Continuous Thinking Aloud

Jan Ole Johanssen, Lara Marie Reimer, and Bernd Bruegge
Department of Informatics
Technical University of Munich
Garching bei München, Germany
jan.johanssen@tum.de, laramarie.reimer@tum.de, bruegge@in.tum.de

***Abstract*—Thinking Aloud is a method that allows the collection of expressive user feedback for software improvement. However, its frequent application in a rapid development processes such as Continuous Software Engineering (CSE) is challenging, since repetitively performing manual observations and evaluations demand high effort. We propose the *Continuous Thinking Aloud* (CTA) approach for conducting Thinking Aloud during CSE. CTA records speech feedback for a user who starts using a new feature increment. The recordings are automatically transcribed and classified into one of four feedback categories that differentiate between insecure, neutral, positive, and negative sentiments. CTA visualizes these feedback classifications on a sentence level, next to its related high-level change of the feature increment. This supports developers in problem discovery, in particular regarding usability. CTA integrates with CSE processes and represents a scalable approach enabling repeated application during the software development lifecycle.**

***Index Terms*—thinking aloud, continuous software engineering, usability engineering, feature crumb, classifier, visualization.**

I. Introduction

Continuous software engineering (CSE), i.e., the rapid and frequent release of software increments [1], [2], puts a special emphasis on user feedback as valuable source for improvement. User feedback might be provided through informal channels, such as email, or as a formal issue or bug report.

Jakob Nielsen's Thinking Aloud method represents another feedback source to retrieve expressive insights: it bases on the idea that users, while using the application, immediately verbalize their thoughts [3]. The access to such a rich and qualitative feedback source benefits software development, in particular with respect to usability. However, the application of Thinking Aloud in a CSE process faces challenges.

First, as minor changes during CSE occur frequently, it is particularly important that users' *spontaneous* impressions are transmitted to developers "cheap, fast, and easy" [4]. Traditional Thinking Aloud impedes these goals, as it requires users to attend a dedicated test setup in a laboratory environment.

Second, stopping processes frequently to perform regression testing accounts for a major waste of development time [5], which most likely further increases if developers are repeatedly performing and analyzing Thinking Aloud protocols.

We argue that a successful integration of Thinking Aloud in CSE requires a scalable approach to reduce manual activities. This decreases the time investment for users and developers: while the former would be able to verbalize thoughts in their target environment, the latter would be able to focus on feedback utilization, rather than on its collection or processing.

In this position paper, we introduce the *Continuous Thinking Aloud (CTA)* approach for the automation of Nielsen's Thinking Aloud method to obtain better insights into users' thoughts on a new feature increment. CTA targets applications developed in a CSE environment; it asks users to record speech feedback for a new feature increment as soon as they start using it. These recordings are analyzed automatically on a sentence level, meaning that we use a natural language classifier to differentiate the content of the user feedback between four categories, namely insecure, neutral, positive, and negative sentiment. Results of the analysis are visualized in accordance with the newly implemented functionality, providing developers with a powerful tool to efficiently utilize and benefit from expressive user feedback during CSE. Overall, CTA represents an additional knowledge source for automated and continuous experimentation that is available to developers.

This paper is structured as follows. Section II outlines foundations and Nielsen's Thinking Aloud method. Section III presents the CTA approach that is further discussed in Section IV based on first experience reports. Section V outlines related work. Section VI summarizes future directions to improve the CTA approach. Section VII concludes the paper.

II. Foundations

Thinking Aloud originates from psychological research with the goal of developing a method to model cognitive processes: by inspecting protocols, i.e., written transcripts, different approaches in solving a problem become identifiable [6].

The method's idea was transferred to software engineering as a means to support the identification of usability problems through asking subjects to provide feedback while using the software [7]. Nielsen describes Thinking Aloud as "the single most valuable usability engineering software engineering method" [3], as it reflects users' thoughts about a software: He emphasizes that the main strength of the method lies on the quality of feedback gained by only a few testers, even though such feedback may have been biased based on the testers' subjective theories regarding the software. Thinking Aloud reveals the underlying reasons for the user's actions [8]: it allows developers to gain a better understanding of application usage and to detect usability issues they were unaware of.

We refer to Nielsen's Thinking Aloud [3] as the *traditional* method, since a tester and a supervisor physically sit together with former talking while using the application, whereas a supervisor takes notes to manually identify usability problems.

Accepted at: Joint 5th International Workshop on Rapid Continuous Software Engineering and
1st International Workshop on Data-Driven Decisions, Experimentation and Evolution (RCoSE/DDrEE 2019), Copyright IEEE

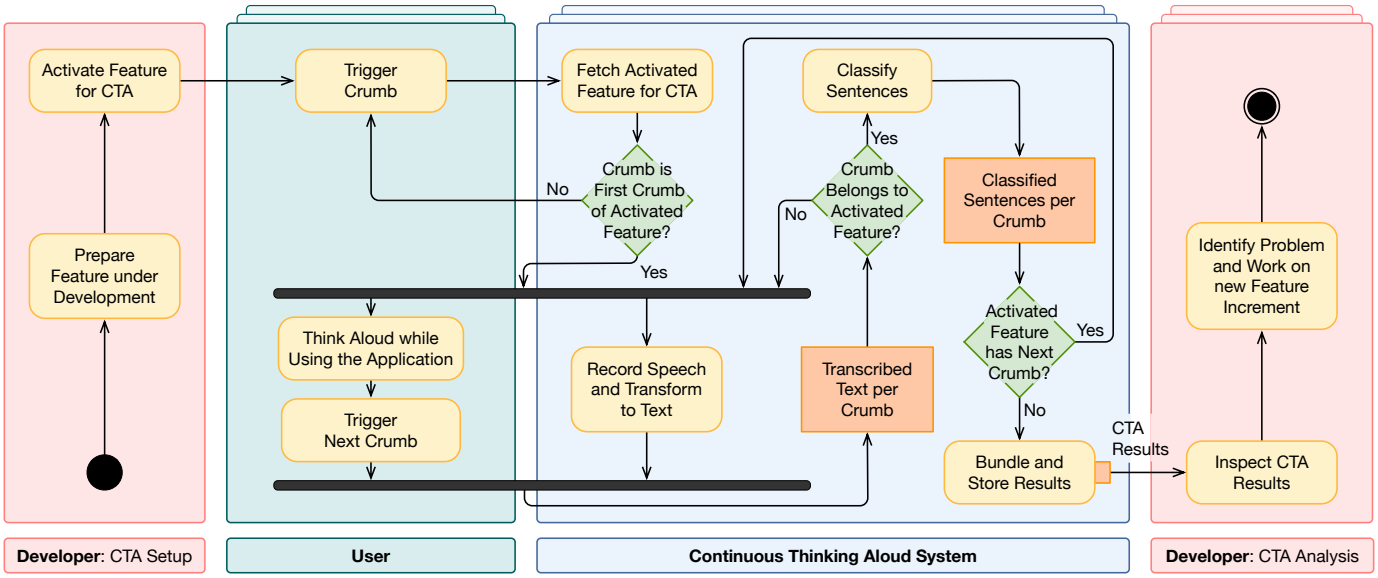

Fig. 1. An activity diagram showing actions and control flows during CTA application, under four perspectives performed by three actors: the developer during the CTA setup and analysis, a user of an application, and the Continuous Thinking Aloud system. Except for the setup, the activities of each actor can be repeated multiple times per feature increment, without the need for further manual interaction by the developer.

III. Continuous Thinking Aloud

The Continuous Thinking Aloud (CTA) approach adapts the traditional Thinking Aloud method to an automated setting that builds upon frequent and rapid software increments. Figure 1 outlines the main dynamic flow of a complete CTA session.

A. Developer Perspective: CTA Setup

To enable a feature for CTA, the developer needs to first *Prepare [a] Feature under Development* by adding *Feature Crumbs* to define the feature's interaction flow [9]. Feature crumbs, or *crumbs*, are a lightweight, code-based approach for describing steps in an interaction flow, similar to a breakpoint during source code debugging [9]. For example, a developer can add crumbs in the source code to reflect button actions or new screen view events. As a result, a new feature version is created consisting of a concatenation of multiple feature crumbs that describe a linear sequence of a feature.

The feature version and the related feature crumbs are defined once per commit. During the development on a branch, whenever new feature crumbs are added, the feature version is increased. Such scenario allows results from different commits to be compared later. In case at least two feature crumbs are added to a commit, the developer can add a flag to the related feature to *Activate [the] Feature for CTA*.

B. User Perspective

Application users, i.e., testers or end user, receive a new release containing a feature under development and start using it. As soon as they *Trigger [a Feature] Crumb* that is part of the previously defined feature, they are asked whether they would want to start participating in a CTA session.[1]

[1]In Fig. 1, we omitted user interface-related activities that are triggered by CTA in the application to maintain the readability of the activity diagram.

If they opt out, feature crumbs will be ignored for the current session and no data is recorded. However, if they agree, the CTA system will start recording their *Think[ing] Aloud while [they are] Using the Application*. A screen initially presented to inform users about the CTA session contains a reference to the feature that was detected and further provides suggestions for the type of speech feedback expected, given their unfamiliarity with Thinking Aloud.

Afterward, the users continue to *Trigger [the] Next [Feature] Crumb*, while they use the application and verbalize their thoughts. After the CTA system determines that the last feature crumb has been triggered, an information screen is shown to the users and the recording is stopped.

C. CTA System Perspective

The *CTA System* performs the core actions for automating the traditional Thinking Aloud method to enable the CTA approach. After a user initially triggers a feature crumb, the CTA system *Fetch[es the] Activated Feature for CTA* from a remote server hosting all features previously flagged by the developer, as described in Section III-A. The system initiates the CTA process if the *Crumb is [the] First Crumbs of [the] Feature*. Subsequently, CTA continuously *Records Speech and Transform[s it] to Text*. Whenever a new feature crumb is triggered and the *Crumb Belongs to [the] Activated Feature*, the resulting text is transmitted to a natural language classifier; if not, CTA proceeds to record speech and awaits the next crumb. On a sentence level, the classifier determines whether the text falls under insecure, neutral, positive-, or negative sentiment. Given the limitation of the available data set as described in Section IV, we restricted the classifier to four categories, while a more fine-grained differentiation may result in a better analysis process; the categories are detailed below.

Accepted at: Joint 5th International Workshop on Rapid Continuous Software Engineering and
1st International Workshop on Data-Driven Decisions, Experimentation and Evolution (RCoSE/DDrEE 2019), Copyright IEEE

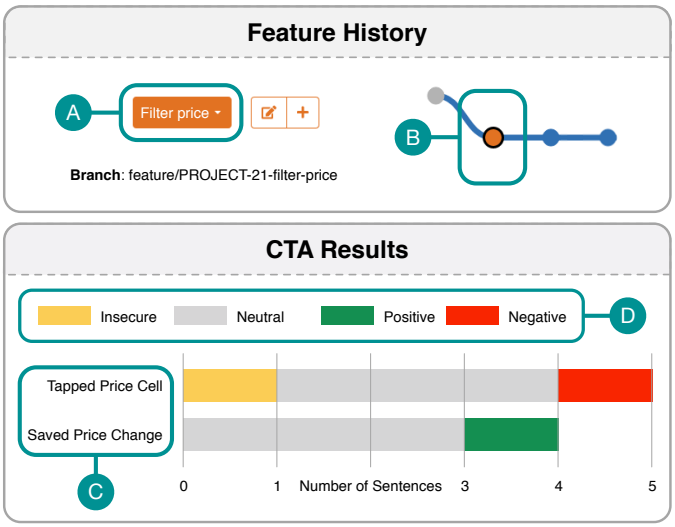

Fig. 2. A dashboard allowing control and assessment of CTA sessions.

- **Insecure** sentences express that users are unsure of what to do. Sentences describing insecurity often contain words such as "maybe" or verb forms such as the conjunctive.
- **Neutral** sentences contain impressions and actions: Impressions describe sentences reflecting the users' visual experience while using the application, which may include sentences as "All right, here I see a picture of a car and a button." Actions refer to sentences reflecting users' interactions within the application, which may include structures such as "Now I am going to tap on the button."
- **Positive** and **Negative** sentences form the most important sentiment and assess the current state of the application. For example, "I love the design of this date picker; it is very simple and intuitive!" expresses a positive sentiment, while "This button is useless; I do not understand what it is good for" indicates a negative attitude.

This procedure is repeated for as long as the *Feature has [a] Next Crumb*: In case the feature consists of more feature crumbs, CTA continues to record the speech. Otherwise, the *Classified Sentences per Crumb* are *Bundle[d] and Store[d]* by the CTA system and handed over to the developer.

D. Developer Perspective: CTA Analysis

After a CTA session becomes available, the developer can *Inspect [the] CTA Results*. Figure 2 sketches a visualization that allows access to the classification results in a widget.[2]

Initially, developers select a feature they wish to inspect in detail for the CTA results (Fig 2-A). This feature typically relates to a development branch, where a new feature is currently implemented. Next, the developers select a commit: In Fig. 2-B, the orange circle indicates a released commit with a first version *v1* of the feature and blue circles other commits.

[2]We developed a CTA system prototype that is integrated in the *Continuous User Understanding* (CUU) platform [10], which includes an interactive knowledge dashboard for developers. The core classes of the client-side CTA system are available as an open source project at https://github.com/cures-hub/cures-cuu-sdk/tree/master/CUU/Classes/ThinkingAloudKit.

A widget presents a bar graph of classification results separated by the feature crumbs that are part of feature v1 (Fig. 2-C). The x-axis shows the number of classification results (Fig. 2-D). For both feature crumbs listed in Fig. 2, there are three instances of recorded sentences classified with a *neutral* sentiment. In fact, this sentiment is expected to be the most prevalent, but can be omitted, since it is encouraged by Thinking Aloud, i.e., describing what one sees. More interestingly, the second feature crumb *Saved Price Change* contains one instance of a *positive* sentiment, indicating well that the latest improvements to the feature have fulfilled their purpose. Nevertheless, for the first crumb *Tapped Price Cell*, there is one instance each for *insecure* and *negative* sentiments, signifying that the developers need to revisit the related part of the application, i.e., where they previously added the feature crumb, to *Identify [the] Problem and Work on [a] new Feature Increment* to resolve issues in future releases.

IV. Discussion and Experience Reports

The overall **performance** of CTA depends on the accuracy of the underlying speech-to-text and classifier performance. We experienced that speech-to-text frameworks are not optimized for transcribing loosely coupled text fragments, although this is typical in Thinking Aloud. To create a classifier prototype for CTA, we had to rely on 10 manually collected and transcribed Thinking Aloud protocols, from which we used 200 sentences as the training data set. The small data set is a limitation of the current prototype: while it fulfills the purpose of a concept proof, a larger data set and more advanced classifier approaches would enhance the results. In addition, the choice of protocols needs to be further considered, as they directly influence the classifier results.

To test the **feasibility** of CTA, we collected various experience reports. First, we successfully integrated the CTA prototype into three mobile applications. Second, a comparison between five CTA and five traditional Thinking Aloud sessions with a total of 10 test subjects revealed that performing a CTA session is on average more than 3.5 times faster, excluding the time for transcription in case of the traditional Thinking Aloud. A reason might be the lack of a supervisor that could be consulted for questions. Third, we showed a visualization sketch similar to Fig. 2 to six developers, all of which were able to identify crumbs with potential usability problems. However, two crumbs did not contain any problem but were likewise considered relevant. This starts the discussion if more roles besides developers are required in the process, i.e., a dedicated role for both the CTA setup and the CTA analysis.

An **advancement** is achieved by the *continuous* aspect of CTA that leads to additional benefits as compared to the traditional Thinking Aloud method. The environment where users perform the CTA sessions is the actual target environment—a CSE principle enabled by continuous delivery. This increases the probability that users will provide additional and more relevant feedback. A potential bias introduced by a supervisor who is constantly observing the user is removed—which, to recall, was encouraged by Nielsen for avoidance [3].

Accepted at: Joint 5th International Workshop on Rapid Continuous Software Engineering and
1st International Workshop on Data-Driven Decisions, Experimentation and Evolution (RCoSE/DDrEE 2019), Copyright IEEE

V. Related Work

Since traditional usability testing exhibits low coverage of tested features due to its expensive application [11], *Remote Usability Testing*, divided into *synchronous* and *asynchronous* approaches [12], was established. During synchronous testing, user and supervisor might be geographically separated, but communicate during the test such that the supervisor observes and evaluates the user's behavior, e.g., through a virtual three-dimensional laboratory [13]. Asynchronous testing removes the need for a supervisor which results in time savings. CTA classifies as a remote asynchronous testing that leverages time benefits, while simultaneously overcoming the limitations of asynchronous testing highlighted by Bruun *et al.* [14].

Marsh *et al.* combine speech recordings with screen capturing for greater context [15]. In contrast, CTA realizes the context using feature crumbs, allowing faster analysis of speech results and better comparability between commits.

Guzman *et al.* visualize user feedback with a focus on user emotions [16]. CTA uses similar approaches for both processing text-based feedback and visualizing the results. However, CTA focuses more on qualitative aspects of feature increments and thereby addresses the high frequency of CSE.

Ferre *et al.* describe the extension of an analytics framework to record user interactions for usability evaluation [17]. CTA relies on a similar concept to describe a task under investigation. However, while Ferre *et al.* focus on usage logs analysis, CTA additionally utilizes speech to identify usability defects.

VI. Future Work

Current, the CTA system only provides access to the classification results. We envision to show the full transcripts of a CTA session to provide greater context to developers. This preserves the richness of feedback that Thinking Aloud provides in the first place and further extends the advantages of CTA, as the user feedback could be inspected on both a high- and a fine-grained level. Furthermore, we will evaluate whether to add assistance in case a user is stuck during a feature execution. Similar to a traditional Thinking Aloud session, in which an observer can step in to provide support, a text-to-speech component of the CTA system could read out meta data of the next feature crumb. Finally, a combination with other knowledge sources would increase the effectiveness of CTA: Provided a trigger in case a user is confused could prevent requesting feedback from each user and only address users in situations when usability problem deem to occur.

VII. Conclusion

Rapid development approaches, such as CSE, rely on user feedback for continuous improvement a software increment. The Thinking Aloud method by Jakob Nielsen [3] enables the collection of expressive user feedback. However, its manual character hinders its application in CSE. To automate the traditional Thinking Aloud method, we introduced the *Continuous Thinking Aloud* approach. CTA adds scalability and cost efficiency to Thinking Aloud while it maintains the method's benefits. We described an integrated action flow that enables CTA application in CSE for increased acceptance by users. Furthermore, CTA enables developers to benefit from multiple Thinking Aloud sessions simultaneously, without the need to manually supervise, transcribe, and analyze the sessions, practically minimizing their efforts during the CSE process.

Acknowledgment

This work was supported by the DFG (German Research Foundation) under the Priority Programme SPP1593: Design For Future – Managed Software Evolution (CURES project). We thank the participants for their experience reports.

References

[1] J. Bosch, *Continuous Software Engineering: An Introduction*. Cham: Springer International Publishing, 2014.

[2] B. Fitzgerald and K.-J. Stol, "Continuous software engineering: A roadmap and agenda," *Journal of Systems and Software*, vol. 123, pp. 176 – 189, 2017.

[3] J. Nielsen, *Usability Engineering*, ser. Interactive Technologies. Elsevier Science, 1994.

[4] K. Schneider, "Focusing spontaneous feedback to support system evolution," in *2011 IEEE 19th International Requirements Engineering Conference*, Aug 2011, pp. 165–174.

[5] D. Saff and M. D. Ernst, "Reducing wasted development time via continuous testing," in *14th International Symposium on Software Reliability Engineering.*, Nov 2003, pp. 281–292.

[6] M. van Someren, Y. Barnard, and J. Sandberg, *The Think Aloud Method: A Practical Guide to Modelling Cognitive Processes*. London: Academic Press, 1994.

[7] C. Lewis, P. G. Polson, C. Wharton, and J. Rieman, "Testing a Walkthrough Methodology for Theory-based Design of Walk-up-and-use Interfaces," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '90. ACM, 1990, pp. 235–242.

[8] A. Holzinger, "Usability Engineering Methods for Software Developers," *Communications of the ACM - Interaction design and children*, vol. 48, no. 1, pp. 71–74, Jan. 2005.

[9] J. O. Johanssen, A. Kleebaum, B. Bruegge, and B. Paech, "Feature Crumbs: Adapting Usage Monitoring to Continuous Software Engineering," in *19th International Conference on Product-Focused Software Process Improvement.*, M. Kuhrmann, K. Schneider, D. Pfahl, S. Amasaki, M. Ciolkowski, R. Hebig, P. Tell, J. Klünder, and S. Küpper, Eds. Cham: Springer International Publishing, 2018, pp. 263–271.

[10] J. O. Johanssen, "Continuous user understanding for the evolution of interactive systems," in *Proceedings of the ACM SIGCHI Symposium on Engineering Interactive Computing Systems*, ser. EICS '18. ACM, 2018, pp. 15:1–15:6.

[11] M. Y. Ivory and M. A. Hearst, "The State of the Art in Automating Usability Evaluation of User Interfaces," *ACM Computing Surveys (CSUR)*, vol. 33, no. 4, pp. 470–516, Dec. 2001.

[12] J. Scholtz, "Adaptation of Traditional Usability Testing Methods for Remote Testing," in *Proceedings of the 34th Annual Hawaii International Conference on System Sciences*. IEEE Computer Society, 2001.

[13] K. Chalil Madathil and J. S. Greenstein, "Synchronous Remote Usability Testing: A New Approach Facilitated by Virtual Worlds," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '11. ACM, 2011, pp. 2225–2234.

[14] A. Bruun, P. Gull, L. Hofmeister, and J. Stage, "Let Your Users Do the Testing: A Comparison of Three Remote Asynchronous Usability Testing Methods," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '09. ACM, 2009, pp. 1619–1628.

[15] S. L. Marsh, J. Dykes, and F. Attilakou, "Evaluating a Geovisualization Prototype with Two approaches: Remote Instructional vs. Face-to-Face Exploratory," in *Tenth International Conference on Information Visualisation (IV'06)*, July 2006, pp. 310–315.

[16] E. Guzman, P. Bhuvanagiri, and B. Bruegge, "Fave: Visualizing user feedback for software evolution," in *2014 Second IEEE Working Conference on Software Visualization*, Sept 2014, pp. 167–171.

[17] X. Ferre, E. Villalba, H. Julio, and H. Zhu, "Extending mobile app analytics for usability test logging," in *Human-Computer Interaction – INTERACT 2017*. Springer, 2017, pp. 114–131.

Accepted at: Joint 5th International Workshop on Rapid Continuous Software Engineering and
1st International Workshop on Data-Driven Decisions, Experimentation and Evolution (RCoSE/DDrEE 2019), Copyright IEEE