Exploring a Research Agenda for Design Knowledge Capture in Meetings

Liz Seero, Janet Burge
Mathematics and Computer Science
Colorado College
{l_seero, jburge}@coloradocollege.edu

Adriana Meza Soria, André van der Hoek
Informatics
University of California Irvine
{amezasor, andre}@uci.edu

***Abstract*—Meetings are a frequent part of life for a software developer. Software design is often performed, discussed, and reviewed in these meetings. This means that meetings may contain important design information that could be captured for later use. Meeting design tools may be a way to capture design information as a byproduct of discussion that arises in these meetings. In this paper, we identify a list of key meeting support tool features that could support the capture and retrieval of design information and compare these to features currently offered in commercial meeting support tools.**

***Index Terms*—software design, design meetings, design rationale, tools**

I. Introduction

In software development, meetings are important for planning and decision-making, as well as information dissemination. Employees typically spend a third of their time in meetings [1].

Consider this scenario, which describes the meetings held for one of the projects developed by a major Healthcare Information Systems company that was studied in [2]. Meetings held for this project include Backlog Refinement (twice a week), Architecture Committee meetings (twice a week), UI Initiative meetings (bi-weekly), Sprint Planning (bi-weekly), ad-hoc Sprint Work meetings, Daily Stand-ups, and Retrospectives. A key observation obtained from analyzing the Architecture Committee meetings of this team, in particular, was that the meetings were an important venue for information dissemination. In fact, the lead architect confirmed researcher observations that many attendees were present with the primary goal of staying informed. Other than notes taken as a direct consequence of the task at hand, a lot of what was talked about and absorbed by participants was entirely verbal.

The challenge of capturing information about design, particularly the decisions made and their rationale, is one that has been studied for many years, yet no approaches or tools have received significant adoption in industry. Meetings are a potential source of this information, as design issues are discussed and decisions made. With the shift to more remote or hybrid work, accelerated by the global pandemic, there has been an increase in development and adoption of meeting support tools. This raises the question of how this might impact the future of Design Rationale (DR) capture and use.

This paper contributes the following towards forming a research agenda on capturing and accessing DR:

1) Providing a list of meeting features that can aid in supporting design.
2) Identifying which of these features are currently supported by commercial meeting support tools.

We will conclude the paper with a discussion of these features and plans for implementing our research agenda.

This material is based upon work supported by the National Science Foundation under grant CCF-2210813, 2210812.

II. Related Work

A. Design Rationale

Design Rationale has been an active area of research for over 50 years, starting with Kunz and Rittel's seminal work on IBIS [3]. Rationale has been studied and applied to many types of design, including Software [4], Human Computer Interaction [5], and Engineering [6]. Much of the initial work focused on notations and tools for the capture and representation of argumentation-format rationale [7], [8], [9], [10], [11], [12], [13], [14]. More recently, there has been work that attempts to extract rationale from existing documents [15], [16], [17], [18], [19], [20], [21]. One of the challenges with retrospective capture of rationale is the possibility of large amounts of information that requires processing before it can be useful. Shipman and McCall recommended a process of "incremental formalizing" where information was structured as needed rather than all at once [22].

B. Software Design Meetings

How software developers engage in collaborative design meetings has been a subject of extensive study. The focus of these studies has been varied, ranging from analyzing the sketches that software developers use as the anchor for their discussions (e.g., [23], [24], [25], [26], [27]) and understanding how they deal with uncertainty (e.g., [28], [29]) to the role of novices and experts (e.g., [30], [31]) and problem solving strategies (e.g., [32], [33]). There are also papers that are not explicitly about design meetings but that use meetings as a way to understand more about software development, such as Lavalee, et al. [34].

C. Meeting Support Tools

Later in the paper we will discuss some commercially available tools. In addition to these, there are also several tools developed as research projects. Calo Meeting Assistant [35] transcribes meetings and uses Machine Learning to perform segmentation, tagging, and automatic action item detection. It also performs meeting summarization. Talk Traces [36] captures and visualizes meeting discussions. This includes word cloud generation and topic visualization. Community Click [37] transcribes meetings and uses custom clickers to capture feedback from participants (agree, disagree, confused, unsure, important). Organizers can use a clicker to tag parts of the meeting (Main Idea, New Issue, etc.) while participants can provide opinions (Agree, Disagree, etc.). Meeting Vis [38] processes a meeting transcript to create a meeting summary. It provides an interactive visualization that identifies speakers and discussion topics. MemTable [39] is a tabletop interface that allows those seated to collaborate during meetings. The table records audio as well as any actions taken using the table interface. KnoCap [40] was developed to support software design meetings by incorporating a shared whiteboard and allowing designers to capture segments of the meeting transcription as "important design bits" so that they can be accessed later.

III. Software Design Meetings

There are many different kinds of meetings that can occur during software development, including stand-up meetings, design critiques, problem-solving sessions, ticket triage, retrospectives, and many more. Not all of these may involve design. Some may have design as a primary activity, others may have design occur only occasionally. The types of meetings held can often be company specific. When design occurs is different in software development than in other disciplines because in a way, it never ends [41]. Design can occur early in the process (for example, user interface design sometimes is done as part of requirements definition because it helps to clarify how the system will work from a user perspective) as well as very late in the process, as software updates performed during maintenance to add new features and correct defects will also require design. What is being designed varies as well—there may be different needs when sketching out a user interface design versus designing the software architecture. The malleability of software means that design is much more distributed across time than it would be in manufacturing, where there is not the ability to modify a design after the artifact has been built. At any point, decisions need to be made as to how to solve various design issues.

Design can occur across meetings—both from consecutive recurring meetings of the same type with the same team, or across meetings of different types with potentially different meeting attendees. Duffy and O'Rourke [42] referred to the concept of "meeting streams"—meetings held across an organization involving different teams that contributed towards developing a product. If these meetings do not include the same personnel, it is possible, if not likely, that critical information may not be successfully transmitted from meeting to meeting.

Our first research contribution is to identify what kind of functionality high-level meeting support tools need to offer in order to support software design meetings, and specifically the capture of design information. As a first step in this direction, we looked at meeting support features in general, as obtained through a combination of an analysis of features in existing meeting tools (commercial, open source, and research) and our own prior research history and experience. We started with brainstorming a list of features and then added to them as we performed a competitive analysis of existing tools. Table I presents the resulting list of meeting support tool features. We grouped these features by Meeting Stage, using the stages identified by Bedingfield and Clarkson [43]:

- *Inception (InC)*—the initial creation and planning of the meeting
- *Initiation (Init)*—attendee transition from activities they were doing before the meeting into meeting participation
- *Meeting Event (ME)*—the meeting itself
- *Leverage (L)*—capturing information during and after the meeting and using that information to achieve meeting outcomes.

Initiation is likely to overlap with the start of the meeting, while Leverage will occur both during- and post-meeting.

This full list of features is lengthy. We would like to focus our research on those that more specifically support the capture of design information using the following criteria:

1) Features needed to capture design artifacts, such as sketches or diagrams;
2) Features needed to capture design deliberation and decision-making;
3) Features needed to support collaboration with different stakeholders to obtain design requirements; and
4) Features needed to connect meetings in cases where design discussion is fragmented. Meetings are usually bounded in time, which means design activities may be need to occur across multiple meetings.

The subset of meeting features that fall into these categories is highlighted in bold type in Table I.

IV. Existing Meeting Support Tools

Our second research contribution is to examine if current tools support the needs of software design meetings. To do this, we looked at existing tools to see which ones implement the critical features highlighted in the previous section.

Meeting support tools can be grouped into five main categories:

- *Computer Mediated Communication tools*—tools such as Zoom and Microsoft Teams that combine video conferencing and messaging to conduct virtual and hybrid meetings.
- *Automated Transcription/Summarization tools*—tools that transcribe and, in some cases, summarize meetings or conversations.

TABLE I
MEETING TOOL FEATURES

Feature	Definition	Stage
Pre-Meeting Request	Support for querying participants on what they would like to see discussed in the meeting	InC
Create Agenda	Create an agenda for the meeting that will be available to participants before and during the meeting	InC
Pre-Set Meeting Tags	Set up a common vocabulary for tags (such as topics) that participants can use. This can involve re-use of existing tags or new ones	InC
Associate Tags with Meeting Agenda	Assign tags to agenda items	InC
Create Meeting Template [New or Previous Meeting]	Create a template that can be used to create meetings in the future	InC
Create Meeting from Template	Use a meeting template to create a new meeting. This would include attendees and tags	InC
Create Meeting	Create a new meeting	InC
Meeting Materials [Preparation and Reference]	Associate documents with a meeting that are provided for attendees to read prior to the meeting or to refer to during the meeting	InC
Link Meetings [Full Meeting; Selected snippets]	Adding a reference to prior meetings on the same topic to the Meeting Materials if participants are expected to be familiar with what was discussed earlier	InC
Link External Tools [Slack, Ticketing Systems, Google Docs, . . .]	Add a reference to external tools that contain information relevant to the meeting	InC
Specify Meeting Host	Indicate who will be the host (or hosts) while the meeting is occurring	InC
Import Invitee List from Previous Meeting	Create meeting attendees for a meeting from a previous meeting	InC
Edit Participant List	Add and delete attendees from the meeting	InC
Invite and Notify	Invite Participants and Notify attendees that the meeting has been scheduled	InC
Mark attendance	As participants join a meeting, their attendance is captured so that we know who was actually at a meeting, rather than only who was invited	ME
Advance agenda	The host can highlight agenda items as we move through the meeting	ME
Off-Topic Timer	The host can indicate when discussion has gone off-topic and a timer will be displayed showing how long the discussion lasts	ME
Collaborative Notes	Take notes during the meeting collaboratively with other attendees. And attaching to other artifacts	ME
In-Meeting Review	If an attendee joins a meeting late or needs to step out and rejoin they will still have access to transcripts, snippets, etc. for the meeting in progress where they can browse back to quickly review	ME
Whiteboard Sketching	Meeting attendees can sketch on a shared whiteboard	ME
Whiteboard Capture and Tagging	Meeting attendees can capture whiteboard contents. Each capture needs to have at least one tag	ME
Transcribe Meeting	Automatically transcribe meeting discussion	ME
Authenticate Access	Determine if someone has visibility into meeting contents	ME, L
Edit Meeting Access	Add or remove meeting access for personnel	ME, L
Capture Snippet	Capture a meeting snippet. By default, this must include one tag. This will capture the transcript and associated audio	ME, L
Edit Snippet	A snippet can be edited to extend it if not everything was captured	ME, L
Tagging [Discussion Status, Topics, Follow-up, Comments]	Add an informative tag to a meeting artifact [Snippet or Whiteboard]	ME, L
Link External Tools [Slack,ticketing, etc.]	Link external tools that have relevant information to the system or that need to be updated based on discussion	ME, L
Personal Notes	Take notes during or after the meeting that are not shared with other attendees	ME, L
Follow-up Request	If an snippet or white-board capture is given a follow-up tag, the person tagged will be notified via some messaging service (such as e-mail or Slack) when the tag is made	ME, L
Access Prior Meeting [Snippets, Tags, Notes]	People can access meetings to view what happened; add additional information (tags, notes, etc.); and make personal notes on the meeting	ME, L
Group Messaging	The host can message attendees	ME, L
Meeting Summary	Provide a summary of important points	ME, L
Search Meetings [Time Frame, Topic, Person/Team Attending, Person Speaking, Meeting Type, Meeting Series, Person Tagged, Transcribed Text, Comment Text]	Search across the whole collection of meetings (or at least those the searcher has access to	ME, L
Search Within Meeting [Topic, Person Speaking, Person Tagged, Transcribed Text, Comment Text]	Search within a meeting, either one that is in-progress or viewed	ME, L
Quick Text Search	Search for a word or words occurring in a meeting transcript, tags, or notes	ME, L
Go To Snippet Context	If someone is viewing a meeting snippet, they can bring up the meeting transcript to see the information in context	ME, L
Add Past Meeting Link	A link to an snippet from a prior meeting can be added to a captured snippet in the current one	ME, L
Add Future Meeting Link	A link to an snippet from this meeting can be added to a future meeting, if that meeting has already been created	ME, L

- *Meeting Support Tools*—tools used during meetings to provide structure (i.e., agendas) and/or capture meeting information.
- *Project Management Tools*—tools that provide some form of meeting support in addition to broader project management support such as scheduling and workflow.
- *Collaborative Note-taking tools*—tools such as Google Docs that can be used to take notes during meetings but also have wider applications.

There are many tools that can be, and are, used in meetings. For the purpose of this evaluation, we looked at three criteria:

1) Tools should understand the concept of a meeting as an event with a fixed duration.
2) Tools should provide value during the meeting event itself (versus only providing a summary after).
3) Tools should offer features that make it easier to leverage meeting results after the meeting is over and for people not at a meeting to catch up on what happened.

As a result of this, we did not look at tools like Zoom or Google Docs despite their frequent use in meetings.

We focused on commercially available tools (research tools are discussed under related research). Our search for tools was primarily opportunistic, with tools found through Google search, although we also consulted a list of tools provided by the Collaboration Superpowers Website [44]. The following tools were analyzed:

- *Butter (B)* - Butter (https://www.butter.us) allows meetings to be set up with an agenda. It captures recordings, chat logs, and personal meeting notes and allows easy access to all past meetings.
- *Docket (D)* - Docket (https://www.dockethq.com) supports agendas and action items. It also keeps track of a recurring meeting history so users can review past meetings. Unfortunately Docket is no longer available.
- *Fellow.app (Fe)* - Fellow (https://fellow.app) allows meeting participants to work from a shared collaborative agenda during a meeting. It includes a meeting timer and supports action items and tagging.
- *fireflies.ai (Fi)* - fireflies (https://fireflies.ai) provides meeting transcription. Participants can tag portions of the meeting. It also supports action items.
- *Hive Notes (Hi)* - Hive Notes is a part of the larger Hive project management platform (https://hive.com). It supports note taking, an agenda, and action items.
- *Hypercontext (Hy)* - Hypercontext (https://hypercontext.com) connects to an agenda to create a meeting event. It supports an agenda and collects action items during the meeting. At the end, a meeting recap is sent to all participants.
- *Lean Coffee (LC)* - Lean Coffee (https://www.leancoffeetable.com) sets up an agenda, and supports progress through the agenda using the tool. It also generates meeting minutes.
- *Lucid Meeting (LM)* - Lucid Meeting (https://www.lucidmeetings.com) supports agendas and action items. The facilitator does the note taking but participants can add comments. Lucid supports scheduling recurring meetings.
- *Otter.ai (O)* - Otter.ai (https://otter.ai) transcribes meetings while they are occurring. Participants can highlight parts of the meeting (transcripts or agenda items) and save them as Meeting Gems. It also creates a meeting summary.
- *Parabol (P)* - Parabol (https://www.parabol.co) focuses on Agile, with templates available for common types of meetings. Contains a meeting timer to keep things on track and produces a meeting summary at the end of the meeting.

Table II indicates which features from those indicated in bold in Table I are implemented by which tool. If a tool only implements some aspects of a feature (such as only allowing one or two types of tags rather than all the types indicated) we are still counting the implementation. On the other hand, if a tool claims to implement a feature but does so in a way that does not meet the feature intent, such as integrating with Slack but only to send meeting invitations and not to connect posts with meetings, we did not mark it as implemented. We also looked for features that were easily available in the tool. For example, in some tools one can capture a whiteboard drawing by sketching on a whiteboard that was screen-shared, using a screen capture tool to create a picture, and then attaching the picture as a file. We did not count that as a feature since it involved using external tools that were not directly integrated. Whiteboard sketching is a special case—none of the tools had their own implementation but some implemented it through integration with other tools—Miro (m) or Zoom (z).

V. DISCUSSION

Our first research contribution was the initial set of desirable features shown in Table I. These features support either capture or retrieval of design information. Tagging and linking are especially key, since they support an "incremental formalization" approach to rationale capture, as described by Shipman and McCall [22]. Tagging can be used to highlight specific types of information, such as the decisions, alternatives, and arguments that comprise the design rationale. Linking supports capture of design history, as the design evolves over time.

For our second contribution, we compared the features to those implemented in commercial meeting support tools, as shown in Table II.

Two of the features are supported by all the tools: Collaborative Notes and Access Prior Meetings. This is expected, since these are features that support both the second criteria for inclusion (providing value within the meeting) and the third (providing features that make it easier to leverage meeting results). Surprisingly, only half the tools studied provided the ability to search across meetings. This means that those looking for information would need to know which meeting contained the relevant information.

The most common type of tagging supported was a follow-up tag, where a meeting attendee was tagged as part of

TABLE II
MEETING FEATURES SUPPORTED

Feature	B	D	Fe	Fi	Hi	Hy	LC	LM	O	P
Pre-Set Meeting Tags			●	●		●		●	●	
Associate Tags with Agenda			●							
Link Meetings		●				●			●	
Link External Tools	●	●	●	●	●	●				●
Collaborative Notes	●	●	●	●	●	●	●	●	●	●
White-board Sketching	m, z	z	z	z		z	z		z	
White-board capture and tagging										
Capture Snippet				●				●	●	
Edit Snippet									●	
Tagging		●	●	●	●		●	●	●	●
Follow-up Request		●	●	●	●		●	●	●	●
Access Prior Meeting	●	●	●	●	●	●	●	●	●	●
Search Across Meetings			●	●		●		●	●	
Search Within Meeting				●		●		●	●	
Quick Text Search			●	●		●		●	●	
Go to snippet Context				●					●	
Add Past Meeting Link		●				●			●	
Add Future Meeting Link		●							●	

an action item. This is why there is a one-to-one mapping between tools that supported tagging (of any form) and those that supported making follow-up requests. Offering an expanded set of tags would support easier access to design rationale. Fellow.app supports creating custom tags that can be used to annotate meetings, which in turn can then be used to tag things like decisions, alternatives, and arguments. fireflies.ai uses ”topic trackers”—keywords set up by the meeting administrator that can be automatically detected in the transcript.

When done in person, design often involves whiteboard sketching, such as when drawing user interface mock-ups or creating modeling diagrams. No tools supported this themselves, only indirectly by integration with other tools.

The ability to capture snippets of design discussion was only offered by a few tools that included transcription. Being able to capture key portions of the discussion, with features like the Otter.ai Meeting Gems that allow for participants to quickly capture key parts of the discussion, makes it easier to find useful information without having to read an entire transcript. None of the tools allowed direct whiteboard capture. This is a critical feature that is needed in order to accurately capture the design history.

Only a few tools supported features that involve linking meetings. Recurring meetings are very common across all kinds of industries. Recurring meetings are one type of meeting where information needs to be carried over. For example, Docket explicitly provided support to link prior meeting notes and action items so they would be discussed as part of the agenda for the next occurrence of the meeting. We also are interested in studying how information may go across different types of meetings, such as in the meetings held by the Healthcare Information Systems company described in our scenario earlier in this paper or the Meeting Streams described by Duffy and O'Rourke [42]. These meetings may not have a large overlap of people. None of the tools studied explicitly provided a way to link meetings that were not recurring.

VI. CONCLUSIONS AND FUTURE WORK

The shift towards hybrid and remote work has increased our reliance on software tools for meeting support. This has also made it possible to capture more information electronically, either by recording discussions or through richer representations of meeting notes and discussion boards. This means that we may now have the ability to capture and access information such as design rationale much more easily.

Here we presented an initial set of meeting tool features that can better support the capture and access of design information and an evaluation of current commercial meetings support tools as a first step in a larger research agenda, outlined here:

1) Validate these criteria with industry software designers. We expect to see more criteria emerge as we start to work with our target users.
2) Study additional meetings to determine where information needs lie.
3) Build and evaluate prototype tools for design information capture and retrieval in and from meetings. We are planning on extending KnoCap [40], a research prototype that supports a shared white-board and capture of meeting snippets.

The capture and use of design knowledge, and in particular rationale, has been a known challenge for many years. The primary obstacle has been concerns over cost and effort required. With new tools and new ways of collaboration, some of these obstacles may be beginning to fall away.

ACKNOWLEDGMENT

We would like to thank our anonymous industrial partners.

REFERENCES

[1] M. Finnegan, "For developers, too many meetings, too little 'focus' time," Aug. 2022. https://www.computerworld.com/article/3669911/for-developers-too-many-meetings-too-little-focus-time.html.

[2] A. M. Soria, A. van der Hoek, and J. Burge, "Recurring distributed software maintenance meetings: toward an initial understanding," in *Proceedings of the 15th International Conference on Cooperative and Human Aspects of Software Engineering*, (Pittsburgh Pennsylvania), pp. 21–25, ACM, May 2022.

[3] W. Kunz and H. W. Rittel, "Issues as Elements of Information Systems (Working Paper 131)," *Center for Planning and Development Research, Berkeley, USA*, 1970.

[4] J. E. Burge, J. M. Carroll, R. McCall, and I. Mistrik, *Rationale-Based Software Engineering*. Berlin: Springer Publishing Company, Incorporated, 2008.

[5] T. P. Moran and J. M. Carroll, eds., *Design rationale: concepts, techniques, and use*. Computers, cognition, and work, Mahwah, N.J: L. Erlbaum Associates, 1996.

[6] J. Lee, "Design rationale systems: understanding the issues," *IEEE Expert*, vol. 12, pp. 78–85, May 1997.

[7] E. J. Conklin and K. B. Yakemovic, "A Process-Oriented Approach to Design Rationale," *Human-Computer Interaction*, vol. 6, pp. 357–391, Sept. 1991.

[8] G. Fischer, A. Lemke, R. McCall, and A. Morch, "Making Argumentation Serve Design," *Human-Computer Interaction*, vol. 6, pp. 393–419, Sept. 1991.

[9] J. Lee, "Extending the Potts and Bruns model for recording design rationale," in *[1991 Proceedings] 13th International Conference on Software Engineering*, (Austin, TX, USA), pp. 114–125, IEEE Comput. Soc. Press, 1991.

[10] C. Potts and G. Bruns, "Recording the reasons for design decisions," in *Proceedings. [1989] 11th International Conference on Software Engineering*, (Singapore), pp. 418–427, IEEE Comput. Soc. Press, 1988.

[11] M. Klein, "DRCS: An Integrated System for Capture of Designs and Their Rationale," in *AI in Design*, pp. 393–412, 1992.

[12] A. MacLean, R. Young, V. Bellotti, and T. Moran, "Questions, Options, and Criteria: Elements of Design Space Analysis," *Human-Computer Interaction*, vol. 6, pp. 201–250, Sept. 1991.

[13] R. J. McCall, "PHI: a conceptual foundation for design hypermedia," *Design Studies*, vol. 12, pp. 30–41, Jan. 1991.

[14] J. E. Burge and D. C. Brown, "Software Engineering Using RATionale," *Journal of Systems and Software*, vol. 81, pp. 395–413, Mar. 2008.

[15] Y. Liang, Y. Liu, C. K. Kwong, and W. B. Lee, "Learning the "Whys": Discovering design rationale using text mining — An algorithm perspective," *Computer-Aided Design*, vol. 44, pp. 916–930, Oct. 2012.

[16] C. López, V. Codocedo, H. Astudillo, and L. M. Cysneiros, "Bridging the gap between software architecture rationale formalisms and actual architecture documents: An ontology-driven approach," *Science of Computer Programming*, vol. 77, pp. 66–80, Jan. 2012.

[17] B. Rogers, J. Gung, Y. Qiao, and J. E. Burge, "Exploring techniques for rationale extraction from existing documents," in *2012 34th International Conference on Software Engineering (ICSE)*, (Zurich), pp. 1313–1316, IEEE, June 2012.

[18] F. Mao, R. E. Mercer, and L. Xiao, "Extracting Imperatives from Wikipedia Article for Deletion Discussions," 2014.

[19] R. Alkadhi, M. Nonnenmacher, E. Guzman, and B. Bruegge, "How do developers discuss rationale?," in *2018 IEEE 25th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, (Campobasso), pp. 357–369, IEEE, Mar. 2018.

[20] Z. Kurtanović and W. Maalej, "On user rationale in software engineering," *Requirements Engineering*, vol. 23, pp. 357–379, Sept. 2018.

[21] M. Lester and J. E. Burge, "Identifying Design Rationale Using Ant Colony Optimization," in *Design Computing and Cognition '18* (J. S. Gero, ed.), pp. 537–554, Cham: Springer International Publishing, 2019.

[22] F. M. Shipman and R. J. McCall, "Incremental formalization with the hyper-object substrate," *ACM Transactions on Information Systems*, vol. 17, pp. 199–227, Apr. 1999.

[23] N. Mangano, T. D. LaToza, M. Petre, and A. van der Hoek, "How Software Designers Interact with Sketches at the Whiteboard," *IEEE Transactions on Software Engineering*, vol. 41, no. 2, pp. 135–156, 2015. Conference Name: IEEE Transactions on Software Engineering.

[24] D. Socha and J. Tenenberg, "Sketching Software in the Wild," in *35th International Conference on Software Engineering*, pp. 1237–1240, 2013. ISSN: 1558-1225.

[25] J. M. Atlee and M. W. Godfrey, "Studying Professional Software Designers and their Use of Abstraction," in *Studying Professional Software Design Workshop*, 2010.

[26] S. Baltes and S. Diehl, "Sketches and Diagrams in Practice," in *22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*, FSE 2014, pp. 530–541, 2014.

[27] U. Dekel and J. D. Herbsleb, "Notation and Representation in Collaborative Object-Oriented Design: an Observational Study," *ACM SIGPLAN Notices*, vol. 42, no. 10, pp. 261–280, 2007.

[28] L. J. Ball, B. Onarheim, and B. T. Christensen, "Design Requirements, Epistemic Uncertainty and Solution Development Strategies in Software Design," *Design Studies*, vol. 31, no. 6, pp. 567–589, 2010.

[29] B. Matthews, "Designing Assumptions," in *Software Designers in Action: A Human-Centric Look at Design Work* (M. Petre and A. van der Hoek, eds.), pp. 249–266, Chapman and Hall/CRC, 2013.

[30] M. Petre, "Insights from Expert Software Design Practice," in *7th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering*, ESEC/FSE '09, pp. 233–242, 2009.

[31] V. Popovich and B. Kraal, "Expertise in Software Design: Novice and Expert Models," in *Presented at Studying Professional Design Workshop*, 2010.

[32] H. Christiaans and R. A. Almendra, "Accessing decision-making in software design," *Design Studies*, vol. 31, pp. 641–662, Nov. 2010.

[33] A. Tang, A. Aleti, J. Burge, and H. van Vliet, "What Makes Software Design Effective?," *Design Studies*, vol. 31, no. 6, pp. 614–640, 2010.

[34] M. Lavallee and P. N. Robillard, "Why Good Developers Write Bad Code: An Observational Case Study of the Impacts of Organizational Factors on Software Quality," in *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, (Florence, Italy), pp. 677–687, IEEE, May 2015.

[35] G. Tur, A. Stolcke, L. Voss, S. Peters, D. Hakkani-Tur, J. Dowding, B. Favre, R. Fernandez, M. Frampton, M. Frandsen, C. Frederickson, M. Graciarena, D. Kintzing, K. Leveque, S. Mason, J. Niekrasz, M. Purver, K. Riedhammer, E. Shriberg, J. Tien, D. Vergyri, and F. Yang, "The CALO Meeting Assistant System," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 18, no. 6, pp. 1601–1611, 2010. Conference Name: IEEE Transactions on Audio, Speech, and Language Processing.

[36] S. Chandrasegaran, C. Bryan, H. Shidara, T.-Y. Chuang, and K.-L. Ma, "TalkTraces: Real-Time Capture and Visualization of Verbal Content in Meetings," in *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, pp. 1–14, New York, NY, USA: Association for Computing Machinery, May 2019.

[37] M. Jasim, P. Khaloo, S. Wadhwa, A. X. Zhang, A. Sarvghad, and N. Mahyar, "CommunityClick: Capturing and Reporting Community Feedback from Town Halls to Improve Inclusivity," *Proceedings of the ACM on Human-Computer Interaction*, vol. 4, pp. 1–32, Jan. 2021.

[38] Y. Shi, C. Bryan, S. Bhamidipati, Y. Zhao, Y. Zhang, and K.-L. Ma, "MeetingVis: Visual Narratives to Assist in Recalling Meeting Context and Content," *IEEE Transactions on Visualization and Computer Graphics*, vol. 24, pp. 1918–1929, June 2018.

[39] S. Hunter, P. Maes, S. Scott, and H. Kaufman, "MemTable: an integrated system for capture and recall of shared histories in group workspaces," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, (Vancouver BC Canada), pp. 3305–3314, ACM, May 2011.

[40] A. M. Soria, "KNOCAP: capturing and delivering important design bits in whiteboard design meetings," in *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: Companion Proceedings*, (Seoul South Korea), pp. 194–197, ACM, June 2020.

[41] D. Socha and S. Walter, "Is Designing Software Different From Designing Other Things?*," *Int. J. Engineering Education*, vol. 22, no. 3, pp. 540–550, 2006.

[42] M. Duffy and B. O'Rourke, "The Agency of Meetings Collectively in an Organizational Setting," in *The Gothenburg Meeting Science Symposium*, (Gothenburg, Sweden), pp. 1–20, 2017.

[43] C. S. Bedingfield and P. J. Clarkson, "Design Meetings: Towards an Understanding of the Stages and Activities that Influence Success," *Proceedings of the Design Society: DESIGN Conference*, vol. 1, pp. 501–510, May 2020.

[44] "Tools for remote teams." https://www.collaborationsuperpowers.com/tools/.