Think-Aloud Computing: Supporting Rich and Low-Effort Knowledge Capture

Rebecca Krosnick*
University of Michigan
Ann Arbor, MI, USA
rkros@umich.edu

Fraser Anderson
Autodesk Research
Toronto, ON, Canada
fraser.anderson@autodesk.com

Justin Matejka
Autodesk Research
Toronto, ON, Canada
justin.matejka@autodesk.com

Steve Oney
University of Michigan
Ann Arbor, MI, USA
soney@umich.edu

Walter S. Lasecki
University of Michigan
Ann Arbor, MI, USA
wlasecki@umich.edu

Tovi Grossman
University of Toronto
Autodesk Research
Toronto, ON, Canada
tovi@dgp.toronto.edu

George Fitzmaurice
Autodesk Research
Toronto, ON, Canada
george.fitzmaurice@autodesk.com

ABSTRACT

When users complete tasks on the computer, the knowledge they leverage and their intent is often lost because it is tedious or challenging to capture. This makes it harder to understand why a colleague designed a component a certain way or to remember requirements for software you wrote a year ago. We introduce *think-aloud computing*, a novel application of the think-aloud protocol where computer users are encouraged to speak while working to capture rich knowledge with relatively low effort. Through a formative study we find people shared information about design intent, work processes, problems encountered, to-do items, and other useful information. We developed a prototype that supports think-aloud computing by prompting users to speak and contextualizing speech with labels and application context. Our evaluation shows more subtle design decisions and process explanations were captured in think-aloud than via traditional documentation. Participants reported that think-aloud required similar effort as traditional documentation.

CCS CONCEPTS

• **Human-centered computing** → Natural language interfaces.

KEYWORDS

Think-aloud, knowledge capture, documentation, process

*This work was done while the author was an intern at Autodesk Research.

CHI '21, May 08–13, 2021, Yokohama, Japan

ACM ISBN 978-1-4503-8096-6/21/05…$15.00
https://doi.org/10.1145/3411764.3445066

ACM Reference Format:
Rebecca Krosnick, Fraser Anderson, Justin Matejka, Steve Oney, Walter S. Lasecki, Tovi Grossman, and George Fitzmaurice. 2021. Think-Aloud Computing: Supporting Rich and Low-Effort Knowledge Capture. In *CHI Conference on Human Factors in Computing Systems (CHI '21), May 08–13, 2021, Yokohama, Japan.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3411764.3445066

1 INTRODUCTION

As a user interacts with software, they often leverage a vast amount of knowledge and skills to achieve high-level goals and intent within certain constraints and contexts. However, when they complete their task, the resulting artifact (e.g., code, 3D model, slide deck) contains little trace of the user's process or design rationale. This makes it harder for colleagues or the user's future self to understand why certain design decisions were made, how workflows were applied, and what elements of the artifact are important. This can result in wasted effort, misinterpreted designs, and steeper learning curves.

While there exist common documentation practices that aim to capture some of this information (e.g., code comments, slide notes), they often require additional effort or are not able to capture the full breadth of information that may be useful to others. Regardless of the kind of knowledge, its full potential is often not realized because it remains within the mind(s) of a single or select few team members. An alternative possible approach to capturing important information is to encourage users to speak in the moment, a technique LiveSnippets [18] uses for capturing narration of a person's travels, cooking, or product reviews for the purpose of experience writing.

We introduce *think-aloud computing*, a new knowledge capture approach where the computer encourages users to think aloud and share their goals, knowledge, and process as they work. This approach is a novel application of the *think-aloud* method typically leveraged in usability studies to learn what a user's process and mental model of a tool is [24]. With this approach, users can document a

wide variety of rich information with low effort, *concurrently* while doing the task, or *retrospectively* as they reflect on their work.

We conducted a formative study where we asked participants to speak *concurrently* (i.e., as they worked) or *retrospectively* (i.e., after they worked) over a 15-minute period, and found that useful and distinct knowledge is shared in these two conditions. Based on prior work and this study we present three components for a think-aloud computing system: 1) *prompting* the knowledge worker to speak and share useful information in a minimally distracting way, 2) *contextualizing* the captured information in a meaningful way by leveraging speech and software context clues, and 3) *presenting* the knowledge appropriately for various use cases.

We built a proof-of-concept system instantiating these prompting, contextualizing, and presenting components to enable think-aloud computing. A small widget encourages the user to speak by helping them visualize how much they have spoken about certain topics, and their audio is captured along with a screen recording and meta-data for later review in a live-archive window. Through an evaluation, we find that participants see benefits of this approach without much cost, and novel information was captured.

The primary contribution of this work is the concept of *think-aloud computing*, where a worker speaks (in an effort to capture their knowledge and intent) while completing work on the computer. To support this, we introduce techniques for explicitly eliciting and interpreting their knowledge in real-time. This contribution is motivated by a formative study, supported by a prototype designed to encourage this behavior, and evaluated through a twelve-person user study of the prototype and the approach overall.

2 RELATED WORK

This work is inspired by (and builds upon) think-aloud protocols, as well as research in documentation practices in knowledge work, context and speech capture, and live-streaming.

2.1 Think-Aloud (Background)

The think-aloud protocol is widely used in usability studies to understand participants' thoughts, processes, and reasons for their behavior [9, 24]. Using this protocol, researchers encourage users to speak about whatever they are thinking as they perform a task (e.g., what they are trying to do, why they are doing it, what they may be feeling, etc.). The think-aloud protocol is widely used because it is a lightweight method to capture rich information about participants' mental processes and reasoning [16]. For instance, researchers can observe that a user clicked the wrong button, and that they understand they clicked the wrong button, because they believed its blue color indicated 'confirmation' to them. Because of the rich data the think-aloud protocol produces, it has become one of the most popular tools to evaluate usability [48].

There are two approaches to the think-aloud process, which can be used in combination: *retrospective* and *concurrent* [20, 44] (alternatively referred to as *reflection-on-action* and *reflection-in-action* [38]). In retrospective think-aloud, the participant verbalizes their thoughts following the task. This approach places less cognitive load on the user during the task and elicits more comments about the final choices made, but may result in the user omitting some information that is lost in the moment [20]. In concurrent think-aloud, the user verbalizes thoughts while completing the task. With this approach, there is slightly more cognitive load during the task, especially for complex tasks, but more comments are elicited regarding the process itself [43].

This work applies the think-aloud protocol outside of the context of a study to capture thoughts, knowledge, process and decisions as users work. It supports both concurrent and retrospective think-aloud protocols to capture a wide breadth of information with minimal impact to user performance.

2.2 Documentation and Process Capture

Much work has examined how to better capture the workflows, decisions, and processes that users have while interacting with modern, complex software. Grossman et al.'s Chronicle project allows users to capture screen recordings of their workflows, as well as meta-data from instrumented applications, creating a rich, searchable knowledge bank [12]. ScreenTrack [17] captures screenshots and other software meta-data to provide a visual history of a user's work and help them retrieve relevant software, websites, and files when later resuming a task. Our think-aloud computing prototype uses some of these techniques. MixT and Torta leverage similar meta-data from a user's demonstration and use it to automatically generate tutorials that other users can benefit from [5, 30].

Beyond supporting learning, process capture has been explored as a way to document rationale, design intent and history, and reflect on and share the creative process. Raison d'Etre enables users to record and organize rationale and informal history in a lightweight way by making recorded video interviews searchable and organized [2]. TaskTracer monitors and captures high-level workflow through UI events for later reflection and exploration [8]. More recently, the maker community has been leveraging technology to share their works-in-progress for feedback, to disseminate their practices, and to encode the narrative or story of what they are building [37, 42]. To facilitate design reflection Co-notate enables audio and video recording of a design activity, and allows users to mark important events, such as 'problems', 'ideas', and 'decisions' [35], an approach our think-aloud computing prototype also leverages.

For programming activities, team members' design decisions can often be found in chat messages with each other. Post-literate programming takes advantage of this by letting users integrate their Slack discussions as comments into their code [31]. Similarly, Callisto stores and associates data scientists' chat messages with each other with elements in computational notebooks [45]. Other programming environments let users insert multimedia comments into their code to make documentation richer, for example narrated programming sessions or code overviews [15].

This prior work demonstrates various values of documenting and capturing how artifacts were created, and presents a variety of useful techniques: context capture, real-time speech narration, and real-time labeling. Think-aloud computing is the first work to leverage all of these techniques in one system.

2.3 Context and Speech Capture and Feedback

Being able to capture and record context in different situations has been an active research area within ubiquitous computing and

HCI. The field of lifelogging has made great strides in developing technology to capture a users' experiences and activities [39]. This can be used as an assistive aid for people with memory impairments to help them remember the past [21, 22], for self-reflection [41], or for improving health or happiness [25]. These systems often use audio/video streams to document the sights, sounds, and speech of the user and their environment [40].

Most similar to think-aloud computing is recent work, LiveSnippets [18], which enables people traveling, cooking, or reviewing a product to record their speech in-situ to draft a blog post for experience writing. During their experience, users can capture photos and videos and provide a speech narration. LiveSnippets asks the user domain-specific scaffolding questions to help guide their speech. After they have captured their experience, users can then review their speech contextualized with photos, videos, and other metadata and finalize the content for a blog post. Think-aloud computing similarly enables users to record their speech and context but for a different purpose – documenting information about their computer work (e.g., creating slides, a 3D model, code). We employ a couple different techniques: 1) subtle prompting through a visual widget, as to not overly interrupt workers, and 2) real-time processing of speech and software context to provide the user live feedback.

Recent advances in natural language processing have enabled novel, robust speech capture use-cases [34, 47]. McGregor and Tang explored the utility of introducing a personal audio assistant into a meeting room, but found users were hesitant to directly engage with it [29]. TalkTraces explored using audio capture to track discussion themes and agenda items over time [3], giving the team more context for their current and past meetings. Relatedly, wearable technology has been employed to measure nonverbal behaviors in individuals and groups and provide users feedback [6, 19, 32]. No prior approaches actively monitor what is being spoken, identify relevant information to the work-task at hand, and prompt the users to elaborate.

2.4 Live-Streaming

In recent years live-streaming [36] has emerged as a popular way for programmers [1, 4], gamers [13, 23], artists [11, 46], educators [14], and other domain experts to engage audiences for social or educational purposes, and to share their processes and knowledge. Live-streamers typically share their screen and narrate as they perform tasks, which viewers can follow in real-time or later when archived. Live-streaming is very related to think-aloud computing as a knowledge capture approach because streamers share knowledge, often tacit, in the context of tasks they are performing. Currently most archived live-streams are simply the video recording along with viewer comments for each timestamp, making it difficult for consumers to index the content. Recent work by Fraser et al. [10] has explored automatically segmenting archived live-stream videos into meaningful sections that the streamer can label to provide consumers with a table of contents to ease navigation. StreamWiki [26] helps streamers create summary content by enlisting their viewers to create such content in real-time. Audiences provide real-time feedback to live-streamers by expressing questions they have and additional information they would like shared. Think-aloud computing users work without a live audience, so an important aspect of think-aloud computing systems is contextualizing users' actions and speech in order to better prompt users to speak about the right things at the right times.

3 FORMATIVE STUDY

We conducted a formative study to explore the potential value and challenges of encouraging people to think aloud while working. Specifically, we wanted to understand the types of information that people would share in retrospective and concurrent think-aloud protocols, their attitudes toward speaking while working, and how concurrent think-aloud would impact their performance and effort.

3.1 Participants and Tasks

We recruited 12 participants aged 22–50 (mean 35) years, three identified as female, nine identified as male, from our organization to participate in a 1-hour study. Participants received a $25CAD gift card for their participation. We asked each participant to indicate which task domains (coding, slide creation, 3D modeling) they were comfortable with, and we assigned them a domain accordingly. Four participants were assigned to each of the three domains.

Our intent with this study was to understand the breadth of how think-aloud might be applicable across a range of computing tasks. We chose three disparate domains to get insights into how participants' opinions and think-aloud patterns may vary based on the task type. For example, coding is primarily text and language-based whereas 3D modeling is primarily visual and spatial.

As the study tasks, participants were asked to create any content of their choosing within their assigned domain. We also allowed them to use any software or programming language of their choice. Participants are referred to by C#, M#, S# for their coding, modeling, and slide creation tasks respectively.

3.2 Procedure

The study was a mixed design, with each participant performing one of three separate domains (between subjects; coding, slide creation, 3D modeling), and each participant completing each of the following two conditions (within subjects, with condition order counterbalanced):

- *Concurrent:* Speak during 15 minute creation task
- *Retrospective:* Speak after 15 minute creation task, adding information they felt was needed

For both conditions, participants were asked to imagine that they or a future colleague would revisit the artifact that they were creating and continue to work on it, learn from it, or otherwise need to understand what they were doing. They were also told to assume that the computer would capture all the information they were speaking and that it would be associated with the artifact when it was being revisited.

Following the tasks, participants were administered a short survey with 7-point Likert scale questions and participated in a short a semi-structured interview.

3.3 Results

Overall, unique and useful information about user processes, design intent, and many other categories was captured from both

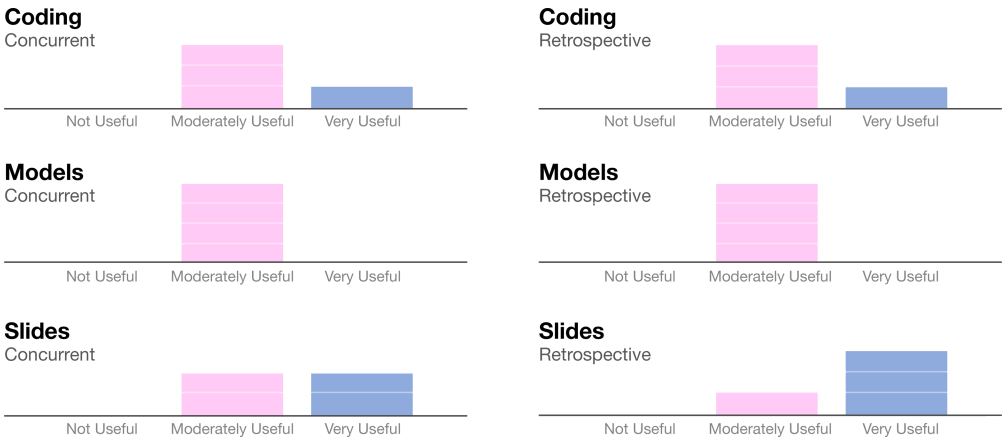

Figure 1: Participants' ratings of how useful the information they shared was, by domain and condition. The 7-point Likert scale results (between 1-Not Useful, and 7-Extremely Useful) have been aggregated into three categories: Not Useful (1, 2), Moderately Useful (3, 4, 5), and Very Useful (6, 7).

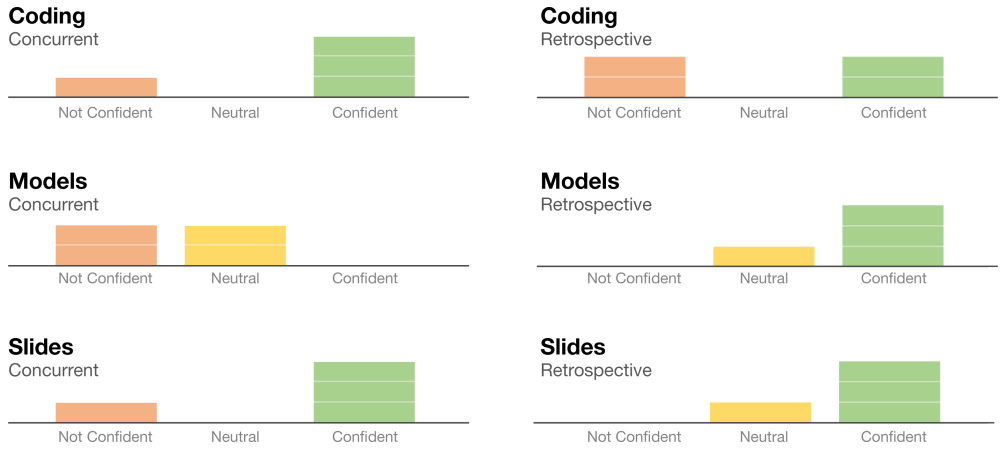

Figure 2: Participants' ratings of how confident they were that they captured all the important information to share. The 7-point Likert scale results (between 1-Very Not Confident, and 7-Very Confident) have been aggregated into three categories: Not Confident (1, 2, 3), Neutral (4), and Confident (5, 6, 7).

the concurrent and retrospective think-aloud conditions. Participants reported that think-aloud requires effort but they believe the information they capture will be useful.

3.3.1 Perceived Effort and Utility. The perception of the amount of effort required for think-aloud and the impact it had on performance varied by participant, and no clear effects of domain were observed. During the concurrent think-aloud task, 6 of 12 participants reported that think-aloud had positive or no impact on their work performance, but 8 of 12 participants reported that it required at least medium effort. Fortunately, participants felt their efforts were worthwhile. In both the concurrent and retrospective conditions all participants reported that the information they shared was at least moderately useful (Figure 1). Regarding confidence in capturing all the important information to share, 6 of 12 participants reported being confident for the concurrent condition, with 8 of 12 for the retrospective condition (Figure 2). The scores suggest the usefulness of sharing through think-aloud, but note that participants may be overly optimistic about the usefulness of their own speech. To understand the true usefulness of the captured speech, we would need to evaluate with consumers actually leveraging this information.

Participants also revealed what kinds of information they thought would be very important to capture, for example, in parametric modeling, sharing whether certain parameter choices were arbitrary or intentional, and why, is important: *"So I think information around those parameters would be key or critical. That's where all that information is. . . you go back and like, 'why did I make this 10 degrees?' And the answer is 'no reason', then great. It's 12 degrees now cause I like it better. . . Whereas like, 'no, it needs to be ten degrees because there's a thing on the ground that it's going to mesh with', then..."* (M2).

Some participants did comment on the effort of concurrently speaking-aloud, for example, "*especially since I didn't have a*

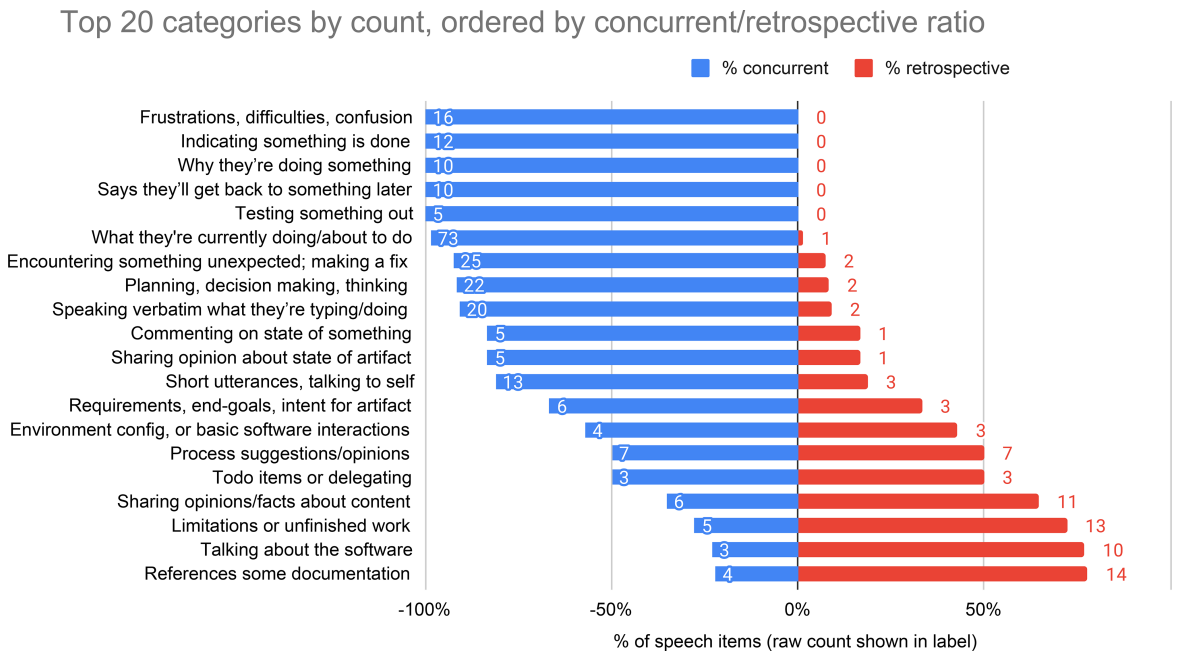

Figure 3: Top 20 categories by count (i.e., all categories with 5 or more speech items), ordered by concurrent to retrospective occurrence ratio. See full data in supplementary material.

complete plan in mind, I was just kind of doing things ad hoc. It was kind of difficult to justify everything that I was actually doing while I was doing it" (C4). However, our analysis of speech content and the types of information shared (reported below) found that a lot of likely useful information is captured in concurrent speech that is less often captured in retrospective speech, mirroring existing research on these protocols [20].

3.3.2 Information Captured. The amount of speech captured varied widely by participant, some very talkative and others relatively quiet. In the concurrent condition participants spoke between 17 and 170 utterances (median: 61), and in the retrospective condition participants spoke between 5 and 114 utterances (median: 16). We define an *utterance* as a contiguous set of words spoken until a pause of greater than one second occurs, as this is the unit the speech-to-text service we use transcribes real-time speech in.

We sampled approximately 450 *speech items* from the retrospective and concurrent think-aloud sessions and bucketed them into 57 categories. We define a *speech item* as a contiguous set of phrases or sentences that hold some shared semantic meaning. A speech item may include one or more utterances and may belong to one or more categories. We saw that many category types occur in high frequency in one of the two conditions relative to the other. For example, lower-level steps, frustrations, and planning and decision making all occurred more frequently during concurrent speech, while expression of limitations or unfinished work occurred more frequently in retrospective speech. Figure 3 shows the top 20 categories by speech item count (i.e., all categories with 5 or more speech items, excluding categories specific to the study environment) ordered by the concurrent to retrospective occurrence ratio.

From this set of 57 categories, we further distilled these into five classes, where each speech item could belong to multiple classes. While other classifications may be useful, we believe these cover most speech items observed in the formative study, as well as the majority of usage scenarios that we explored.

- **Design intent**: These referred to the high-level goals and intents of the user, and often explain why they are doing something, e.g., "*I would intend to be able to move forward and back in space here, it's like a focusing mechanism*" (M1).
- **Process**: These referred to the operations the user is doing, and the tools, commands and environment they are using, e.g., "*Now I'm going to add the card content*" (C3).
- **To-do item**: These referred to tasks that the user or someone else would have to complete later, e.g., "*so we need to do a lot of adding images, formatting, making it look pretty*" (S3).
- **Problem**: These referred to problems with the software or the user's approach, or other issues that needed to be dealt with, e.g., "*ooh, we have a bug*" (M4).
- **Important:** These referred to elements of the artifact or process that are flagged for special consideration, e.g., **"***it's also important for the team to know that. . . if you're doing any online research, please check the source*" (S1).

In addition to the content of the speech items, there is also value in capturing affect or sentiment. While we did not formally code for these, there were moments of excitement and frustration that could potentially add rich context. These potentially represent 'a-ha moments', or points of failure or frustration that others could learn from in the future [7].

4 THINK-ALOUD COMPUTING

Based on results of the formative study and prior work on process and context capture, we believe adding the think-aloud protocol to everyday computing tasks has the potential to add great value by capturing rich knowledge that is otherwise lost. We introduce *think-aloud computing*, where computer users are prompted to speak about what they are doing and why, with a system to capture speech, categorize it, and archive it for later use.

Think-aloud computing is related to but distinct from other recorded spoken language scenarios (e.g., medical examiners recording speech during an autopsy, lawyers recording audio of client meetings, live-streaming, and other cognitively-demanding tasks requiring concurrent documentation). Think-aloud computing is specifically focused on the techniques to effectively and non-intrusively capture and leverage verbalized knowledge for work performed on the computer: *prompting* the user to speak about particular information, *contextualizing* the speech to understand context, and *presenting* the captured speech in a meaningful way for various applications. We developed these three components based on results of the formative study (e.g., that sometimes people are quiet and need to be reminded to speak), prior work (e.g., that contextualizing audio with video capture and command metadata can be useful when presented to consumers), and our intuition (e.g., that processing and categorizing speech in real-time can help prompt speakers to speak more and about certain topics, and that these categorizations can be useful to consumers too).

4.1 Core Concepts

4.1.1 Prompting. From the formative study, we found there is a wide range of how much people speak while working and what they speak about. Additionally, certain use cases, users, or organizations might want to set goals or targets about the types of things they are speaking about (similar to organizational practices around comment quality and quantity). However, care must be taken not to overly interrupt the user and add significant cognitive load [27]. Several relevant factors are important when considering when and how to prompt or notify someone [33]:

- *Notification Level:* The degree to which you interrupt the user. As think-aloud computing should minimize the cognitive load, the system should subtly prompt the user, e.g., with slight changes in size or color of a graphic, so as not to frequently distract the user.
- *Representational Fidelity:* How the information is encoded is relevant, as it should be readily consumable. To allow users to readily parse how well they are meeting their goals, the representation should be simple and iconic.
- *Information Capacity*: As the system has a limited understanding of the domain the user is working in, the information capacity conveyed in the prompts should be small.

4.1.2 Contextualizing. A think-aloud computing system should capture and process the user's speech as well as other software context. With natural language processing, it should automatically detect important information or topics of interest. Processing speech sentiment or affect could also be useful in identifying when the user has encountered a problem (i.e., is frustrated) or has solved a problem (i.e., is excited). Capturing software context (e.g., applications open, actions performed, mouse/key events, cursor/scrollbar position), as seen in prior work [5, 12, 30], could be helpful for consumers to understand the context around captured speech. Cues from the captured speech and software context can also help the system intelligently prompt the user.

4.1.3 Presenting. The information captured can be presented in different ways depending on the application. It is likely useful to have a filtering or search interface for viewing particular kinds of information. If the interface presents a full text transcript, it might be helpful to highlight particularly important information and gray out unimportant information.

It could also be useful to embed transcribed speech appropriately in the given software artifact, for example attached to the created artifact or the line of code written. This may provide useful context for users working on the content in the future without requiring them to consult a separate archive to view the captured information.

4.2 Sample Usage Scenarios

Based on interviews with participants and the knowledge they captured, we believe think-aloud computing could be useful in a variety of scenarios.

4.2.1 Building on a Colleague's Work. Within an organization, often a given software artifact will be created by one person then worked on by many others. This happens when the original creator leaves the company, project ownership changes, or the project enters a new stage. The new artifact owner is now tasked with adding a new feature, adapting for a new use case, or adjusting for a new manufacturing material. The original creator had ideas in mind when they made design decisions, and likely there were particular constraints they were adhering to. If these constraints are not clear to the new owner, there is a risk they might violate them, for example, removing a software dependency they thought was no longer needed, or shrinking a gap in a model too much. Think-aloud computing could help capture design intents of the original creator that likely would not have been documented in writing.

4.2.2 Learning from a Colleague. People want to learn from each other, for example, to improve their skills with a particular software or learn best practices. The information captured via think-aloud computing will help people review a colleague's video capture and find relevant parts via the text transcript and labels. By not only being able to view their end product or a screen recording of their actions, having think-aloud data alongside this information would allow the learner to better understand the reasoning behind the colleague's actions.

4.2.3 Providing Feedback on a Colleague's Work. More senior team members often give their junior colleagues feedback—for example, in a code review, on a user interface mockup, or on a data analysis. They provide feedback both on the final artifact as well as the process of getting there. Think-aloud computing could give senior team members insight about their colleague's design intent and work process, helping them give more useful and focused feedback.

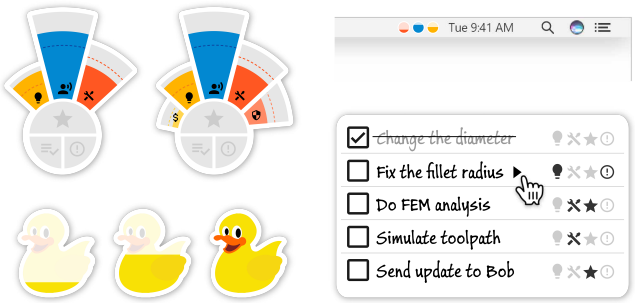

Figure 4: Several possible designs for the user interface of a think-aloud computing system.

4.2.4 Refresh Context from Your Own Work. Projects often span long time scales and old work documents need to be revisited. Even if users create a document themselves, if they created it many weeks or months ago, likely they may not remember why all the design decisions were made. Think-aloud computing could help them capture more of these design decisions so their future-self can access them.

4.2.5 Create Opportunities for Reflection. The ability to revisit thoughts and actions from someone's prior work session opens doors for reviewing their full design space. One could see the design options they tried or considered, and why they were or were not chosen. This could be especially useful for more creative domains like graphic design or writing.

4.3 Potential Interface Designs

Depending on the user, their organization, and their use case, there are many possible user interface designs within the *prompting-contextualizing-presenting* design space that might make sense for a think-aloud computing system (Figure 4). For example, the wedge designs (Figure 4, top-left) indicate how much the user has spoken about certain pre-specified topics (e.g., wedges for design intent, process information, cost, security) to encourage them to speak more in the areas that are lacking.

An alternative, more subtle version displayed in the operating system's menu-bar might still encourage the desired speech (Figure 4, top right). If a user only wanted to use it during a 'rubber duck debugging' session, a simplified version that just encourages overall speech might be more useful (Figure 4, bottom left). Lastly, users or organizations may see value in only capturing and acting on a specific category, such as 'to-do', so a tailored version could be used (Figure 4, bottom right).

For the purpose of exploring the viability of a think-aloud computing system, we chose to implement and evaluate one interface design, a wedge-based design (Figure 4, top-left) using the 5 classes (design intent, process, problem, to-do, and important) from the formative study. This is the prototype we discuss for the remainder of this paper. The wedge design and the 5 classes are just a single instantiation of the think-aloud computing approach and we are not claiming this is the optimal design. Likely there is no optimal design and instead different interface designs are better for some scenarios than others. Additionally, we focus on exploring *prompting* and *contextualizing* in this work, leaving *presenting* for future work.

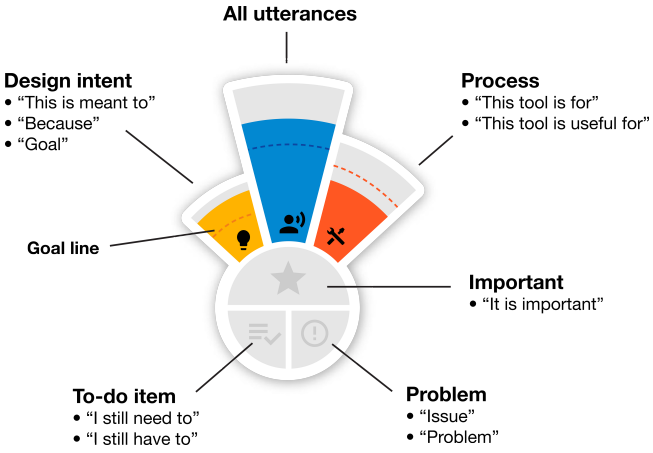

Figure 5: Visual widget that allows people to reflect on the amount they have spoken about different categories of information. Wedges fill up as that element is spoken about, and subtle prompts encourage the user to speak about a topic. The annotations indicate which words will ensure automatic classification into those categories.

5 PROTOTYPE

Based on the think-aloud computing design space and implications from the formative study, we built a software-agnostic prototype system that prompts users to speak via a small, always-visible widget (Figure 5), captures and processes their speech and software context, and lets them view/refine captured information in a live archive window (Figure 6). Users can also retrospectively capture knowledge by recording speech snippets or typing text comments while reviewing a previous work session.

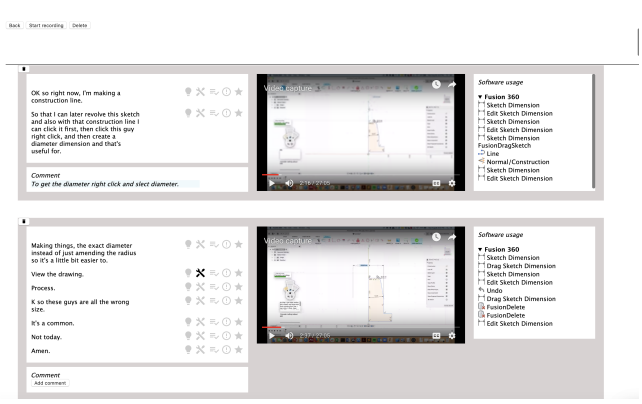

Figure 6: Live archive window, where users can review recorded speech, screen recording, and meta-data from past sessions, as well as the current capture session.

5.1 Ambient Display

The user's primary interaction with the system is just to speak, and occasionally reference the visual widget to understand if they are meeting targeted speech levels for the various categories. As they speak, the system will capture and transcribe their speech, as well as capture video of their screen and available software metadata.

As our goal is to avoid distracting the user from their work, the widget is intentionally small and can be placed in an unobtrusive area of their screen. The widget serves two purposes: 1) to inform the user how much they are speaking relative to their targets and 2) to collect speech labels from the user to improve the classification process.

The blue 'wedge' conveys how much the user has spoken overall. Each time the user speaks, the blue wedge fills a small amount. The wedge represents the number of speech utterances in a given time range (e.g., the last five business days), so as the user's speech rate decreases, the wedge fill-level will decrease. The wedge includes a dotted 'goal line' that users are encouraged to reach. Here we set the goal line based on average speech rates in the formative study, but in an in-the-wild system, the value could be based on past speech rates, best practices, or another metric or context.

In addition to encouraging speech in general, in the formative study we identified five kinds of information that we want to encourage users to share: *design intent*, *process*, *to-do items*, *problems*, and anything particularly *important*.

Design intent (yellow wedge) and *process* (red wedge) are filled as the user speaks that type of information. The widget leverages rudimentary keyword matching to classify the user's speech. The user can speak particular keywords (depicted as annotations in Figure 5; e.g., "This is meant to" for design intent) to help the widget classify their speech and update the wedge fill-levels. A thick black border is also briefly added to matching wedges to make the automatic classification apparent to the user. If the widget fails to classify the user's speech correctly, the user can manually click a wedge to classify their last utterance. In the future, more robust natural language processing approaches would allow for more flexibility in speech and less manual classification from the user.

To-do item, *problem*, and *important* are presented as buttons in the widget, and can also be activated through speaking keywords or through manual button clicks. These labels are buttons rather than wedges because although these kinds of information are useful to capture, they will not necessarily occur in a given work session. For example, it does not make sense to encourage a user to speak about *problems* if they are not experiencing any.

5.2 Prompting

In addition to subtly cueing users with wedge fill-levels, the widget will pulsate if it thinks the user could have something interesting to say, to encourage them to reveal more about their current actions or thoughts. For example, currently the "design intent" wedge will pulsate if the user performs multiple "Undo" operations in a row, as the user likely has important but potentially undocumented reasons for these actions (e.g., they made a mistake and want to try a different approach). The important, to-do item, and problem buttons will pulsate if the user speaks a curse word, which we are using to approximate sentiment recognition. Future versions of this system could prompt users based on other actions, words, or context clues for eliciting speech at critical points.

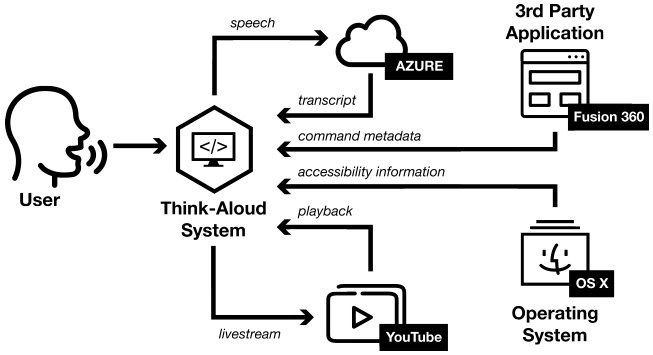

Figure 7: System overview of the implemented think-aloud computing system

5.3 Live Archive

A *live archive window* (Figure 6) presents transcribed utterances and their real-time labels alongside corresponding video capture and software commands used. This serves both as an initial presentation interface for people who want to leverage the spoken information and its context, as well as a post hoc editing interface for creators.

For the most part, creators will keep the live archive window minimized during work time, and then can retrospectively view the information they captured and correct transcription mistakes, remove utterances, correct utterance labels, and type or speak additional comments. If the transcription is erroneous, raw audio/video is available for the user to reference. The ability to make edits gives the user control over what is captured and shared, and gives them an opportunity to reflect on their past work and add any new insights they have. With improved speech recognition and classification, we anticipate fewer interactions with the editing functionality, and this interface will largely be used for retrospective think-aloud. To facilitate retrospective think-aloud, audio transcription is supported as a means of adding content.

5.4 Implementation

The system (Figure 7) is built as a two-window Electron application. Utterance data is persisted using MongoDB. Video capture is performed using FFmpeg for encoding a livestream to YouTube. The system uses Microsoft Azure Cognitive Services speech-to-text for transcribing the think-aloud speech in real-time. We created plugins to capture command data directly from Fusion 360, as well as system-wide for currently active software using the MacOS accessibility API. Keyword-based natural language processing is used for classifying speech utterances.

6 EVALUATION

To better understand the value and limitations of the think-aloud computing concept, we conducted a study to answer two key questions: What information is captured with think-aloud versus more traditional documentation practices? How do knowledge capture

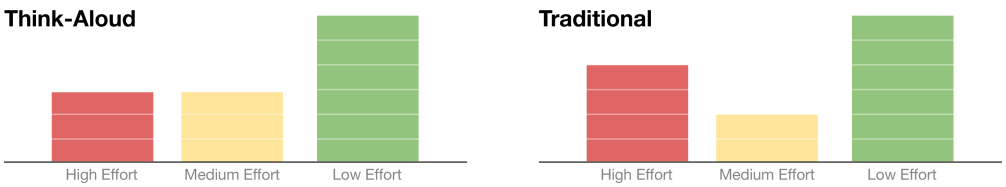

Figure 8: Ratings of effort it took to document knowledge, for think-aloud and traditional conditions. The 7-point Likert scale results (between 1-Very Low Effort, and 7-Very High Effort) have been aggregated into three categories: Low Effort (1, 2, 3), Medium Effort (4), and High Effort (5, 6, 7).

workflows of think-aloud and traditional text documentation compare?

6.1 Participants and Tasks

We recruited twelve participants from within our organization to complete the study. There was no overlap with the formative study. Five identified as male, seven female, and the mean age was 33 years (range 19–57). The study lasted 90 minutes and participants were compensated with a $50CAD gift card. Participants all had some experience with the task they were being asked to complete.

The study was a mixed design, with each participant performing one of three separate domains (between subjects; coding, slide creation, 3D modeling; four participants per domain), and each participant performing the task twice (within subjects; once with the think-aloud system, and once using traditional documentation). Condition order and associated task were counterbalanced. For coding, participants coded tic-tac-toe and connect four games; for slide creation, participants made slide decks about a historical event and an activity they were interested in; and for 3D modeling, participants created models of kitchen appliances and furniture.

As with the formative study, conducting a more in-depth study with more participants for a particular domain would be valuable, but our main goal was to understand the value and challenges of think-aloud computing compared to traditional processes across a variety of domains.

6.2 Procedure

6.2.1 Training Phase. Participants were provided context for the kinds of information they might want to share, why they might be sharing, and who their audience may be. It is a similar script to the one used in the formative study, with the addition of five potential information types we highlight (design intent, process, to-do, problem, important). For each condition, we briefly explained how to capture knowledge and gave participants an opportunity to practice.

- *Traditional text documentation:* We provided participants a Word document they could capture information in and showed them how to take screenshots. They were also informed they could document in any other natural way (e.g., code comments or slide notes). We chose text documentation as our baseline condition because this is how people commonly document in practice.
- *Think-aloud tool:* We gave a brief tutorial on interacting with the prompting widget (Figure 5), including how to use the keywords shown in Figure 5 to automatically classify speech, pointing out that the wedges fill as you speak more, and that dotted lines indicate fill-goals per wedge. The number of utterances required to hit the goal lines were chosen based on data from the formative study.

6.2.2 Work Phase. For each condition, participants were first given 25 minutes to work on the instructed task and asked to capture knowledge along the way.

6.2.3 Review Phase. Participants were then given 5 minutes to reflect on the work they had done and information they had captured and make any additions or edits to the captured information. For the think-aloud condition, we demonstrated how to make additions or edits in the live archive window. We then conducted a survey and interview.

6.3 Results

6.3.1 Survey Ratings. Participants rated the amount of effort it took to document knowledge similarly for the two conditions (Figure 8). In both conditions, 6 of 12 participants reported that it took "low effort" to document their knowledge while working. For the think-aloud condition, 3 participants rated "medium effort" and 3 "high effort", while for the traditional condition 2 rated "medium effort" and 4 rated "high effort".

6.3.2 Information Captured only with Think-Aloud Computing. The amount of information captured varied widely by participant. In the think-aloud condition, participants spoke between 46 and 297 utterances (median: 172). In the traditional documentation condition, participants captured between 10 and 41 lines (median: 26) in their Word documents, code comments, and slide notes. While utterance and line counts cannot be directly compared, they do give a sense of the volume of information captured in each condition.

After thematically clustering participants' captured content, we found *five categories* of information that were captured with the think-aloud protocol, but not traditional documentation. That these categories of information were not captured with the traditional techniques suggests that besides simply enabling a different capture paradigm, the think-aloud technique actually enables the capture of *different types* of information when compared to existing techniques.

- *Problem solving/debugging.* Lightweight labeling of a "problem" could provide easy indexing if the user wants to come back later to debug it and refresh their memory about the failure and debugging attempts (captured via speech and software context). Since identification of a problem is sometimes

followed-up with a solution, this could also provide a helpful collection of problem/solution pairs.

- *Context.* In many cases, participants provided additional contextual information in think-aloud, detailing why they visited a given website or why they found it useful. With traditional documentation, some of this information may be lost.
- *Available design choices or alternatives.* Some participants discussed very subtle design alternatives or choices. For example, slides participant S4 said "*Something to think about, are we using Russian or Soviet terminology here.*" Access to design choices could encourage discussion amongst collaborators or curiosity in other consumers.
- *Checkpoints.* Across all three tasks, participants uttered self-congratulatory remarks such as *"ok that looks reasonable"* (C1) after finishing a step and seeing a correct output/behavior. These checkpoints are interesting in that they could be used as state markers to enable users to find or revert back to past places in their task. These utterances also suggest an opportunity to enhance the segmentation of speech and potentially create dynamic task checklists or process steps.
- *Unconscious design decisions/constraints.* There were numerous instances where the user made on-the-fly, unconscious decisions that they likely would not have included in traditional textual documentation, for example, M3 said "*so new body, 2 inches, that's good, actually probably make it less, make it 1.5, just because that would be a thick piece of stock*" and "*want to give it a bit of flare, tapered edge, 10 degrees is aggressive, maybe something like 6, or 7*". In both these examples, valuable information about the user's requirements or constraints are captured and could later be used to understand why decisions were made and to learn from the users' knowledge and skills.

6.3.3 Observations and Interview Results. Overall, participants found value in capturing knowledge through think-aloud, and many themes such as causes of disruption, refining captured speech, and speaking preferences emerged in the semi-structured interviews.

Disruptions Dependent on Task and Documentation Type. A dominant theme within the interview data was the disruptions that using a voice-based method such as think-aloud would prevent or cause. Some thought that the think aloud-technique would be less disruptive, especially when performing a task that was not rooted in language, for example 3D modeling, *"I'm just speaking out loud and I'm already doing that when I'm modeling in my head so, I don't think it adds additional effort or much effort or burden on me versus trying to write in a document"* (M2). Because 3D modeling programs currently do not have any commenting functionality, the need to switch to another window to write traditional text documentation was cumbersome for many modelers and interrupted their workflow, e.g., "*when you are working on a screen and then you want to switch to something else and you have to just switch the screens and go to the Word document, I really hate it*" (M4) and "*it was like a total drain. It's like half the time I was spending documenting, half the time was spent designing. . . it's definitely hurting productivity*" (M1). Alternatively, coding participants found less distraction with traditional documentation, "*I think maybe talking out loud was a little bit difficult to do while coding cause it kinda takes a little bit of time to switch between explaining your code and then actually coding. . . I guess cause coding is kinda close to just typing a comment, so it wasn't as hard to switch*" (C2). These comments are consistent with Mayer's redundancy principle [28].

Capture and Refinement Workflows. During the study sessions, many participants spoke using a stream of consciousness, likely because it was easier than actively determining what to say and what not to say. As a result, the prompting widget captured a lot of speech, which has both benefits and disadvantages: "*when I could see the results, I think it was cool that I could see the breakdown of what was said, but there was a lot of garbage in there*" (M1). Some participants had difficulty determining what would be useful to say while performing a task, e.g., "*I think my thought process is kind of almost forming as I'm working on something. So, I find that a lot of these intermediate thoughts are actually not that useful.*" (C1). Relatedly, many participants decided to speak but only minimally classify their speech because they wanted to focus on their work or were not exactly sure how to classify their speech. As noted by M3, "*I preferred speaking rather than speaking and then thinking about what category that fell under. Because it's easy to just talk through your thing, rather than talking through it and then thinking what bucket it needs to go with.*" We believe these effects will decrease as people become more familiar with speaking while working and learn the kinds of information that they and their colleagues find valuable.

Value of Thinking-Aloud. Regardless of the challenges in labeling and self-filtering speech, most participants saw value in being able to capture intent and process information using a think-aloud method. As noted by C2, think-aloud could be useful for capturing assumptions or subtle design choices, e.g., "*the assumptions that I'm making I think are probably like the most important things to record because things like the top corner is (0,0) on the top left, it's stuff that isn't really part of how the game works, it's just something that you need to know for it to work*".

Others thought captured information would be useful for understanding the design history and components of a model, e.g., "*like for robotics, it's really difficult to just look at a design and be like, 'oh this is why they did it'. . . What you usually do is go scroll through the history of design. . .And you step-by-step see how they built it. . .But if that could just be spoken to you, that's super useful*" and "*Like if you open up like an assembly of this thousand part model, I have no idea why this part is there. 'Like what is this part, is it a custom part?'...So just being able to speak to like, 'oh, I just got this from McMaster-Carr and imported the model in',,,*" (M1).

Participants expressed the benefits of automatically capturing a list of to-do items or unresolved problems, "*Keeping a rolling tally of things that you have left to do, that can be super helpful, or problems that you've encountered along the way, because you could then, later on, almost be like, 'okay, well I resolved that problem', or 'these problems are still outstanding'*" (M3).

Others thought it would be useful for teaching, "*I could see it being almost like when you're recording a tutorial or something where if you went back and you had to send it to someone after, it'd be useful to sort of mark up what you had been saying or presenting*" (S2).

S4 commented on the increased utility of think-aloud computing if captured information is embedded within the software artifact, "*I could see using some of these things if it was embedded, I think in the tool itself. I could imagine the [PowerPoint] presenter notes being enhanced with some of these capabilities to categorize*".

6.3.4 Study Limitations. Through this study we learned participant opinions about think-aloud computing and we observed the kinds of information that think-aloud computing may capture. However, we did not explicitly evaluate whether consumers of information captured via think-aloud computing would find it useful for their own work. Information utility should specifically be studied in the future, but based on participant interviews and our own intuition we believe that such knowledge would be useful for a variety of work scenarios.

7 DISCUSSION AND FUTURE WORK

We found that a wide variety of unique information is captured when people think aloud while working, and that in fact unique information about *subtle design decisions/constraints*, *design choices/alternatives*, *problem solving/debugging*, and *checkpoints* is captured that is rarely captured in written documentation. Without this information, the original creator or other team members revisiting a software artifact might perform redundant work or make a change that breaks something. Even though think-aloud captures unique and subtle information that traditional documentation does not, participants found that they still require similar effort. Below we discuss where we think think-aloud computing could be particularly helpful and future work to explore improved techniques and particular applications.

7.1 Where Could Think-Aloud be Effective?

We suspect think-aloud could be particularly useful for spatial and visual tasks (like 3D modeling, creating slides, or editing 2D graphics). Using speech enables users to capture information without needing to switch modalities between graphical manipulation and typing. Since writing and coding are both text-based, writing textual notes seems to be more manageable, and is already commonplace in text and code editors.

Every tool has scenarios and people it is and is not ideal for. Think-aloud computing probably will not work well for people who are uncomfortable thinking-aloud, or those in an environment where speaking-aloud is not socially feasible. Relatedly, novices (i.e., to a domain or particular software) may be less willing to speak because they must remain very focused on their work to make progress, or, they fear anything they say could be incorrect. Further work is needed to better understand the specific impacts of think-aloud computing within these scenarios, as well as further identify benefits unique to these situations.

7.2 Better Prompting, Contextualizing, and Presenting

While the current implementation was sufficient to evaluate the potential of this approach, integrating more advanced natural language processing algorithms would allow the think-aloud computing widget to automatically classify and *contextualize* more utterances. This would approach our vision of the user simply speaking while working, with useful information automatically getting captured, meaningfully organized and summarized. Better NLP would also enable the widget to more intelligently identify opportune times to *prompt* the user to elaborate.

It would also be useful to explore how to present captured think-aloud information depending on the scenario, e.g.: adjusting the information presented to a learner based on their skill level; a personal task manager for to-do items and recently completed tasks; or transcribed think-aloud embedded contextually appropriately within code, a model, or other artifact.

7.3 Think-Aloud Computing Applications

We explored think-aloud computing as an approach specific to computer-based tasks. However, thinking-aloud while working probably brings similar knowledge capture benefits to other task domains, such as physical tasks (e.g., sports training, construction work) or collaborative tasks (e.g., group search, planning). Future work should explore technology for appropriately prompting think-aloud as well as contextualizing and presenting knowledge in these new domains.

Think-aloud computing could be useful beyond simply capturing and presenting knowledge. It could be leveraged for providing the user automated real-time assistance in their work, for example, suggesting relevant tools and workflows, sharing Stack Overflow posts when the user encounters a challenge, generating bug reports, or synthesizing test cases based on the speaker's words, speech affect, and interactions with software.

We also believe that this approach could be helpful to the traditional use case of the think-aloud protocol, usability testing, in particular for non-supervised user studies. If a system could take over the role of prompting users to speak the right kinds of information at the right times, user studies could be conducted on a longer-term scale, in different time zones, or otherwise without the researcher present. Researchers could then post hoc filter data to see how users interacted with specific components. However, more work would be necessary to understand the qualitative and quantitative difference between the results generated by using the automated approach compared to a traditional human-driven prompt.

8 CONCLUSION

Think-aloud computing is a promising new approach for capturing knowledge. Through a formative study we discovered the wide variety of information that people share when they think-aloud while working. We built a prototype think-aloud computing system and found that think-aloud captures information that traditional written documentation typically does not: problem solving steps, additional context, available design choices, checkpoints, and unconscious design decisions; information that could be useful to workers. Participants also felt that think-aloud computing requires similar effort to traditional text-based documentation practices. We believe think-aloud computing will enable people to more effectively leverage the knowledge of their colleagues and past selves.

ACKNOWLEDGMENTS

We the reviewers for their feedback, which has helped improve this work. We also thank Nikhita Joshi, Kimia Kiani, and Roya Shams for their feedback and our study participants for their time.

REFERENCES

[1] Abdulaziz Alaboudi and Thomas D. LaToza. 2019. An Exploratory Study of Live-Streamed Programming. In *2019 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, IEEE, Memphis, TN, USA, 5–13. DOI: https://doi.org/10.1109/VLHCC.2019.8818832

[2] John M. Carroll, Sherman R. Alpert, John Karat, Mary Van Deusen, and Mary Beth Rosson. 1994. Raison d'etre: capturing design history and rationale in multimedia narratives. In *Proceedings of the CHI'94 Conference on Human Factors in Computing systems*, Publ by ACM, 192–197.

[3] Senthil Chandrasegaran, Chris Bryan, Hidekazu Shidara, Tung-Yen Chuang, and Kwan-Liu Ma. 2019. TalkTraces: Real-Time Capture and Visualization of Verbal Content in Meetings. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, ACM, 577.

[4] Yan Chen, Walter S. Lasecki, and Tao Dong. 2021. Towards Supporting Programming Education at Scale via Live Streaming. *Proc. ACM Hum.-Comput. Interact.* 4, CSCW3 (January 2021), 1–19. DOI: https://doi.org/10.1145/3434168

[5] Pei-Yu Chi, Sally Ahn, Amanda Ren, Mira Dontcheva, Wilmot Li, and Björn Hartmann. 2012. MixT: automatic generation of step-by-step mixed media tutorials. In *Proceedings of the 25th annual ACM symposium on User interface software and technology*, ACM, 93–102.

[6] Ionut Damian, Chiew Seng (Sean) Tan, Tobias Baur, Johannes Schöning, Kris Luyten, and Elisabeth André. 2015. Augmenting Social Interactions: Realtime Behavioural Feedback using Social Signal Processing Techniques. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems - CHI '15*, ACM Press, Seoul, Republic of Korea, 565–574. DOI: https://doi.org/10.1145/2702123.2702314

[7] Sidney D'Mello, Tanner Jackson, Scotty Craig, Brent Morgan, P. Chipman, Holly White, Natalie Person, Barry Kort, R. El Kaliouby, and Rosalind Picard. 2008. AutoTutor detects and responds to learners affective and cognitive states. In *Workshop on emotional and cognitive issues at the international conference on intelligent tutoring systems*, 306–308.

[8] Anton N. Dragunov, Thomas G. Dietterich, Kevin Johnsrude, Matthew McLaughlin, Lida Li, and Jonathan L. Herlocker. 2005. TaskTracer: a desktop environment to support multi-tasking knowledge workers. In *Proceedings of the 10th international conference on Intelligent user interfaces*, ACM, 75–82.

[9] K. Anders Ericsson and Herbert A. Simon. 1980. Verbal reports as data. *Psychological review* 87, 3 (1980), 215.

[10] C. Ailie Fraser, Joy O. Kim, Hijung Valentina Shin, Joel Brandt, and Mira Dontcheva. 2020. Temporal Segmentation of Creative Live Streams. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, ACM, Honolulu HI USA, 1–12. DOI: https://doi.org/10.1145/3313831.3376437

[11] C. Ailie Fraser, Joy O. Kim, Alison Thornsberry, Scott Klemmer, and Mira Dontcheva. 2019. Sharing the Studio: How Creative Livestreaming can Inspire, Educate, and Engage. In *Proceedings of the 2019 on Creativity and Cognition*, ACM, San Diego CA USA, 144–155. DOI: https://doi.org/10.1145/3325480.3325485

[12] Tovi Grossman, Justin Matejka, and George Fitzmaurice. 2010. Chronicle: capture, exploration, and playback of document workflow histories. In *Proceedings of the 23nd annual ACM symposium on User interface software and technology*, ACM, 143–152.

[13] William A. Hamilton, Oliver Garretson, and Andruid Kerne. 2014. Streaming on twitch: fostering participatory communities of play within live mixed media. In *Proceedings of the 32nd annual ACM conference on Human factors in computing systems - CHI '14*, ACM Press, Toronto, Ontario, Canada, 1315–1324. DOI: https://doi.org/10.1145/2556288.2557048

[14] William A. Hamilton, Nic Lupfer, Nicolas Botello, Tyler Tesch, Alex Stacy, Jeremy Merrill, Blake Williford, Frank R. Bentley, and Andruid Kerne. 2018. Collaborative Live Media Curation: Shared Context for Participation in Online Learning. In Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems - CHI '18, ACM Press, Montreal QC, Canada, 1–14. DOI: https://doi.org/10.1145/3173574.3174129

[15] Yiyang Hao, Ge Li, Lili Mou, Lu Zhang, and Zhi Jin. 2013. Mct: A tool for commenting programs by multimedia comments. In *Proceedings of the 2013 International Conference on Software Engineering*, IEEE Press, 1339–1342.

[16] Morten Hertzum, Kristin D. Hansen, and Hans HK Andersen. 2009. Scrutinising usability evaluation: does thinking aloud affect behaviour and mental workload? *Behaviour & Information Technology* 28, 2 (2009), 165–181.

[17] Donghan Hu and Sang Won Lee. 2020. ScreenTrack: Using a Visual History of a Computer Screen to Retrieve Documents and Web Pages. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, ACM, Honolulu HI USA, 1–13. DOI: https://doi.org/10.1145/3313831.3376753

[18] Hyeongcheol Kim, Shengdong Zhao, Can Liu, and Kotaro Hara. 2020. LiveSnippets: Voice-based Live Authoring of Multimedia Articles about Experiences. In *22nd International Conference on Human-Computer Interaction with Mobile Devices and Services*, ACM, Oldenburg Germany, 1–11. DOI: https://doi.org/10.1145/3379503.3403556

[19] Taemie Kim, Agnes Chang, Lindsey Holland, and Alex Sandy Pentland. 2008. Meeting mediator: enhancing group collaborationusing sociometric feedback. In *Proceedings of the ACM 2008 conference on Computer supported cooperative work - CSCW '08*, ACM Press, San Diego, CA, USA, 457. DOI: https://doi.org/10.1145/1460563.1460636

[20] Hannu Kuusela and Paul Pallab. 2000. A comparison of concurrent and retrospective verbal protocol analysis. *The American journal of psychology* 113, 3 (2000), 387.

[21] Matthew L. Lee and Anind K. Dey. 2008. Using lifelogging to support recollection for people with episodic memory impairment and their caregivers. In *Proceedings of the 2nd International Workshop on Systems and Networking Support for Health Care and Assisted Living Environments*, ACM, 14.

[22] Matthew L. Lee and Anind K. Dey. 2008. Lifelogging memory appliance for people with episodic memory impairment. In *Proceedings of the 10th international conference on Ubiquitous computing*, ACM, 44–53.

[23] Pascal Lessel, Alexander Vielhauer, and Antonio Krüger. 2017. Expanding Video Game Live-Streams with Enhanced Communication Channels: A Case Study. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, ACM, Denver Colorado USA, 1571–1576. DOI: https://doi.org/10.1145/3025453.3025708

[24] Clayton Lewis. 1982. Using the'thinking-aloud'method in cognitive interface design. *Research Report RC9265, IBM TJ Watson Research Center* (1982).

[25] Ian Li, Jon Froehlich, Jakob E. Larsen, Catherine Grevet, and Ernesto Ramirez. 2013. Personal informatics in the wild: hacking habits for health & happiness. In *CHI'13 Extended Abstracts on Human Factors in Computing Systems*, ACM, 3179–3182.

[26] Zhicong Lu, Seongkook Heo, and Daniel J. Wigdor. 2018. StreamWiki: Enabling Viewers of Knowledge Sharing Live Streams to Collaboratively Generate Archival Documentation for Effective In-Stream and Post Hoc Learning. *Proc. ACM Hum.-Comput. Interact.* 2, CSCW (November 2018), 1–26. DOI: https://doi.org/10.1145/3274381

[27] Gloria Mark, Daniela Gudith, and Ulrich Klocke. 2008. The cost of interrupted work: more speed and stress. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, ACM, 107–110.

[28] Richard E. Mayer and Cheryl I. Johnson. 2008. Revising the redundancy principle in multimedia learning. *Journal of Educational Psychology* 100, 2 (2008), 380–386. DOI: https://doi.org/10.1037/0022-0663.100.2.380

[29] Moira McGregor and John C. Tang. 2017. More to meetings: challenges in using speech-based technology to support meetings. In *Proceedings of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing*, ACM, 2208–2220.

[30] Alok Mysore and Philip J. Guo. 2017. Torta: Generating mixed-media gui and command-line app tutorials using operating-system-wide activity tracing. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology*, ACM, 703–714.

[31] Soya Park, Amy X. Zhang, and David R. Karger. 2018. Post-literate Programming: Linking Discussion and Code in Software Development Teams. In *The 31st Annual ACM Symposium on User Interface Software and Technology Adjunct Proceedings*, ACM, 51–53.

[32] Rupa Patel, Andrea Hartzler, Mary Czerwinski, Wanda Pratt, Anthony Back, and Asta Roseway. 2013. Visual Feedback on Nonverbal Communication: A Design Exploration with Healthcare Professionals. In *Proceedings of the ICTs for improving Patients Rehabilitation Research Techniques*, IEEE, Venice, Italy. DOI: https://doi.org/10.4108/icst.pervasivehealth.2013.252024

[33] Zachary Pousman and John Stasko. 2006. A taxonomy of ambient information systems: four patterns of design. In *Proceedings of the working conference on Advanced visual interfaces*, ACM, 67–74.

[34] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. URL https://s3-us-west-2.amazonaws.com/openai-assets/research-covers/language-unsupervised/language_understanding_paper.pdf (2018).

[35] Søren Rasmussen, Jeanette Falk Olesen, and Kim Halskov. 2019. Co-notate: Exploring Real-time Annotations to Capture Situational Design Knowledge. In *Proceedings of the 2019 on Designing Interactive Systems Conference*, ACM, 161–172.

[36] Raquel Robinson, Jessica Hammer, and Katherine Isbister. 2019. All the World (Wide Web)'s a Stage: A Workshop on Live Streaming. In *Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems*, ACM, Glasgow Scotland Uk, 1–8. DOI: https://doi.org/10.1145/3290607.3299016

[37] Daniela K. Rosner and Kimiko Ryokai. 2008. Spyn: augmenting knitting to support storytelling and reflection. In *Proceedings of the 10th international conference on Ubiquitous computing*, ACM, 340–349.

[38] Donald A. Schon. 1984. *The reflective practitioner: How professionals think in action*. Basic books.

[39] Abigail Sellen and Steve Whittaker. 2010. Beyond total capture: a constructive critique of lifelogging. *Communications of the ACM* (2010).

[40] Mohit Shah, Brian Mears, Chaitali Chakrabarti, and Andreas Spanias. 2012. Lifelogging: Archival and retrieval of continuously recorded audio using wearable devices. In *2012 IEEE International Conference on Emerging Signal Processing Applications*, IEEE, 99–102.

[41] Alice Thudt, Dominikus Baur, Samuel Huron, and Sheelagh Carpendale. 2015. Visual mementos: Reflecting memories with personal data. *IEEE transactions on visualization and computer graphics* 22, 1 (2015), 369–378.

[42] Tiffany Tseng. 2016. Build in progress: Building process-oriented documentation. *Makeology: Makerspaces as learning environments. New York: Routledge* (2016).

[43] Maaike Van Den Haak, Menno De Jong, and Peter Jan Schellens. 2003. Retrospective vs. concurrent think-aloud protocols: testing the usability of an online library catalogue. *Behaviour & information technology* 22, 5 (2003), 339–351.

[44] Maaike J. Van den Haak and Menno DT De Jong. 2003. Exploring two methods of usability testing: concurrent versus retrospective think-aloud protocols. In *IEEE International Professional Communication Conference, 2003. IPCC 2003. Proceedings.*, IEEE, 3 pp.

[45] April Yi Wang, Zihan Wu, Christopher Brooks, and Steve Oney. 2020. Callisto: Capturing the "Why" by Connecting Conversations with Computational Narratives. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, ACM, Honolulu HI USA, 1–13. DOI: https://doi.org/10.1145/3313831.3376740

[46] Saelyne Yang, Changyoon Lee, Hijung Valentina Shin, and Juho Kim. 2020. Snapstream: Snapshot-based Interaction in Live Streaming for Visual Art. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, ACM, Honolulu HI USA, 1–12. DOI: https://doi.org/10.1145/3313831.3376390

[47] Tom Young, Devamanyu Hazarika, Soujanya Poria, and Erik Cambria. 2018. Recent trends in deep learning based natural language processing. *ieee Computational intelligenCe magazine* 13, 3 (2018), 55–75.

[48] Thinking Aloud: The #1 Usability Tool. *Nielsen Norman Group*. Retrieved September 18, 2019 from https://www.nngroup.com/articles/thinking-aloud-the-1-usability-tool/