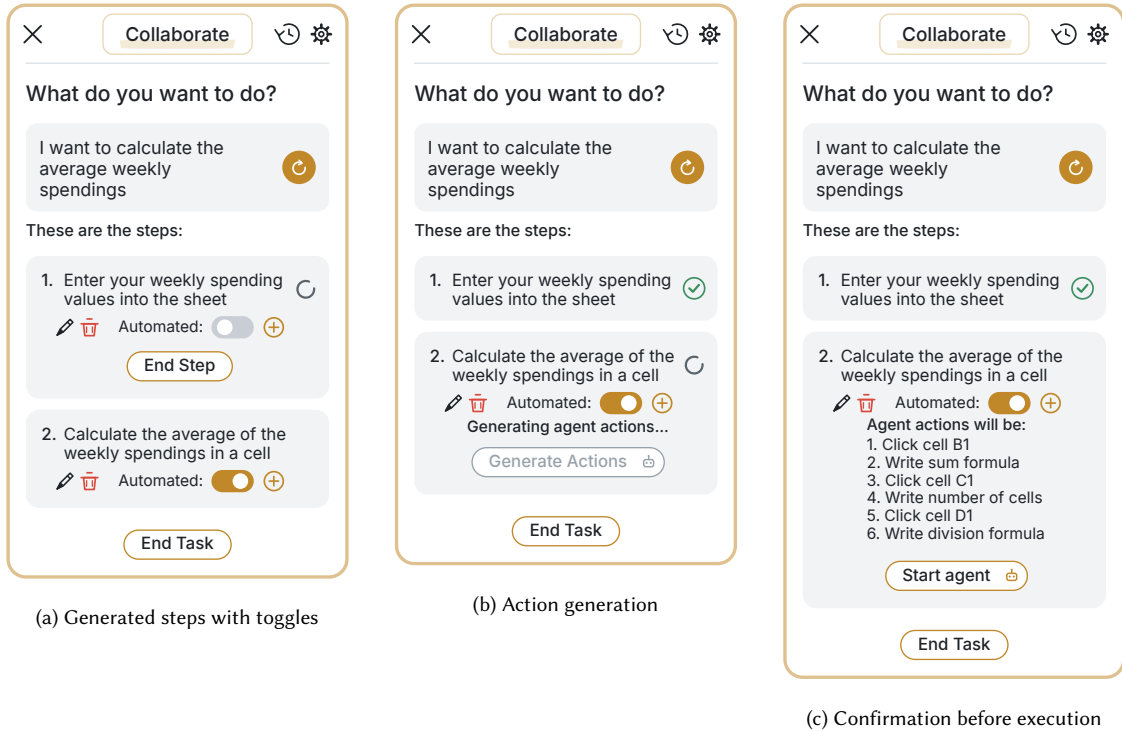


Fig. 5. COLLABORATE modality interface. (a) The agent proposes steps for the request and current table, then assesses their availability for automation. (b) Actions are generated for a selected step. (c) The user reviews proposed actions and confirms before execution.

Steps appear in cards with a switch that both reports the step's availability for automation and requests it. Turning it on reveals a control for generating that step's actions, and operating it on an unavailable step offers to teach that step (Section 4.7). Action generation uses the current table DOM together with the matched operation or task record (Section 4.2). The card lists the generated actions as short descriptions naming the target cell and the operation of each, without the formula text. Execution then requires two further acts by the user, pressing *Start agent* and confirming in a dialog drawn in the page.

A step the agent executes is marked complete when its replay finishes. A step the user performs carries an *End Step* control, which the user presses after acting on the spreadsheet. The cards stay revisable while the plan runs. Editing a description marks that step unavailable for automation, and the interface offers no control for assessing the new text.

4.6 Visual Feedback

We draw the messages about an action in progress inside the page (Figure 6). This assumes that a user performing or watching actions attends to the spreadsheet, while the panel beside it holds their attention less. The components drawn there are the status and error popups of Learn, the hints panel with its *Show Me* control, and the dialogs confirming a replay in Teach and an execution in Collaborate. The error popup names the cell it concerns and, for an input, the value expected there. The step-completion and error popups disappear after a few seconds, or sooner on a click anywhere in the page, while the popup announcing a completed task carries a close button. The confirmation dialogs persist until answered, since the replay or execution they gate waits on that answer.

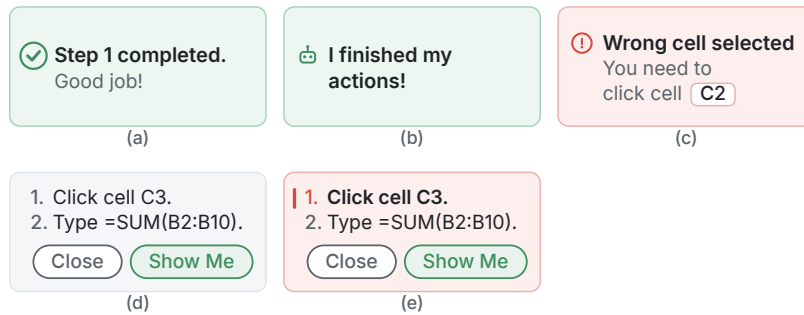


Fig. 6. Visual feedback components injected into the webpage. (a-b) Status notifications for a completed step and a finished sequence of agent actions. (c) A popup naming the cell expected after an unexpected action. (d) The hints panel for the current step, with *Show Me*. (e) The same panel with the action concerned by a deviation marked.

4.7 Cross-Modality Switching

When capability assessment reports a task or a plan step as unavailable, users can select a dialog option to switch to Teach (Figure 7a–b). The requested task or step is carried into Teach as the task description. When that teaching session ends, the panel returns to the Teach entry interface, so the user selects a modality and submits the request once more. The resubmitted request is then assessed against a knowledge base that holds the new record.

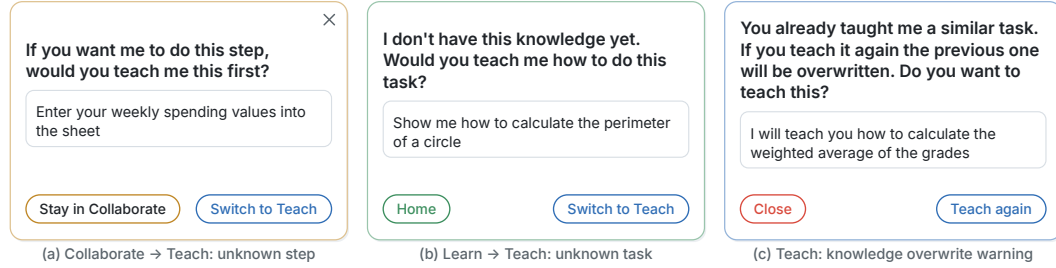


Fig. 7. Dialogs for transitions into Teach from Collaborate (a) and Learn (b), and an overlap warning for an existing user-taught task (c). These dialogs expose availability judgments and let users choose whether to enter or repeat a teaching interaction.

5 Evaluation

Each modality fixes one answer to the three questions of Section 3, who specifies the procedure, who executes its actions, and when the user reviews what the agent has produced. We conducted an exploratory mixed-methods study with 18 participants to establish what each answer asks of a person meeting it for the first time. Each participant used all three modalities, each paired with a distinct task, and we collected behavioral logs, accounts of interaction difficulties, and ratings of usability and control. Sections 5.2.1 and 5.2.2 report what participants taught and what assistance they requested. Section 5.2.3 locates the time within a teaching cycle, and Sections 5.2.4 and 5.2.5 report completion, ease, and the questionnaire scales.

5.1 Study Design

Participants. We recruited 18 participants (Table 2), 17 university students and one working professional in training design. We screened for basic digital literacy and familiarity with simple formula creation, so that difficulty composing a formula would not be confused with difficulty specifying a procedure. All signed informed consent forms allowing screen and audio recording.

Tasks and their pairing with the modalities. Each modality was assigned one task, the same task for every participant. In Learn, participants computed the area of a triangle from pre-filled base and height values, and in Collaborate they calculated the average of their weekly spending starting from an empty table. In Teach, participants taught the weighted average of university grades over a table already holding the grades and their weights, a procedure of three dependent calculations.

Learn received the shortest task, since guidance is observable on a procedure a participant can carry out unaided once the steps are displayed. Teach received the weighted average, a procedure of several distinct operations where the participant must decide where one step ends and the next begins. Collaborate received the task beginning from an empty table, which holds both work a participant may prefer to do personally and a calculation worth delegating. A session of about an hour held one task per modality. Each modality is therefore tied to a single task, and task and modality effects are confounded in the duration and ease measures. Participants used the three modalities in counterbalanced order, with two to four participants in each of the six orderings, too few to establish the absence of practice, carryover, or position effects.

Measures. The Single Ease Question (SEQ) captured task ease on a 1–7 scale after each task, and the System Usability Scale (SUS) measured overall perceived usability once, after all three [5]. Neither instrument separates the parts of

Table 2. Overview of the 18 participants, including demographic information, occupation, self-reported spreadsheet usage frequency, and technology confidence level.

ID (Gender, Age)	Occupation	Spreadsheet Frequency	Technology Confidence	AI Agent Experience
P1 (M, 40)	Training Design Expert	●●●○	●●●○	No
P2 (F, 20)	Student, Pharmacy	●●●●	●●○○	No
P3 (M, 24)	Student, Cybersecurity	●○○○	●●○○	Yes
P4 (M, 21)	Student, Physics	●●●○	●●●●	No
P5 (F, 20)	Student, Physics	●●●●	●●●○	No
P6 (M, 27)	Student, Building Eng.	●●●●	●●○○	No
P7 (F, 21)	Student, Physics	●●●●	●●●●	No
P8 (F, 24)	Student, Sociology	●●●○	●●○○	No
P9 (M, 20)	Medical Student	●○○○	●●●●	No
P10 (F, 23)	Medical Student	●●●●	●●○○	No
P11 (M, 21)	Student, Software Eng.	●●●●	●●○○	No
P12 (F, 20)	Student, Languages	●●●○	●●○○	No
P13 (M, 25)	PhD Student, Economics	●●●○	●●●●	Yes
P14 (M, 23)	Student, Environ. Chemistry	●●●●	●●●○	No
P15 (F, 20)	Student, CFT	●●●○	●○○○	No
P16 (M, 25)	PhD Student, Computer Sci.	●●●●	●●●○	No
P17 (F, 22)	Student, Physics	●○○○	●●●●	No
P18 (F, 22)	Student, Philosophy	●○○○	●○○○	No

Frequency: ●○○○ Never ●●○○ 1–2/wk ●●●○ 2–3/wk ●●●○ 4–5/wk ●●●● Daily
Confidence: ●○○○ Very Low ●●○○ Low ●●●○ Mid ●●●○ High ●●●● Very High

an interaction that the three modalities assign differently. We therefore developed five-point agreement scales for controllability (4 items), trust (5 items), and transparency (7 items), with items derived from the controls each modality provides (Appendix B). The controllability items ask about interrupting a replay, dismissing the agent, guiding it during teaching, and perceived control.

Four items were optional, since they concern components a participant might not encounter. We used available responses for item-level descriptions and complete cases within each scale for composite means and Cronbach’s alpha (controllability $n = 16$, $\alpha = 0.82$; trust $n = 15$, $\alpha = 0.88$; transparency $n = 10$, $\alpha = 0.64$). Three of the four optional items belong to the transparency scale, whose alpha is below conventional thresholds, so we treat its seven items individually.

Logged data. The extension timestamped the interface events of the three modalities, among them the submission of each step description, which starts the recording, the moment the participant stops the recorder, requests for action generation, and the authorization and acceptance of a replay. It also logged corrections, requests for *Hints* and *Show Me*, openings of the plan controls, the end of each task, and page clicks and committed inputs. The event stream does not mark whether an action came from the participant or from a replay. We therefore attribute actions by the interval they fall in (Section 5.2.3). Display of the status and error popups produces no event, so the logs do not show how many participants saw one.

Procedure. Sessions lasted approximately 60 minutes on the experimenter’s laptop, with the extension already configured. An introduction and a tutorial task familiarized participants with the spreadsheet website and with invoking and stopping the agent. Participants then completed the three tasks while thinking aloud and answered the SEQ after each one. Participants and facilitator afterwards reviewed selected agent actions together, before the questionnaires and the final interview.

Facilitation. The facilitator intervened when an agent or interface error blocked a participant, and when a participant stopped making progress, including with the formulation of teaching steps. An intervention consisted of suggesting a division of the procedure into steps or recovering from the error. The observations therefore describe facilitated use, the recorded durations include assistance and think-aloud activity, and the study record holds no per-participant count of interventions.

Quantitative analysis. We matched timing and questionnaire records by participant identifier, retaining the same 18 participants throughout. One further session is excluded from every analysis, a preliminary case with incomplete task data whose log file was also unusable. The ease ratings are ordinal and the durations are skewed at this sample size, so we used nonparametric procedures. Friedman tests and Kendall’s W summarize differences across the three assigned task–modality combinations. Pairwise Wilcoxon signed-rank tests used exact sign permutations with average ranks for ties, and we report Bonferroni-adjusted p -values (p_{adj}) for each family of three task contrasts against .05.

Qualitative analysis. One researcher worked from the interview responses, the think-aloud and facilitator notes, and the screen and audio recordings. The interview follows a fixed set of questions per modality. The counts below are therefore tallies of answers to those questions, produced without an inductive coding scheme or a second coder. The Learn questions were asked only of participants who had met the component concerned. Each observation was compared with that participant’s logs, so the episodes below rest on timestamped events and recorded inputs.

5.2 Results

5.2.1 Specifying and Reviewing Taught Steps. Teaching the weighted average required each participant to describe the coming step before demonstrating anything. No worked example was available, the interface stated no expected size for a description, and the division of the calculations was left to the participant. Eight of the 18 participants reported difficulty writing the steps and said they had needed facilitator guidance to decide what to include or how to divide the task. Five described the process as confusing and placed that confusion at the beginning of the task. The two counts come from separate questions and may overlap.

Participants accepted a median of 3 steps per task (range 1–5), and the division of the calculations varied. P5 and P6 each submitted the whole procedure as a single step description. P6 wrote two numbered items covering the weighted sum, the sum of the weights, and the division of the two results. P17 submitted a separate description for each calculation, naming the function to apply and the cells holding its arguments, and a fourth step asking the agent to write a completion message.

The actions captured while recording P6’s single step contained one committed input, a weighted-sum formula over the grade and weight columns. The description named three calculations while the demonstration performed the first of them. The two inputs to generation covered different extents of the same procedure. P6 accepted the replay and ended the task, and the logs do not record how generation reconciled the two inputs.

Participants reviewed 57 replays, accepting 54 and rejecting 3, an acceptance rate of 94.7%. The three rejections came from two sessions, P8 twice and P15 once. The path through rejection, textual correction, and regeneration was

therefore exercised in two of 18 sessions. Each correction restated the formula the step required and the cells it should use. Acceptance records approval of the displayed replay and does not verify the correctness of the generated actions.

Submitting a step description starts the recording, and the panel shows the step with a message asking the user to demonstrate it and an activity indicator marking it in progress (Section 4.4). The facilitator noted that participants read this indicator as the agent still generating a response and waited during part of the interval in which they were expected to demonstrate. No event marks the start of a demonstration, so this observation derives from the facilitator's notes. Most participants reported greater confidence after the first step, which places the confusion they described at the opening of the task.

Ending the task writes the record to storage without a statement of what was stored, and post-task feedback included a request for a message confirming that Teach had saved the task.

5.2.2 Requesting Guidance and Authorizing Execution. Learn displays one generated step at a time and holds its more detailed assistance until a participant requests it. Five participants (P1, P2, P11, P15, and P16) opened the hints panel, and three of them (P2, P11, and P16) went on to request a *Show Me* demonstration. P2 opened the panel three times and requested the demonstration each time, and P16 did so twice. All five described the hints as useful, and the other thirteen participants completed the task without opening the hints panel.

The plan Collaborate proposed for the weekly spending task had two steps, entering the values the participant alone held and calculating their average. In most sessions participants typed the values into the sheet themselves, ended that step, and then requested and authorized the agent's actions for the calculation. P17's log shows the sequence in full. Three participants (P6, P16, and P17) operated the automation switch on the data-entry step as well. P6 and P16 then met the dialog raised for a step the agent cannot perform, and both declined its offer to move that step into Teach. One session (P3) records no end of the first step, so participants did not all advance the plan through the same control.

The switch on each step card reports whether capability assessment found the step available, and turning it on requests automation and reveals the control that generates the step's actions. Some participants found the step-card buttons unclear at first, the automation switch in particular, and most reported that their meaning became clear after use. The interval from requesting a step's actions to authorizing their execution had a median of 0:56 min across the 18 intervals, one per participant, with a minimum of 29 seconds. That window contains the model's response time as well as the participant's reading of the proposal, so it does not measure inspection alone.

Four participants (P8, P14, P16, and P17) opened the controls for adding or editing plan steps, each closing the add-step interface again within four seconds without submitting anything. P14 also selected the edit control twice with no new step text recorded. The logs record no completed addition or edit and no reason for the closings, so these openings establish only that the controls were reached.

Participants who encountered the error popups Learn raises on a deviation described them as unobtrusive, and two asked for a longer display duration. Across the three modalities the facilitator observed participants clicking past such messages without reading them, which on occasion led to errors.

5.2.3 Time Within the Teaching Sequence. A teaching cycle asks the user for three contributions in sequence, writing the step description, performing the actions the extension records, and judging the interpretation the agent replays. We segmented every accepted step into three elapsed intervals bounded by interface events. Formulation runs from the acceptance of the previous step to the submission of the next description, and for a first step from the submission of the task or a later confirmation to teach it again. Recording runs from that submission to the moment the participant stops the recorder. Validation runs from the end of recording to the acceptance of a replay, and it contains generation,

confirmation, execution, and any rejected replay. All three intervals include waiting and facilitator assistance, so none of them isolates cognitive effort or model latency.

Figure 8 shows the interval durations by step position. For paired comparisons we used the 16 participants with at least two accepted steps, comparing each first interval with that person's mean over all later steps. The criterion excludes P5 and P6, the two who taught the whole procedure as one step. Four exact Wilcoxon tests covered the full cycle and its three intervals, Bonferroni-adjusted within this family. The first cycle took a median of 4:33 min, compared with 2:31 min for later cycles ($p_{\text{adj}} < .001$). Formulation medians were 1:32 and 0:47 min ($p_{\text{adj}} = 0.0085$); recording medians were 1:43 and 0:34 min ($p_{\text{adj}} < .001$). Validation medians were 0:41 and 0:35 min, and the paired test did not detect a difference at this sample size ($p_{\text{adj}} = 0.606$). The reduction is located in the two intervals holding the participant's own contribution, writing a description and producing the demonstration, while the interval containing generation and the replay showed no difference between positions.

Three differences accompany step position, the content of a step, the guidance available, and what a participant had already seen. Later steps operate on results the earlier steps produced and are shorter to describe and to demonstrate. The interviews place both the guidance participants reported needing and the confidence they reported gaining at the start of the task. The first cycle is therefore also the unfamiliar one. The recording interval also holds the waiting described in Section 5.2.1.

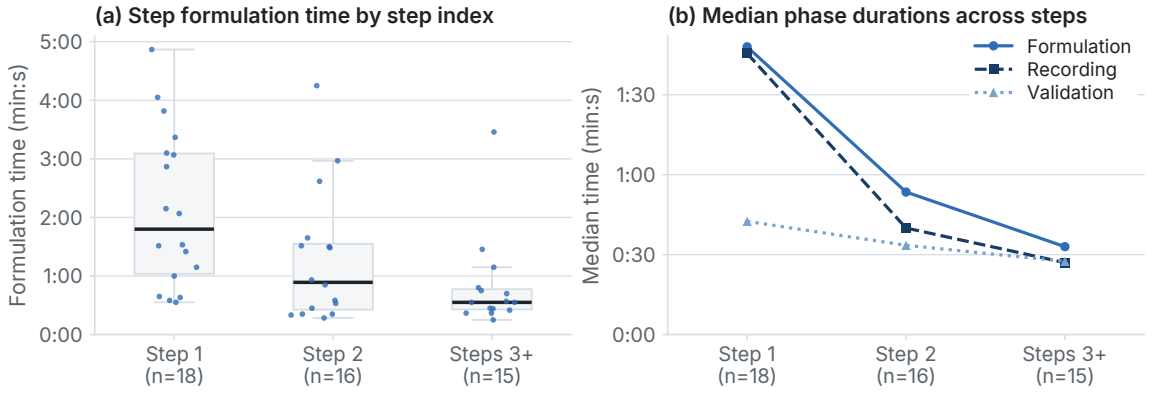


Fig. 8. Teach interaction intervals by step position. (a) Formulation intervals, with one point per participant and steps beyond the second averaged within participants. (b) Median phase durations. The plots include every participant reaching each position ($n = 18, 16$, and 15); the paired comparisons in the text use the same 16 participants for the first step and their mean over all later steps. Elapsed intervals include waiting and assistance.

5.2.4 Task Completion and Perceived Ease. An attempt counted as completed when the spreadsheet held the requested result at the end of the task. Participants completed 52 of 54 tasks, and the two unsuccessful attempts occurred in Teach. Every participant reached *End Task* in Teach having accepted at least one step. Both unsuccessful attempts therefore ended with a record saved and a sheet without the requested result. The study record does not identify which sessions these attempts came from, and the logs do not distinguish a replay's actions from the participant's. The point at which either attempt diverged is not recoverable.

The Learn task took a mean of 2:20 min, Collaborate 3:57 min, and Teach 11:19 min (Figure 9a), measured from the start of a task to its end and so counting generation time. Durations differed across the task-modality combinations ($\chi^2(2) = 33.72, p < .001$, Kendall's $W = 0.94$), with all three pairwise comparisons below $p_{\text{adj}} = .001$.

Median SEQ ratings were 7.00 for Learn, 6.00 for Collaborate, and 5.00 for Teach (Figure 9b; $\chi^2(2) = 16.11, p < .001$, Kendall's $W = 0.45$). The Teach task received lower ease ratings than each of the other two (Learn–Teach $p_{\text{adj}} = 0.0015$; Collaborate–Teach $p_{\text{adj}} = 0.025$), while the difference between Learn and Collaborate did not survive adjustment ($p_{\text{adj}} = 0.094$). The same task drew the reports of difficulty in Section 5.2.1.

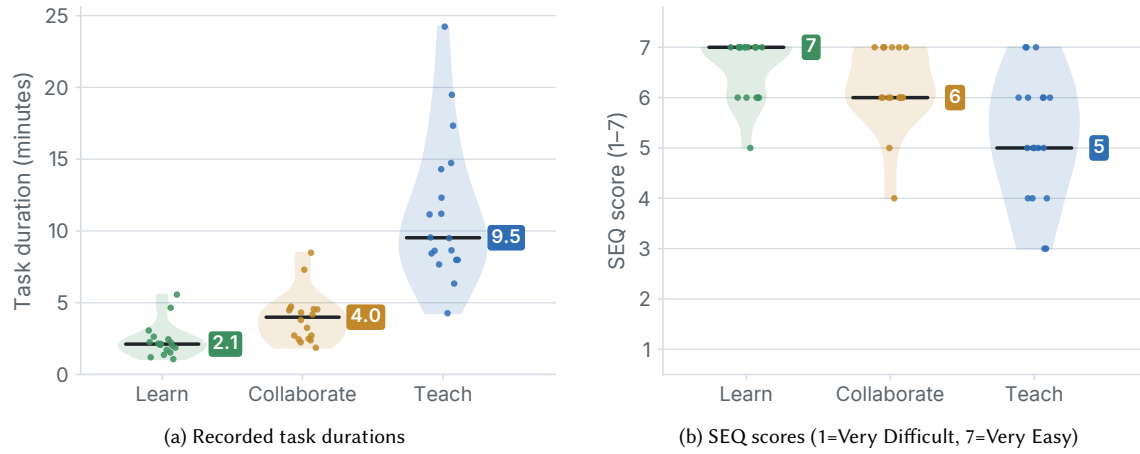


Fig. 9. Recorded duration and ease for the three assigned task-modality combinations ($N = 18$). (a) Durations include the two unsuccessful Teach attempts. Median durations are 2.12, 3.99, and 9.53 minutes for Learn, Collaborate, and Teach. (b) SEQ ratings summarize participants' perceived ease for each assigned activity.

5.2.5 Usability, Controllability, Trust, and Transparency. Overall SUS scores averaged 79.86 ($SD=9.76$) on the 0–100 scale (Figure 10). 15 of the 18 participants scored above 68, the value Sauro reports as an average score for the instrument [34], and 13 scored 80 or above. The highest positive item concerned integration of the system's functions (Q5), and confidence in using the system (Q9) received the lowest positive-item rating. Among the negatively worded items, the need to learn things before getting started (Q10) received the highest rating. Those two items concern readiness to operate the system. SUS was administered once for the system as a whole, so these ratings cannot be located in a particular modality.

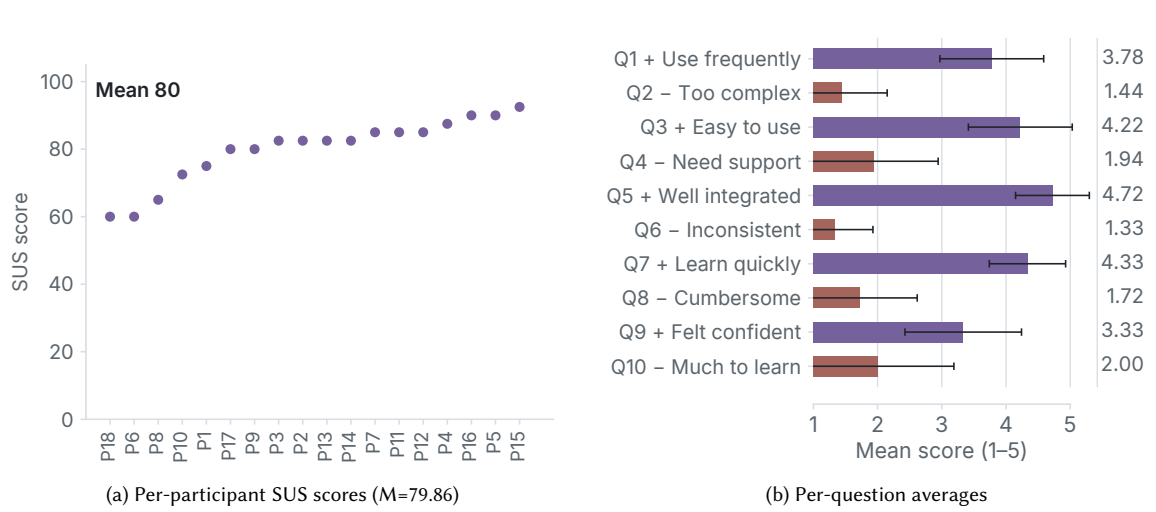


Fig. 10. System Usability Scale results for the same 18 participants. (a) Individual scores computed from the ten raw items and sorted ascending. (b) Item means and sample standard deviations; indigo and + indicate positively worded items, terracotta and - negatively worded items. All items have 18 responses.

The complete-case composite means were 4.53/5 for controllability ($n = 16$), 4.32/5 for trust ($n = 15$), and 3.94/5 for transparency ($n = 10$). These summaries use different participant subsets and item sets, so their ordering is not a tested difference between constructs. Figure 11 reports each item's mean and sample size.

The item asking whether the agent's capabilities and limits were clear received the lowest mean of the seven transparency items, with responses from all 18 participants. The item asking what the switch in Collaborate did received a lower mean than each of the items about the agent's explanations in Learn. Both items concern mechanisms participants reached only through what those mechanisms produced. Capability assessment decides availability without showing the entry it matched, and the switch reports that judgment through the control that requests the actions.

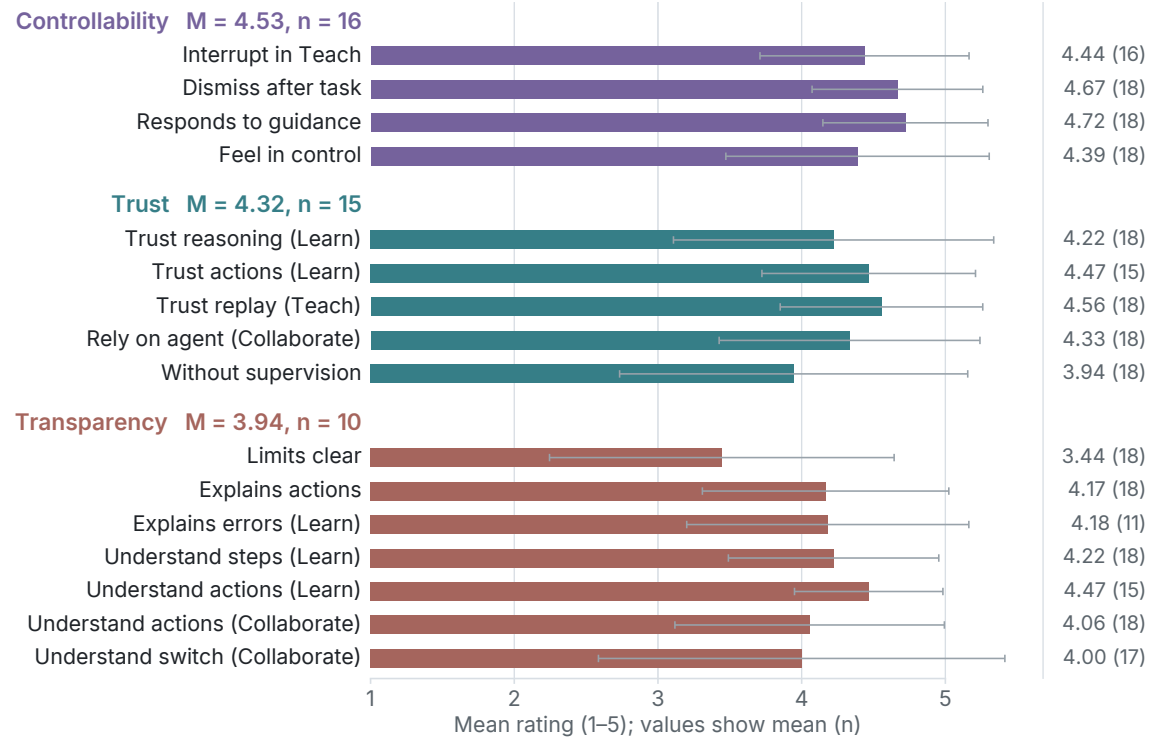


Fig. 11. Custom questionnaire item means and sample standard deviations using available responses, with n reported per item. Group headings show participant-level composite means from complete cases within that scale (controllability $n = 16$, trust $n = 15$, transparency $n = 10$). Item-level and composite summaries use the observed responses within their respective participant subsets.

6 Discussion

Specification and *coordination*, the two demands introduced in Section 1, organize what our observations show about teaching an agent and delegating execution to it. Specification asks the user to divide a procedure into steps and to word each description, and TALEA joins both acts to a single structure, the teaching step. That structure also fixes the scope of each judgment the user gives when the agent replays its interpretation. The user draws the boundary of a step at the outset, before observing what that boundary determines about the recording, the replay, and the stored record. We examine specification first, then what a judgment certifies, then the allocation of execution within a task, then coordination across a teaching cycle, and finally the artifact that teaching leaves behind.

6.1 Articulating a Procedure for an Agent

Participants divided the same procedure in different ways (Section 5.2.1). P5 and P6 each submitted the whole weighted average as one description, while P17 submitted one description per calculation and a fourth asking the agent to write a completion message. Ending a step determined four things at once, namely the text written as the step's description, the actions captured under it while recording, the replay offered for a single acceptance, and the entry retained in the task record. Programming by demonstration is an inductive problem in which the supplied examples bound the program that can be produced [18], so this boundary also bounds what the agent generalizes from. Systems that precede

TALEA derive the unit from the recording, and the user corrects and reuses a unit whose extent the system has already fixed [6, 19, 20]. In TALEA the unit originates with the user, who determines all four functions before observing what any of them produces.

Eight participants reported difficulty writing step descriptions and said they had needed facilitator guidance to decide what to include or how to divide the task. Among the 16 participants with at least two accepted steps, the interval ending when a description is submitted had medians of 1:32 min at a task's first step and 0:47 min at later steps ($p_{adj} = 0.0085$; Section 5.2.3). The first step is both the unfamiliar one and the facilitated one, so this interval does not isolate the cost of dividing a procedure without a previous step to constrain it. A plan is a resource elaborated in the course of acting [36], so a decomposition fitted to the occasion is produced in the doing. TALEA states no expected size for a step, and the operations the agent can perform on the sheet are represented nowhere in the interface. Entering Teach with a candidate decomposition would turn the division into a revision of a proposal. Collaborate already derives such a plan from a request, and Cocoa [12] and Magentic-UI [24] supply generated plans for editing.

A step description states the operations the user intends, while the demonstration supplies one execution of them, so each channel carries information the other cannot express. P6's description covered the weighted sum, the sum of the weights, and the division of the two results, whereas the only input recorded during that demonstration was a weighted-sum formula. P6 accepted the replay and ended the task, and our logs do not show how the agent reconciled the two inputs. P17's descriptions exhibit the opposite imbalance, naming the function to apply and the meaning of its arguments, which a demonstration cannot express on its own. TALEA represents no correspondence between the two channels, so a user cannot determine which input produced the replay and cannot ground a rejection in a divergence between them. The 3 rejections recorded across 57 reviewed replays are therefore uninformative about divergences of this kind. MIWA establishes such a correspondence between a generated step and the elements it acts on [6]. Applied to the two inputs a user supplies, the same correspondence would show which described calculations a demonstrated formula leaves unmatched, and the user could then revise the description, demonstrate the missing operations, or accept the narrower scope.

6.2 What a Judgment Certifies

An acceptance in Teach approves the replay displayed, which is one execution on the table in front of the user. The property that matters for a taught procedure is a generalization that no single execution exhibits. One replay is displayed per step whatever the step contains, so the boundary the user drew also fixes the scope of one acceptance. P5 and P6 each confirmed a whole procedure with a single judgment, while P17 confirmed three calculations and a completion step separately. Reliance develops from the outcomes a user observes [28], so what an acceptance can certify is bounded by what the interface displays at the moment of judging. Two attempts left the sheet without the requested result, both in Teach, and each passed through the same sequence of description, demonstration, and acceptance as the others before a record was saved (Section 5.2.4). The logs do not recover the point at which either attempt diverged. They establish only that accepted steps and a saved record can coexist with a spreadsheet that lacks the requested result. Displaying the formula and the ranges each action will use, together with a report of which operations earlier judgments have covered, would direct one confirmation at identifiable operations [32].

The same judgment carries different consequences in the two modalities. In Teach the user assesses whether the agent's interpretation matches the step being taught, and that assessment enters a record that later requests will draw on, whereas the corresponding judgment in Collaborate applies only to the current task. Both are solicited in the same terms, as a per-step approval of what the agent generated, so the interface can state at the point of approval what each

approval will be recorded against. A user accepting a step in Teach would then read that the accepted actions become the stored version of that step.

6.3 Where Control Over Execution Is Assigned

Each modality sets a default allocation of execution, and participants adjusted that allocation within a task at the granularity of one step. The plan Collaborate proposed for the weekly spending task separated entering the values from computing their average. P17 typed the values into the sheet, ended that step, and then requested and authorized the agent's actions for the calculation (Section 5.2.2). Learn assigns execution to the user by default, and three of the five participants who opened the hints panel requested a *Show Me* demonstration, which transfers those actions to the agent. Adjustment in the other direction carried a cost, because editing the description of a plan step withdraws that step from automation. Four participants opened the controls for adding or editing plan steps (P8, P14, P16, P17) and closed them without a recorded change.

The switch on a Collaborate step card serves two functions at once, reporting whether capability assessment found the step available and asking the agent to prepare its actions. Some participants found the switch unclear at first, and most reported that its meaning became clear after use. Autonomy and the moments at which actions require approval are separable commitments [7, 11]. A switch that both reports availability and requests the agent's actions joins them in one control. Each commitment can carry its own control. A report of availability would name the operation or taught task that was matched, and a request for actions would be separated from the permission to run them. A revision of a step's wording would state what that revision withdraws.

6.4 Where the Two Demands Fall Within a Cycle

A teaching cycle passes control between the two parties at four points, the writing of a description, the demonstration the extension records, the wait while the agent generates, and the authorization and judgment of the replay. The facilitator observed that participants read the activity indicator beside a submitted description as a sign of ongoing generation, and that they waited during part of the interval in which a demonstration was expected (Section 5.2.1). A report of progress and the contribution the interface awaited therefore reached the user through a single channel, the panel beside the spreadsheet. Participants had selected Teach themselves, so their difficulty concerned the act the sequence required next, which is knowledge distinct from knowing which modality is running. The waiting preceded any generation, so coordination was required while the user's procedural input was still being supplied. Bansal et al. separate communication about what an agent can do from communication about what it is doing [4], and our observations identify a third object of communication, the act currently expected of the user. Information that supports coordination belongs in the object both parties act on [10], which places that third object in the spreadsheet where the work takes place [12].

Lau groups the effort of teaching an automation with the effort of supervising it [17], and our intervals give no grounds for treating the two as one demand that declines together. Across the 16 participants with at least two accepted steps, the full cycle decreased from a median of 4:33 to 2:31 min, and the interval containing the demonstration from 1:43 to 0:34 min. The interval containing generation and the judgment of the replay was of similar length at both positions (0:41 and 0:35 min, $p_{\text{adj}} = 0.606$; Section 5.2.3). Articulation and demonstration therefore shortened at later steps while judgment did not. An interval bounded by interface events contains the waiting described above along with the acting, so these comparisons bound each demand without isolating it. Our observations identify no point at which the evidence needed for judging what the agent returns could be withdrawn.

6.5 The Procedure as a Retained Artifact

Ending Teach writes a task record, which capability assessment consults for a later request and which Learn and Collaborate receive together with the current page (Section 4.2). Approval is solicited step by step, the record is assembled when the session ends, and its only display is its appearance among the recently taught tasks on the Teach entry interface. Post-task feedback included a request for a message confirming that Teach had saved the task. A contribution is complete once both parties hold evidence that it has been registered [8], and Amershi et al. recommend conveying how a user's actions shape later system behavior [2]. P5 and P6 each ended the task after accepting one step, so the entire content of those records derives from a single replay. Presenting the record before it is committed, as an object the user can open and revise, would allow a step covering several operations to be divided while the unit can still be changed.

Controllability received the highest composite of the three rated properties (4.53/5) and transparency the lowest (3.94/5), with trust between them (4.32/5). These composites derive from different participant subsets and item sets, so we read the ordering as a description of the responses obtained (Section 5.2.5). The ordering corresponds to what the interface puts on screen, since the controllability items ask about controls the participant operates, whereas capability assessment and the entry it matched reached participants only through outcomes. The knowledge base is consulted whenever a request is refused or served, and its contents are displayed at neither moment, so a user choosing between following guidance, teaching, and delegating decides before any list of the stored procedures is shown. Listing the available operations and taught tasks before a modality is selected would give that choice a basis. Naming the matched record when a request is served would identify which stored procedure produced the response, so that a refusal would read as a missing record the user can supply.

TALEA supplies a matched task record with the current page when generating guidance in Learn or actions in Collaborate, so operations approved on one sheet inform the actions proposed for a sheet whose values and cell positions may differ. When grades and weights occupy different columns, the generated actions must address the corresponding ranges of the new table, and nothing in this mechanism guarantees that the stored references transfer. The acceptance given during teaching concerns the interpretation displayed at that time, whereas a review during reuse concerns the operations and targets generated for the present context. Li et al. observed revision of a demonstrated procedure at two such moments, one addressing the representation and one addressing its effects [20]. An interface for reuse can accordingly display the matched record beside the generated steps and name the cells each step will read and write, as Pista does with the applied formula and the affected ranges [33].

6.6 Limitations

Participants used TALEA on a custom spreadsheet website, with prompts, predefined operations, and page interactions configured for its structure, so the results do not establish robustness across websites with different DOM structures, authentication flows, or navigation requirements. The facilitator helped participants recover from agent and interface errors and formulate teaching steps. Completion of 52 of 54 tasks and the usability ratings therefore describe performance under supported conditions, and independent use was not observed.

The sample comprised 18 participants, 17 students and one working professional, screened for basic digital literacy and familiarity with simple formulas. Each modality was paired with a distinct task, so task and modality effects are confounded. The custom perception measures used complete-case subsets of 16, 15, and 10 participants, and the transparency scale's alpha of 0.64 falls below conventional thresholds. One researcher carried out the qualitative

analysis, without a second coder and without an inductive coding scheme. That analysis and the sample size limit the inferences our account of participants' experience can support.

Participants used the three modalities in separate tasks during approximately one hour, so no participant reused a procedure they had taught. The study included no comparison against manual execution, existing automation tools, or a reduced set of modalities, so it establishes no benefit from combining the three. Assessing human learning would require subsequent performance of the procedure without assistance. The tasks were low-stakes and the sessions short, so the trust ratings describe reported confidence in that setting. Longer use would be needed to examine how expectations change after repeated successes and failures, and whether decisions to delegate track the correctness of agent actions. It would also show whether users find enough value in supplying procedural input to keep doing so.

7 Conclusion

This paper examined an interface in which the same user follows guidance from a browser agent, teaches it a procedure through description and demonstration, and authorizes it to execute selected steps. TALEA connects these activities through task records, which retain the step descriptions a user wrote together with the actions that user approved. Those records are supplied to the agent when it generates guidance or actions for a later request. In an exploratory, facilitated study, 18 participants used the three modalities on distinct spreadsheet tasks, and we analyzed their interaction logs, questionnaires, and interviews.

Participants divided the same procedure in different ways, and where a step ended determined at once what was described, what was recorded, what one acceptance covered, and what was retained. Specification was left to the participant, since the interface stated no expected size for a step and displayed no correspondence between a description and the actions demonstrated under it. Coordination was required while procedural input was still being supplied, since a single channel carried both the report that the agent was working and the expectation that the user would demonstrate. These difficulties admit different remedies, one addressed to the content of a step and the other to the act the interface expects at a given moment. An interface that relates a description to its recorded and generated actions before a step is stored, and that announces the contribution it awaits, would address them separately. A subsequent evaluation should follow one procedure from demonstration to reuse on changed values and cell positions. It should record whether users identify the matched procedure and correct discrepancies in the actions generated from it.

References

- [1] James E Allen, Curry I Guinn, and Eric Horvitz. 1999. Mixed-initiative interaction. *IEEE Intelligent Systems and their Applications* 14, 5 (1999), 14–23.
- [2] Saleema Amershi, Dan Weld, Mihaela Vorvoreanu, Adam Fourney, Besmira Nushi, Penny Collisson, Jina Suh, Shamsi Iqbal, Paul N. Bennett, Kori Inkpen, Jaime Teevan, Ruth Kikin-Gil, and Eric Horvitz. 2019. Guidelines for Human-AI Interaction. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (Glasgow, Scotland UK) (CHI '19). Association for Computing Machinery, New York, NY, USA, 1–13. <https://doi.org/10.1145/3290605.3300233>
- [3] Anthropic. 2024. Introducing computer use, a new Claude 3.5 Sonnet, and Claude 3.5 Haiku. <https://www.anthropic.com/news/3-5-models-and-computer-use>. Accessed: 2025-02-01.
- [4] Gagan Bansal, Jennifer Wortman Vaughan, Saleema Amershi, Eric Horvitz, Adam Fourney, Hussein Mozannar, Victor Dibia, and Daniel S. Weld. 2024. *Challenges in Human-Agent Communication*. Technical Report MSR-TR-2024-53. Microsoft Research. <https://doi.org/10.48550/arXiv.2412.10380>
- [5] John Brooke. 1996. SUS: A 'Quick and Dirty' Usability Scale. In *Usability Evaluation In Industry*, Patrick W. Jordan, B. Thomas, Ian Lyall McClelland, and Bernard Weerdmeester (Eds.). Taylor & Francis, London, UK, 189–194. <https://doi.org/10.1201/9781498710411-35>
- [6] Weihao Chen, Xiaoyu Liu, Jiacheng Zhang, Ian Iong Lam, Zhicheng Huang, Rui Dong, Xinyu Wang, and Tianyi Zhang. 2023. MIWA: Mixed-Initiative Web Automation for Better User Control and Confidence. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology* (San Francisco, CA, USA) (UIST '23). Association for Computing Machinery, New York, NY, USA, 1–15. <https://doi.org/10.1145/3586183.3606720>
- [7] Nazli Cila. 2022. Designing Human-Agent Collaborations: Commitment, responsiveness, and support. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems* (New Orleans, LA, USA) (CHI '22). Association for Computing Machinery, New York, NY, USA, Article 420.

- 18 pages. <https://doi.org/10.1145/3491102.3517500>
- [8] Herbert H. Clark and Susan E. Brennan. 1991. Grounding in communication. In *Perspectives on Socially Shared Cognition*, Lauren B. Resnick, John M. Levine, and Stephanie D. Teasley (Eds.). American Psychological Association, Washington, DC, 127–149. <https://doi.org/10.1037/10096-006>
 - [9] Allen Cypher, Daniel C. Halbert, David Kurlander, Henry Lieberman, David Maulsby, Brad A. Myers, and Alan Turransky (Eds.). 1993. *Watch What I Do: Programming by Demonstration*. MIT Press, Cambridge, MA, USA.
 - [10] Paul Dourish and Victoria Bellotti. 1992. Awareness and coordination in shared workspaces. In *Proceedings of the 1992 ACM conference on Computer-supported cooperative work* (Toronto, Ontario, Canada) (CSCW '92). Association for Computing Machinery, New York, NY, USA, 107–114. <https://doi.org/10.1145/143457.143468>
 - [11] K. J. Kevin Feng, David W. McDonald, and Amy X. Zhang. 2025. Levels of Autonomy for AI Agents. *Knight First Amendment Institute at Columbia University* (2025). <https://knightcolumbia.org/content/levels-of-autonomy-for-ai-agents-1> Essay 25-15, “Artificial Intelligence and Democratic Freedoms” essay series, published July 28, 2025. Also available as arXiv:2506.12469.
 - [12] K. J. Kevin Feng, Kevin Pu, Matt Latzke, Tal August, Pao Siangliulue, Jonathan Bragg, Daniel S. Weld, Amy X. Zhang, and Joseph Chee Chang. 2026. Cocoa: Co-Planning and Co-Execution with AI Agents. In *Proceedings of the 2026 CHI Conference on Human Factors in Computing Systems* (Barcelona, Spain) (CHI '26). ACM, New York, NY, USA, 1–23. <https://doi.org/10.1145/3772318.3791673>
 - [13] Sumit Gulwani, William R. Harris, and Rishabh Singh. 2012. Spreadsheet Data Manipulation Using Examples. *Commun. ACM* 55, 8 (Aug. 2012), 97–105. <https://doi.org/10.1145/2240236.2240260>
 - [14] Hongliang He, Wenlin Yao, Kaixin Ma, Wenhao Yu, Yong Dai, Hongming Zhang, Zhenzhong Lan, and Dong Yu. 2024. WebVoyager: Building an End-to-End Web Agent with Large Multimodal Models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Bangkok, Thailand, 6864–6890. <https://doi.org/10.18653/v1/2024.acl-long.371>
 - [15] Wenyi Hong, Weihang Wang, Qingsong Lv, Jiazheng Xu, Wenmeng Yu, Junhui Ji, Yan Wang, Zihan Wang, Yuxiao Dong, Ming Ding, and Jie Tang. 2024. CogAgent: A Visual Language Model for GUI Agents. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, Seattle, WA, USA, 14281–14290. <https://doi.org/10.1109/CVPR52733.2024.01354>
 - [16] Faria Huq, Zora Zhiruo Wang, Frank F. Xu, Tianyue Ou, Shuyan Zhou, Jeffrey P. Bigham, and Graham Neubig. 2025. CowPilot: A Framework for Autonomous and Human-Agent Collaborative Web Navigation. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (System Demonstrations)*. Association for Computational Linguistics, Albuquerque, New Mexico, 163–172. <https://doi.org/10.18653/v1/2025.naacl-demo.17>
 - [17] Tessa Lau. 2009. Why Programming by Demonstration Systems Fail: Lessons Learned for Usable AI. *AI Magazine* 30, 4 (2009), 65–67. <https://doi.org/10.1609/aimag.v30i4.2262>
 - [18] Tessa A. Lau and Daniel S. Weld. 1998. Programming by demonstration: an inductive learning formulation. In *Proceedings of the 4th International Conference on Intelligent User Interfaces*. Association for Computing Machinery, New York, NY, USA, 145–152. <https://doi.org/10.1145/291080.291104>
 - [19] Gilly Leshed, Eben M. Haber, Tara Matthews, and Tessa Lau. 2008. CoScripter: automating & sharing how-to knowledge in the enterprise. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Florence, Italy) (CHI '08). Association for Computing Machinery, New York, NY, USA, 1719–1728. <https://doi.org/10.1145/1357054.1357323>
 - [20] Jiawen Li, Zheng Ning, Yuan Tian, and Toby Jia-Jun Li. 2025. ALLOY: Generating Reusable Agent Workflows from User Demonstration. *arXiv preprint arXiv:2510.10049* (2025). <https://doi.org/10.48550/arXiv.2510.10049>
 - [21] Toby Jia-Jun Li, Amos Azaria, and Brad A. Myers. 2017. SUGILITE: Creating Multimodal Smartphone Automation by Demonstration. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems* (Denver, Colorado, USA) (CHI '17). Association for Computing Machinery, New York, NY, USA, 6038–6049. <https://doi.org/10.1145/3025453.3025483>
 - [22] Xiao Liu, Bo Qin, Dongzhu Liang, Guang Dong, Hanyu Lai, Hanchen Zhang, Hanlin Zhao, Iat Long Iong, Jiadai Sun, Jiaqi Wang, Junjie Gao, Junjun Shan, Kangning Liu, Shudan Zhang, Shuntian Yao, Siyi Cheng, Wentao Yao, Wenyi Zhao, Xinghan Liu, Xinyi Liu, Xinying Chen, Xinyue Yang, Yang Yang, Yifan Xu, Yu Yang, Yujia Wang, Yulin Xu, Zehan Qi, Yuxiao Dong, and Jie Tang. 2024. AutoGLM: Autonomous Foundation Agents for GUIs. *arXiv preprint arXiv:2411.00820* (2024). <https://doi.org/10.48550/arXiv.2411.00820> arXiv:2411.00820 [cs.HC]
 - [23] Yimeng Liu, Misha Sra, Jeevana Priya Inala, and Chenglong Wang. 2026. ReUseIt: Synthesizing Reusable AI Agent Workflows for Web Automation. In *Proceedings of the 31st International Conference on Intelligent User Interfaces* (Paphos, Cyprus) (IUI '26). Association for Computing Machinery, New York, NY, USA, 885–908. <https://doi.org/10.1145/3742413.3789083>
 - [24] Hussein Mozannar, Gagan Bansal, Cheng Tan, Adam Fourney, Victor Dibia, Jingya Chen, Jack Gerrits, Tyler Payne, Matheus Kunzler Maldaner, Madeleine Grunde-McLaughlin, Eric Zhu, Griffin Bassman, Jacob Alber, Peter Chang, Ricky Loynd, Friederike Niedtner, Ece Kamar, Maya Murad, Rafah Hosn, and Saleema Amershi. 2025. Magentic-UI: Towards Human-in-the-loop Agentic Systems. *arXiv preprint arXiv:2507.22358* (2025). <https://doi.org/10.48550/arXiv.2507.22358> arXiv:2507.22358 [cs.AI]
 - [25] Brad A. Myers. 1986. Visual programming, programming by example, and program visualization: a taxonomy. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Boston, Massachusetts, USA) (CHI '86). Association for Computing Machinery, New York, NY, USA, 59–66. <https://doi.org/10.1145/22627.22349>
 - [26] OpenAI. 2025. Introducing ChatGPT agent: bridging research and action. <https://openai.com/index/introducing-chatgpt-agent/>. Product release announcement, 17 July 2025. Accessed July 2026.
 - [27] OpenAI. 2025. Introducing Operator. <https://openai.com/index/introducing-operator/>. Accessed: 2026-09-11.

- [28] Raja Parasuraman and Victor Riley. 1997. Humans and Automation: Use, Misuse, Disuse, Abuse. *Human Factors* 39, 2 (1997), 230–253. <https://doi.org/10.1518/001872097778543886>
- [29] Perplexity AI. 2025. Introducing Comet: Browse at the speed of thought. <https://www.perplexity.ai/hub/blog/introducing-comet>. Accessed: 2026-02-12.
- [30] Sundar Pichai. 2025. Google I/O 2025: Sundar Pichai’s Opening Keynote. <https://blog.google/innovation-and-ai/technology/ai/io-2025-keynote/>. Published May 20, 2025; accessed September 9, 2026.
- [31] Yujia Qin, Yining Ye, Junjie Fang, Haoming Wang, Shihao Liang, Shizuo Tian, Junda Zhang, Jiahao Li, Yunxin Li, Shijue Huang, Wanjun Zhong, Kuanye Li, Jiale Yang, Yu Miao, Woyu Lin, Longxiang Liu, Xu Jiang, Qianli Ma, Jingyu Li, Xiaojun Xiao, Kai Cai, Chuang Li, Yaowei Zheng, Chaolin Jin, Chen Li, Xiao Zhou, Minchao Wang, Haoli Chen, Zhaojian Li, Haihua Yang, Haifeng Liu, Feng Lin, Tao Peng, Xin Liu, and Guang Shi. 2025. UI-TARS: Pioneering Automated GUI Interaction with Native Agents. *arXiv preprint arXiv:2501.12326* (2025). <https://doi.org/10.48550/arXiv.2501.12326>
- [32] Karen J. Rothermel, Curtis R. Cook, Margaret M. Burnett, Justin Schonfeld, T. R. G. Green, and Gregg Rothermel. 2000. WYSIWYT Testing in the Spreadsheet Paradigm: An Empirical Evaluation. In *Proceedings of the 22nd International Conference on Software Engineering* (Limerick, Ireland) (ICSE ’00). ACM, New York, NY, USA, 230–239. <https://doi.org/10.1145/337180.337206>
- [33] Sadra Sabouri, Zeinabsadat Saghi, Run Huang, Sujay Maladi, Esmeralda Eufrazio, Sumit Gulwani, and Souti Chattopadhyay. 2026. Auditing and Controlling AI Agent Actions in Spreadsheets. *arXiv preprint arXiv:2604.20070* (2026). <https://doi.org/10.48550/arXiv.2604.20070>
- [34] Jeff Sauro. 2011. *A Practical Guide to the System Usability Scale: Background, Benchmarks & Best Practices*. Measuring Usability LLC, Denver, CO, USA. <https://measuringu.com/book/a-practical-guide-to-the-system-usability-scale/>
- [35] Ben Shneiderman. 2022. *Human-Centered AI*. Oxford University Press, Oxford, UK. <https://doi.org/10.1093/oso/9780192845290.001.0001>
- [36] Lucy A. Suchman. 1987. *Plans and situated actions: The problem of human-machine communication*. Cambridge University Press, Cambridge, UK.
- [37] Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. 2025. Agent Workflow Memory. In *Proceedings of the 42nd International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 267)*, Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (Eds.). PMLR, 63897–63911. <https://proceedings.mlr.press/v267/wang25bx.html> arXiv:2409.07429.
- [38] Justin D. Weisz, Jessica He, Michael Muller, Gabriela Hoefer, Rachel Miles, and Werner Geyer. 2024. Design Principles for Generative AI Applications. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) (CHI ’24). ACM, New York, NY, USA, 1–22. <https://doi.org/10.1145/3613904.3642466>
- [39] Shuning Zhang, Jingruo Chen, Zhiqi Gao, Jiajing Gao, Xin Yi, and Hewu Li. 2026. Characterizing Unintended Consequences of GUI Agents For Web Browsing. In *Proceedings of the 2026 CHI Conference on Human Factors in Computing Systems* (Barcelona, Spain) (CHI ’26). Association for Computing Machinery, New York, NY, USA, 1–19. <https://doi.org/10.1145/3772318.3790696>

A Exploratory LLM Prompt Experiments

During development, we explored model and prompt configurations for capability assessment, guidance, and action generation. These experiments informed prototype development but were separate from the participant study. The archive contains prompt revisions, output workbooks, and a summary workbook; it does not contain a controlled comparison of identical prompts across all configurations. We therefore describe the experimental scope without reporting a single model ranking or treating these results as end-to-end system reliability.

Table 3. Configurations named in the archived comparison. Reasoning effort is a configuration setting, not a quality ranking. GPT-5 medium-effort checks were additional experiments outside these comparison series.

Model family	Reported reasoning effort
GPT-4o	Not applicable
GPT-5	Low
GPT-5-mini	Medium
GPT-5-mini	Low
GPT-5-nano	Low

Experimental scope. Table 4 reports scenario counts documented in the development record and corresponding execution counts in the summary workbook. Each scenario was repeated four times. These repetitions probe variation in model output; they are not additional independent task scenarios. Inputs varied requests, capability lists, page states,

and, where applicable, demonstration or correction examples. Outputs were inspected manually for appropriate task matching, step content, formulas, selectors, and action sequences. The archive does not document an independent annotation protocol sufficient to estimate rating reliability.

Table 4. Scope of the archived prompt experiments. Counts are per configuration within a reported prompt series; revised series are retained separately in the source materials. Executions equal scenarios multiplied by four repetitions.

Function	Scenarios	Executions	Output inspected
Learn assessment	12	48	Match to listed capabilities
Teach assessment	12	48	Match to listed capabilities
Collaborate planning	12	48	Steps and availability judgments
Learn guidance	12	48	Steps, hints, formulas, and actions
Teach actions	4	16	Actions generated from demonstrations
Teach corrected actions	4	16	Actions generated with correction examples
Collaborate actions	8	32	Actions for the requested task

Interpretation. The archive shows that results depend on both function and prompt revision. For example, the initial Teach-action series records 16 of 16 outputs as correct for every configuration, whereas its revised series ranges from 8 to 16 correct outputs. These series cannot be combined selectively into a model comparison. Collaborate planning also contains distinct original and revised functions. The Learn and Collaborate generation prompts use reference task descriptions, original page states, and demonstrated actions alongside the current context. Correct output on these constructed examples does not independently establish procedural transfer in the participant study.

Cost and latency. The workbooks retain recorded response times and estimated costs, but differences in prompt revisions, input content, output structure, and token accounting limit direct comparisons. Cost estimates were derived using development-time assumptions and should not be interpreted as current prices or equivalent per-task expenditure.

Model selection. We selected GPT-5-mini with low reasoning effort for task assessment and Collaborate planning. Following additional manual stress tests, we selected GPT-5 with medium effort for Learn guidance, Teach actions and corrected actions, and Collaborate actions. This medium-effort configuration was outside the main comparison series, so the tabulated experiments do not quantify its performance.

B Study Instruments

The study used the ten-item System Usability Scale after all three tasks and a Single Ease Question after each task (1=Very Difficult, 7=Very Easy). The custom questionnaire and final interview prompts are reproduced below from the study materials. Custom items used a five-point agreement scale. An asterisk marks questions designated optional because they depended on events encountered during the session. Missing answers were not imputed; the analysis distinguishes item-level available responses from complete responses to each custom scale.

B.1 Custom Questionnaire

Controllability

- (1) I could easily interrupt the agent when repeating actions in Teach mode.
- (2) I could easily dismiss the agent after the task was completed.
- (3) The agent responds appropriately when I try to guide it in Teach mode.
- (4) I feel in control of the interactions with the agent.

Trust

- (1) I trust the agent's reasoning when generating the steps in Learn mode.
- (2) I trust the agents when playing actions on the screen for the Learn mode.*
- (3) I trust the agents when replaying actions on the screen for the Teach mode.
- (4) I can rely on the agent performing steps on my behalf in Collaborate mode.
- (5) I would be comfortable using this agent without constant supervision.

Transparency

- (1) The capabilities and limits of the agent were clear.
- (2) The agent explains its actions in a way I can understand.
- (3) The agent explains the Errors in Learn mode in a way that I can understand.*
- (4) I understand why the agent provides specific steps and hints in Learn mode.*
- (5) I understand what are the actions that the agent performs in Learn mode.*
- (6) I understand why the agent generates the actions for the Collaborate step.
- (7) I understood what the switch in the Collaborate mode did.

B.2 Final InterviewLEARN

- (1) Did you find the error messages too invasive?*
- (2) Did you find the hints helpful?*

TEACH

- (1) Was it easy to write the steps for the task? Was it clear to you how to split the task into multiple steps?
- (2) Was the process of teaching a step confusing?

COLLABORATE

- (1) Did you find it difficult to understand what the buttons did for each step?