KNOCAP: Capturing and Delivering Important Design Bits in Whiteboard Design Meetings

Adriana Meza Soria

Informatics

University of California Irvine

amezasor@uci.edu

ABSTRACT

It is well known that it is desirable to capture the most essential parts of software design meetings that take place at the whiteboard. It is equally well known, however, that actual capture rarely takes place. A few photos may be taken, informal notes might be scribbled down, and at best one of the developers may be tasked with creating a summary. Regardless, problems persist with important information being lost and, even when information is captured, that information not being easily located and accessed. To address these problems, I propose to design and evaluate a novel suite of tools that enables software designers working at the whiteboard to: (1) efficiently and in-the-moment capture important information produced during that meeting, and (2) be delivered, either by request or proactively by the tools, relevant information captured in the past when it is needed in a future design meeting.

CCS CONCEPTS

• Information systems • Information retrieval • Users and interactive retrieval • Search interfaces

KEYWORDS

Design meetings, important design bits, design knowledge, design rationale

ACM Reference format:

Adriana Meza Soria. 2020. KNOCAP: Capturing and Delivering Important Design Bits in Whiteboard Design Meetings. In *Proceedings of ACM 42nd International Conference on Software Engineering Companion (ICSE '20 Companion)*, May 23–29, 2020, Seoul, Republic of Korea, 4 pages.

1 Problem Statement

Consider the following scenario as depicted in Figure 1.

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

ICSE '20 Companion, May 23–29, 2020, Seoul, Republic of Korea

© 2020 Association for Computing Machinery.

ACM ISBN 978-1-4503-7122-3/20/05 \$15.00

<https://doi.org/10.1145/3377812.3381397>

Three developers—David, Adi, and Ela—are colleagues, who are tasked with redesigning the communication software that connects two geographically distributed offices. What is special about the situation is that, for security reasons, the two offices are connected with a proprietary, dedicated communications line that was installed many decades ago. The software, however, has become brittle and very difficult to maintain. To start their new task, Adi schedules a design meeting in one of the conference rooms and invites his two colleagues to attend. They are working at the whiteboard, having decided to pursue TCP/IP, and in the throes of designing their solution, when Joe – nearly retired, but still walking the hallways happy to talk to anyone – happens to walk by, overhears what they are talking about, turns around, sticks his head into the office, announces “The designers used UDP because the actual communications link is too slow to support TCP/IP; messages time out because the acknowledgment that a message is received takes too long to arrive back with the sender, causing TCP/IP to fail”, and walks off, leaving them with the realization that Joe just saved them from making a serious blunder and that they have to fully reconsider their design.

Paraphrased, the goal of my proposed research is to create new technology that can take on the role of Joe (and the many other Joe’s and Jill’s who exist) when he is not available – whether he is just not walking by at the right time, is busy and thus cannot be reached at the moment he is needed, has moved to a different job elsewhere, or has retired and no longer works. This represents a two-part challenge. On one hand, it means that

Figure 1: A typical whiteboard design meeting.

Figure 2: A whiteboard meeting augmented with smart capture and delivery of important design bits.

critical information has to be captured during design meetings, so it is available for later use. On the other hand, it means that captured information that is relevant in a future meeting is actually recognized and delivered in that meeting (as depicted in Figure 2).

2 Approach

My proposed work centers on the following research question:

To what degree is it possible to offer software designers tool support with which they can capture important design bits in a collaborative, lightweight manner, and be delivered relevant such important design bits in a (semi-)automatic manner in future whiteboard design meetings?

In this question, the term *important design bits (IDBs)* covers anything that might be considered imperative to future meetings, which could include decisions, alternatives considered, rationale, assumptions, constraints, stakeholder goals, future to do's, and more.

A critical choice is for IDBs to be small snippets of spoken language. Rather than capturing video, which is both difficult to capture well and difficult to process meaningfully afterwards, technological advances in capturing and understanding audio, machine learning, and conversational interfaces offer a timely opportunity: to capture small slices of spoken language, catalogue them, and (in near real time) analyze whether any of the captured slices could be relevant to an ongoing conversation. Each IDB, then, represents a single, short audio snippet that captures something that was critical to the progress of a design meeting and could be equally critical to a future design meeting. An IDB, then, could be a 10 second recap of an important decision, a 2 minute articulation of why one of the discussed alternatives is better than another, or a 43 second mention of an idea that is not relevant now but may be later.

A second critical choice is human-machine partnership: my research does not seek to be entirely automated, as which IDBs developers want to keep is going to be subjective. While certain amounts of commonality will likely exist that may be explored by machine learning techniques (e.g., the way decisions are worded, the way a question from one developer interrupts a stream of thought from another), what is or is not important is domain, context, and developer dependent. In a similar vein, if a tool continuously offers up all potentially relevant IDBs in a meeting, it would very quickly be seen as too disruptive. Thus, for both capture and delivery, it is important that my approach establishes a fruitful but unobtrusive human-machine partnership.

The final critical choice pertains to my approach being collaborative. Putting the responsibility of capturing IDBs on a single developer renders it difficult for them to meaningfully participate in a meeting. Moreover, it would provide a singular point of view as to what is captured as important. Therefore, my approach offers the ability to capture IDBs to all participants in a meeting. Equally, we plan to create tools through which IDBs can be retrieved not just by the person at the whiteboard, but also by other participants in a meeting.

3 KNOCAP

To explore my research question, I am designing KNOCAP (short for Knowledge Capture), a suite of tools that together cover *capture*, *organization*, and *delivery* of IDBs. Below, I briefly sketch each of these three functionalities.

3.1 Lightweight, unobtrusive capture

It is crucial that designers are interrupted as little as possible when they are designing, as they otherwise would lose their flow. Yet, it is equally important that developers have at least some control over what is captured. My approach is based on the simple observation that, while it is difficult for developers to anticipate when something important is going to be said next, it is much easier for them to recognize when something important was said in hindsight and especially 'just now'. Leveraging this fact, I am designing two interfaces through which a developer can quickly signal that something important just happened:

1. *An interface on the electronic whiteboard consisting of simple buttons that, with a single tap, capture what was just said. One button captures 30 seconds; a second 60 (these lengths will be adjusted based on studies I am conducting now).*
2. *A mobile UI that consists of a single button with an integrated sliding action to decide upon the length of the audio that is to be captured. Tapping the button, holding the finger down, and then sliding it up extends the time beyond the default (30 seconds) up to 2 minutes.*

The first interface is important because it gives the developer at the whiteboard control without needing to use another device. Also, the interface is intentionally simple in not requiring so-

phisticated actions, so it can be a very quick diversion and lessen the chance the developer losing their train of thought. The second is important for others in the meeting, as they clearly would not want to interfere by having to go to the whiteboard and tap. It makes capture truly collaborative, and furthermore introduces a very subtle but important capability: when multiple developers signal a particular moment should be captured as an IDB, it implicitly gives weight to that moment – it might be more important than other IDBs that only one developer attempted to capture.

I plan to explore other interfaces as well. One, in particular, will explore machine learning. Assuming I am able to collect a large set of IDBs, it may be possible to train a machine learning model to start providing suggestions for when something should be captured. The buttons on the whiteboard could light up, for instance, or the mobile phone app could buzz, and share “Tap to capture a new IDB, 1 ½ minutes in length.” This solution would represent two important advances: (1) it would regularly remind developers of the need to capture IDBs, and (2) it would merely need them to confirm rather than to be the ones to have to think about when to capture. This solution is longer-term than the others, but an important part of my planned research.

3.2 Intelligent storage

To enable future use of IDBs, KNOCAP stores not just the raw audio snippets, but actually processes them to make them available to the next stage in a convenient manner. Modern advances in speech understanding serve a critical role here. I have implemented automated transcription (by sending IDB voice snippets to Google cloud speech-to-text API), but I also plan to explore:

1. *Indexing to provide the ability to quickly search for potentially relevant IDBs.* Once transcribed into text, indexing is a matter of applying standard technology to build a quick index. However, I will build a richer index. Particularly, I plan to index who said what, when it was said in the meeting, as well as various keywords. The latter can be done generically, but I anticipate exploring ways of surfacing domain knowledge so that IDBs can be searched based on key concepts in the discussions.
2. *Classification to enable the creation of useful metadata surrounding the IDBs.* I am interested in labeling IDBs with the type of contribution they make to a discussion (e.g., a decision, a constraint, an assumption). Such metadata would enable future queries that concern the logic and flow of past discussions, which is how developers tend to remember. My advisor already has a dataset of past design meetings from the SPSPD workshop¹. In addition, I am collecting data from whiteboard design meetings. I will use both sources as the basis for training a classifier.

¹ <https://www.ics.uci.edu/design-workshop/>

3.3 Relevant delivery

The final component of KNOCAP is the least developed to date, but no less important as it is *the* reason why I am engaged in this research: existing tools typically stop short of delivering information captured in the past, but it is exactly here that a major opportunity lies in taking advantage of today’s speech understanding and conversational interface technology. My ultimate goal is for KNOCAP to deliver exactly the right IDBs, at the right time, to the right people, in the right meetings. I plan to explore two avenues to doing so:

1. *A reactive conversational agent, through which developers can query for specific IDBs.* I plan to develop a query language and dialog model specifically for this purpose, leveraging the intelligent storage of IDBs so to leverage the metadata and classifications to support the dialog. I plan to use Google Assistant as the platform upon which to implement this.
2. *A proactive conversational agent, which independently listens to the ongoing design discussion of the developers, interprets the discussion, relates it to past IDBs, and offers up suggestions for potentially relevant IDBs that can then be further explored (see Figure 2).* It is critical that this proactive agent does not overwhelm the designers. To this end, I will explore a range of possibilities, from icons on the board that indicate an IDB is available, to brief verbal cues spoken softly (“I have an IDB available related to the choice of UDP.”).

With both the reactive and proactive agents, I foresee brief dialogues through which developers can ask such questions as “What other IDBs cover this topic?”, “Did anyone talk about this later?”, “Were any alternatives offered at the time?”, and so on.

4 Related Work

A long history exists of tools within the same broad space as KNOCAP. Early tools such as Filochat [12], Liteminutes [1], and The Audio Notebook [8], for instance, fully recorded meetings so to avoid interrupting developers altogether. This, then, meant that the stream that was captured needed to be post-processed, for instance by indexing the stream via keywords that were pre-specified by users. Design rationale capture tools (e.g., [3,7]) equally attempt to support capture of information for later use, often by promoting a particular notation with an associated GUI that makes it easier for developers to capture decisions, constraints, and so on, but also places the design in somewhat of a straitjacket – little flexibility is offered by the tools, since a methodology needs to be followed. Other strands of work focus on topics such as design reuse [13], architectural knowledge [10,11], and domain knowledge [2], with tools that support developers in capturing this information. Typically, however, such capture would not be done during a design meeting, but afterwards, based on what the developers remembered that they then formalized in these tools.

5 Evaluation Plan

For KNOCAP evaluation, I am planning a series of three experiments:

1. *IDB capture.* In a first experiment, I will focus on IDB capture. The experiment will take the form of a design meeting at the whiteboard, where I will task a group of developers with designing a traffic simulator (as used in similar studies, e.g., [4], [9]). The experiment will focus on questions such as: (1) what IDBs do developers capture, (2) what IDBs should they have captured but did not, and (3) which specific interface do they prefer to use. I will study various conditions, including asking everyone in a group to participate in both design and capturing IDBs through KNOCAP, versus having one person assigned as notetaker using pen and paper, versus having one person assigned as notetaker using KNOCAP. Additionally, I will interview participants after they complete their design to understand what worked and what did not work for them.
2. *IDB delivery.* In a second experiment, I will focus on delivery of IDBs. The experiment will again take the form of a meeting at the whiteboard, with the same design prompt as the first experiment. This time, however, I will use the Wizard of Oz method to manually interject (hypothetical) past IDBs into the design meeting in order to test developers' reactions and openness to such a mechanism. I will once again vary conditions, with IDBs being delivered via small bubbles on the screen, versus via voice, versus via delivery to a specific person on the team whose role is to identify and record IDBs that might be useful to the design at hand. Again, I will interview the participants afterwards to understand their attitudes.
3. *Roundtrip.* Based on the lessons learned from both experiments, I plan to refine and complete KNOCAP, so all of its envisioned functionality is available (e.g., machine learning assistance, classification, additional delivery mechanisms). Then, I will perform a third experiment to look at developers designing across a series of meetings, exactly the kind of scenario at which KNOCAP is aimed. Depending on the opportunity that arises, I may perform this experiment in the context of a design course or by bringing KNOCAP to an industrial partner organization, where actual professional software developers would use it. In this experiment, I will again study the kinds of IDBs that are and are not captured, whether and how developers access past IDBs, and the overall utility and effectiveness of KNOCAP. I will once more use interviews to understand the perceptions of the participants.

6 Current State

Overall, KNOCAP occupies a unique space among the pre-existing research and industrial tools: it combines lightweight interactions with advanced speech understanding and conversational agent technology to deliver a lightweight and collaborative experience that seamlessly integrates capture, smart storage, and delivery of IDBs when so needed. I have already published a preliminary paper describing my envisioned lightweight and un-

obtrusive capture of design information [5]. At present, I am focusing on the evaluation of a preliminary version of the whiteboard and mobile UIs, as well as on developing the knowledge delivery tools that I plan to use for the second experiment.

7 Timeline for Completion

Table 1: Timeline for completion

Year	Activities
Previous work	
2018	– Defining research topic and literature search [6].
2019	– KNOCAP lightweight voice notes collector development [5]. – KNOCAP whiteboard and mobile UI development.
Future work	
2020	– IDB capture evaluation using the whiteboard and mobile UI. – Augmentation of whiteboard UI to integrate a mock mechanism able to simulate IDB delivery (voice and text-based). – IDB delivery evaluation using the augmented whiteboard UI.
2021	– KNOCAP redesign. – Empirical study.

REFERENCES

- [1] Chiu, P., Boreczky, J., Girgensohn, A. and Kimber, D. 2001. LiteMinutes: An Internet-Based System for Multimedia Meeting Minutes. *In Proc. Tenth World Wide Web Conference*, 140–149.
- [2] Evans, E. 2004. Domain-driven Design: Tackling Complexity in the Heart of Software. Addison-Wesley Professional.
- [3] Falessi, D., Cantone, G. and Becker, M. 2006. Documenting design decision rationale to improve individual and team design decision making: an experimental evaluation, 134–143.
- [4] Mangano, N., LaToza, T.D., Petre, M. and van der Hoek, A. 2014. Supporting Informal Design with Interactive Whiteboards. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 331–340.
- [5] Meza Soria, A. and van der Hoek, A. 2019. Collecting Design Knowledge Through Voice Notes. *Proceedings of the 12th International Workshop on Cooperative and Human Aspects of Software Engineering* (Piscataway, NJ, USA, 2019), 33–36.
- [6] Meza Soria, A. and van der Hoek, A. 2018. Toward Collecting and Delivering Knowledge for Software Design at the Whiteboard. *Proceedings of the 11th International Workshop on Cooperative and Human Aspects of Software Engineering* (Gothenburg, 2018), 108–109.
- [7] Moran, T.P. 1996. *Design Rationale: Concepts, Techniques, and Use*. L. Erlbaum Associates Inc.
- [8] Stifelman, L., Arons, B. and Schmandt, C. 2001. The Audio Notebook: Paper and Pen Interaction with Structured Speech. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 182–189.
- [9] Shaw, M. 2012. The Role of Design Spaces, in *IEEE Software*, vol. 29, no. 1, pp. 46–50.
- [10] Tang, A., Avgeriou, P., Jansen, A., Capilla, R. and Ali Babar, M. 2010. A comparative study of architecture knowledge management tools. *Journal of Systems and Software*. 83, 352–370.
- [11] Tang, A., Jin, Y. and Han, J. 2007. A rationale-based architecture model for design traceability and reasoning. *Journal of Systems and Software*. 80, 918–934.
- [12] Whittaker, S., Hyland, P. and Wiley, M. 1994. FILOCHAT: Handwritten Notes Provide Access to Recorded Conversations. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 271–277.
- [13] Woodfield, S.N., Embley, D.W. and Scott, D.T. 1987. Can Programmers Reuse Software? *IEEE Software; Los Alamitos*. 4, 52–59.