Design Decisions by Voice: The Next Step of Software Architecture Knowledge Management

Rafael Capilla, Rodi Jolak, Michel Chaudron, Carlos Carrillo

Design Decisions by Voice: The Next Step of Software Architecture Knowledge Management

Rafael Capilla[1][0000−0002−6943−1285], Rodi Jolak[2][0000−0001−5656−9253], Michel R. V. Chaudron[2][0000−0001−7517−6666], and Carlos Carrillo[3][0000−0002−5343−3323]

[1] Rey Juan Carlos University, Madrid, Spain
rafael.capilla@urjc.es
[2] Chalmers and Gothenburg University, Gothenburg, Sweden
rodi.jolak@cse.gu.se
chaudron@chalmers.se
[3] Technical University of Madrid, Madrid, Spain
carlos.carrillo@upm.es

Abstract. Architectural Design Decisions (ADDs) capture the essence of relevant Architectural Knowledge (AK) and the underpinning rationale in order to produce well-designed software architectures. AK and design rationale might get lost if not captured at the same time when the architecture is discussed and modeled in early design phases. For years, this relevant knowledge has been captured using text templates and supported by a number of research tools. Nevertheless, as no commercial tool is still available combining AK capturing with UML notations to facilitate capturing the design decisions at the same time the architecture is modeled, is the major barrier to convince software architects and companies to invest in documenting the significant design decisions. As capturing AK using text templates requires an extra effort, we propose an approach to make the documentation process easier and reduce the effort thereof by using voice commands. In particular, we suggest an approach to: (i) capture ADDs using voice commands during design conversations, and (ii) link the captured ADDs to UML notations. Our approach integrates OctoUML, a modeling tool with voice commands for capturing design decisions by voice.

Keywords: Software Architecture · Architectural Knowledge · Design Decisions · Voice Decisions · Knowledge Capturing · UML.

1 Introduction

It is well recognized by the software architecture community that documenting Architectural Design Decisions (ADDs) is extremely relevant to avoid knowledge vaporization [4]. For years, software architecture documentation has been focusing on documenting architecture models, design patterns, and the results of architecture evaluations. However, it is uncommon to find documented design decisions explicitly [10]. This is mainly because of the burden and cost of ADDs'

capturing effort, as well as the lack of suitable tool-support [9]. Furthermore, capturing relevant Architectural Knowledge (AK) in Open Source Software (OSS) projects becomes more complex due to loosely coordinated software contributors who tend to code solutions without producing adequate documentation [7].

In a survey that we detail in Section 3, we find that software architecture experts and practitioners perceive capturing ADDs alongside other software artifacts as valuable. However, the poor flexibility, usability, and effectiveness of the majority of ADDs documentation tools is the main barrier to the adoption of these tools [6]. In order to facilitate the effectiveness of the AK-capturing process, we propose a novel approach that uses voice commands integrated with a modeling tool for capturing and management of Architecture Design Decisions by Voice, which is design decisions that are communicated by voice (ADDsV) during architecture design conversation. In this work in progress paper, we investigate the usefulness of "capturing ADDsV" through a survey with software architecture experts and practitioners, and we describe an approach to support: (i) Capturing ADDsV during early-phase design and modeling meetings, and (ii) Managing the captured ADDsV by linking these decisions to UML artifacts. The remainder of this article is as follows. Section 2 describes some related works and in Section 3 we outline our approach using a modified version of the OctoUML tool (https://github.com/Imarcus/OctoUML). In Section 4 we describe our first experiments and in Section 5 we provide the feedback collected from software architecture researchers and practitioners. In Section 6 we discuss some limitations of our approach and in Section 7 we discuss our conclusions and future research steps.

2 Related Work

Since 2004 architectural knowledge has become the next step [4] for documenting the design decisions and supported by the ISO/IEC 42010 standard [1]. There exist approaches suggesting a number of AK research tools for capturing and sharing the design decisions [6]. However, the diversity of these tools did not solve, in a satisfactory way, the duality of the AK capturing problem while software architects model an architecture [5], except two tools (i.e., AREL and ADMentor) that enable capturing design rationale with some modeling capabilities. In a survey described in [17], the authors study human aspects in software architecture decision-making such as collaborative group making, the role of agile practices in decision-making, and the facilities provided by several decision-making tools for capturing design rationale. However, the role of multimodal interfaces facilitating the tasks of capturing design decisions is not investigated by the authors.

Although some AK research tools offer some collaborative support for sharing the design decisions (e.g. WiKi tools like EAGLE [8]), only some recent tools (e.g. SAW [14], ASQ [18]) provide explicit support to achieve a consensus when stakeholders vote about decisions alternatives. As effective software design requires decision making and reasoning to solve the problem-solution co-

evolution [16], we need to increase the ease of use of tools for capturing design decisions and modeling their architectures concurrently.

The use of multimodal interfaces for modeling architectures is now possible such as discussed in [13], where a prototype speech control system is used to enhance the interaction with UML tools. The approach described in [3] presents SketchLink, a tool to sketch diagrams in software engineering so designers can capture and annotate the diagrams and link these to code artifacts. Modern solutions like OctoUML [11] allow designers to interact combining touch screens and voice commands to depict architectural elements in a collaborative way [12]. Unfortunately, capturing design decisions and their rationale is not supported by OctoUML. Another work [20] envisions using video walls for collaborative decision making and capturing the decisions on the fly. However, the approach is not capable to extract the decisions from the voice files. Finally, a recent experience [15] suggests a similar approach to ours for capturing voice conversations during design meetings by using the KnoCap tool, with which designers can mark the most relevant fragments to be used at any time.

3 Approach

OctoUML [11] is a proof-of-concept open-source prototype that explores supporting software development teams by offering more human-centered interaction modalities in software development tools. In order to address the challenge for capturing design decision by voice and link these to UML artifacts we address the following research questions:

- RQ1 How useful is capturing ADDsV during collaborative design meetings?
- RQ2 How can design modeling tools be adapted to capture and manage ADDsV?

To this end, we use OctoUML, an interactive whiteboard software that supports touch and voice interaction, which can be used on standard PCs and tablets. Figure 1 shows the main interface of the tool.

The key features of OctoUML are summarized below:

- Supporting the creation and mixing of software models at different levels of formality – in particular sketches and formal 'geometric' models. This enables a smooth transition from design ideation to more formal design representations.
- Collaborative distributed development by offering a joint canvas where developers on different locations can draw and edit shared diagrams.
- Combining navigation in design and source code in a single view.

3.1 Voice interaction

Current technology provides opportunities for more intuitive and efficient interaction with the software. One notorious challenge in software development

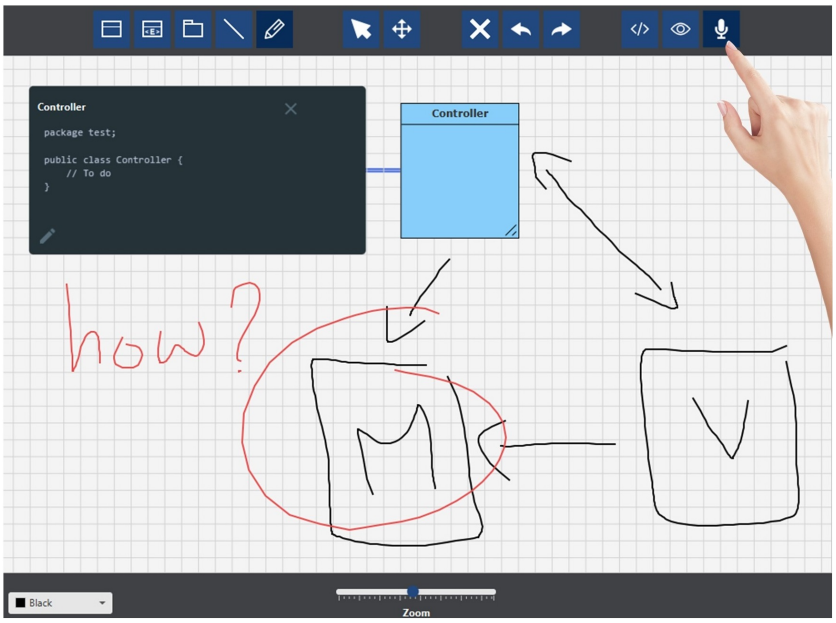

Fig. 1. Modeling an architecture with OctoUML using the touch interface.

has been the production and maintenance of documentation, partly due to the old-fashion use of typing as the way to edit documents. In the last decade, mobile phones have pushed additional interaction modes, such as touch, voice, and gesture. We propose that these modalities can also be used to ease the work for software developers to enter and maintain documentation [11]. In particular, enabling voice-recording near the interactive whiteboard where design discussions take place, opens up new ways for registering design discussions, including design decisions and rationale. The OctoUML design environment supports interaction via voice commands by using the Sphinx4 voice recognition library [19].

3.2 OctoUML Grammar Extension

OctoUML provides a basic grammar (Java Speech Grammar Format) to define the voice commands the tool supports. In order to capture design decisions, we extended the grammar in the following way (See Figure 2):

- We provide the ability to create a design decision that can be captured using voice.
- We support capturing the design rationale of the decision.
- We support replaying, changing (updating), and removing a decision by voice.
- We support sharing a decision with other users, which is useful in group decision making.

```
public <greet> = (name) (user | student | system | teacher| course |
           car | archivist | supervisor | detective | case
               | material | person | fact | photo | photograph
               | other | investigator | suspect | victim | task
               | subject | testimony | witness | personnel
               | agent | photographer | investigation | people | event | content |
   criminal | persons connected | persons associated | associated | collaborator |
   people connected | people associated | connected people | associated people |
   clue | class | package);
public <command> = (share) (decision);
public <command> = (create) (class | edge | package | design decision | rationale);
public <selection> = (choose) (draw | select | move | select decision | replay
decision | update decision | delete decision | show decision);
public <stop> = (stop naming);
```

Fig. 2. OctoUML grammar extension for capturing design decisions.

According to the changes in the grammar, we provide the command to `create design decision`, but we can also add the rationale for each decision using the command `create rationale`. Other useful commands to manipulate design decisions or rationale by voice are `replay decision`, `update decision`, `delete decision`, and `show decision`. We also created a separate command, `share decision`, for sharing the decisions with other users. This command is useful in distributed teams for group decision making.

3.3 Implementation with Sphinx

During the implementation, we integrated the OctoUML code with the CMUSphinx library (https://cmusphinx.github.io/) used for controlling the voice. The graphical user interface is defined by the FXML (i.e. XML-based user interface markup language) created by Oracle Corporation, so in OctoUML we have two main files (i.e. classDiagramView.fxml, sequenceDiagramView.fxml), that we modified to include an icon supporting the voice management.

In order to include an icon to start the voice recording of the decisions, we used the abstract class *AbstractDiagramController.java*, so we recognize the voice commands based on the grammar. The problem, in this case, was that the microphone is an exclusive resource that can't be shared by other threads and once it recognizes a command, we must stop the recognition facility and start capturing the voice until the user presses the new icon to stop capturing the decisions. By the moment we cannot use a voice command to stop recording the decisions because we cannot know when a user finished describing a decision with the voice.

We also modified the class *voiceController* to add the new code supporting the changes in the grammar and avoid locking the microphone. Therefore, we implemented a new class named *RecordDDController* to manage the recording mode and capture voice decisions as WAV files. Once the microphone is unlocked and the record is finished, the class *RecordDDController* stores the voice file and replays the voice decision in the speakers. The recording mode finishes once the

user clicks on the recording mode icon in the OctoUML smart touch screen releasing the resources.

Finally, we created another class, *RecordDDManagement* to assign a name to the voice file which stores the description of the design decision. The CMUSphinx4 package used for voice recognition and capturing also decodes the voice and translates this into a text string in case we need to display the decision on the screen. As a summary, the list of the classes we modified are the following:

- SpeechSourceProvider.java: allocates a new microphone each time it is required by the VoiceController class and to avoid locking the microphone which is managed by the RecordDDController class
- AbstractSpeechRecognizer.java: it allows access to the microphone enabling different operations
- LiveSpeechRecognizer.java: implements the method to unlock the microphone
- Microphone.java: is used to access the voice files

4 Capturing Voice Decisions

As a first experience based on the extension implemented in Section 3.3, we do a trial for capturing ADDsV. A decision and its rationale can be captured before or after the architecture is depicted using OctoUML's drawing tools. Figure 3 shows ADDsV capturing process. A user can interact with the tool by (1) activating the microphone (white button) to enter in "command mode". Then, the user-interface automatically switches into "voice recording" mode and starts (2) capturing the ADDsV using the "create design decision" command or capturing a rationale using "create rationale" command. Once the user finishes discussing the design decision or rationale, he/she interacts (3) with the screen by deactivating the microphone (red button). After that, the voice design decision or rationale is reproduced (4) so the user can know the last decision taken. The use of meaningful names for ADDsV is useful for searching, retrieving, or listing such decisions. Currently, the functionality used for naming decisions is implemented using the keyboard. This is because naming decisions can use symbolic names or a few short meaningful words. It turns out that using voice commands is not the easiest way for this step. One solution to this could be to train people to assign appropriate names. Once the decision is recorded, the tool automatically waits until the user gives a meaningful name to the decision (step 5). Once the user spells out the word "end", the tool stores (6) the ADDsV as *.wav* and *.txt* files.

5 Perception of Experts and Practitioners

We conducted a short survey with software architecture experts and practitioners. We asked them to answer 10 questions related to the usefulness of using

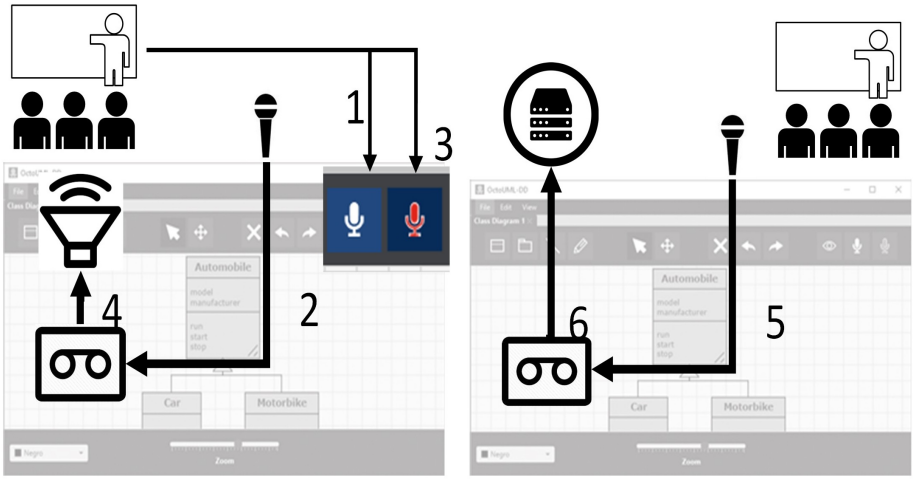

Fig. 3. Capturing design decisions by voice (left side), and giving a name to the recorded decision (right side).

voice for capturing architecture design decisions. We remark that this survey does not investigate the use and usability aspects of our prototype, but rather highlights: (i) to what extent it is important for practitioners capturing design decisions by voice that can be linked to software artifacts, and (ii) how this approach could be useful in agile software development approaches and collaborative decision-making activities. The results are shown in Figure 4. We collected the data during October and November 2019, and we received 17 responses from 10 different countries in Europe, South America, and Canada. The age of the respondents varies between [27-67] years. The experience of the respondents in software architecture design ranges between [4-35] years, but most of them have between [6-9] years of experience in architecture design. The average age of the participants is 39.5 years and the average year of experience is 14 years.

From the responses of the subjects, we derived the following major findings.

1. Most of the respondents think that capturing architectural knowledge (AK) is valuable for software architecture design practice. This indicates the importance of our endeavor in capturing design decisions and AK via voice.
2. The respondents perceive that capturing decisions during modeling tasks is highly relevant for collaborative decision-making, especially for distributed teams. Sharing ADDsV is also perceived as valuable for developers. These results indicate that capturing and sharing design decisions of developers working in teams is still a challenging task that is not effectively supported by tools or practices.
3. Some respondents believe that capturing and linking ADDsV to architectural artifacts is important, but not as much as linking these decisions to a UML diagram. While we think that linking design decisions to the architecture and

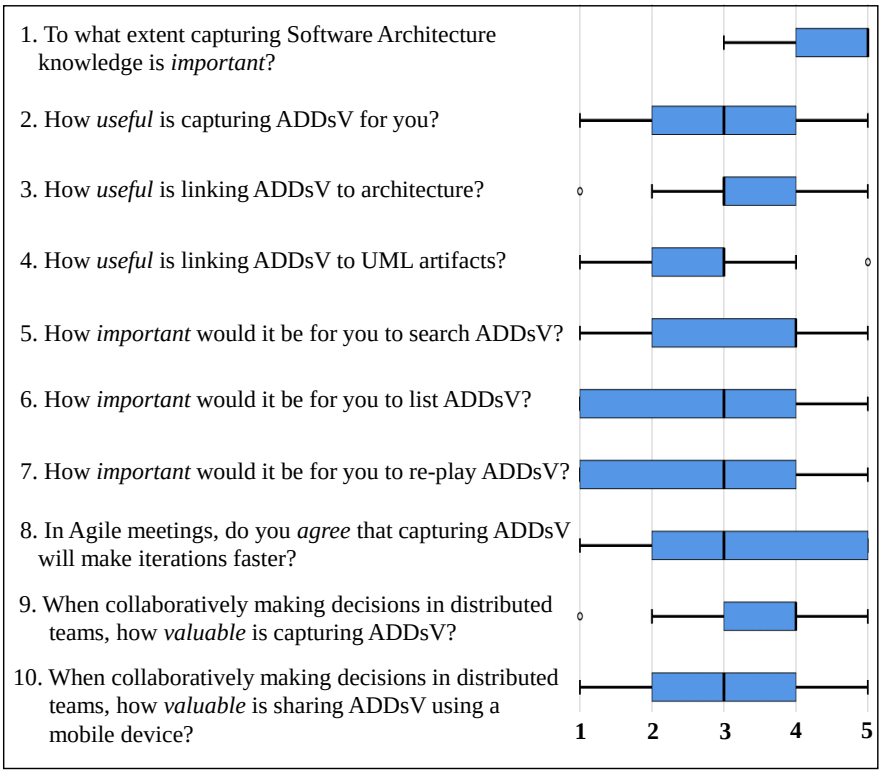

Fig. 4. Perception of practitioners capturing Architecture Design Decisions by Voice (ADDsV).

UML diagrams is important for traceability, it seems that some respondents prefer to use other diagrams than the UML to represent the architecture and, thus, indicated that the linking of ADDsV to UML diagrams is of less importance.

4. The respondents indicate that searching for, listing, and re-playing ADDsV are relevant and important matters, however, the diversity of the responses is wide. We think that this diversity in the responses is normal, as different users usually have different preferences of the features for the manipulation and interaction with the artifacts.
5. Many respondents believe it to be important to have ADDsV in agile meetings, as these decisions can be shared in a faster way than via documenting them. Again, this indicates the importance of our endeavor in capturing design decisions and AK via voice.

In addition to these findings, we performed a qualitative analysis of the personal opinions of the respondents. This analysis yielded some interesting issues:

One subject indicated:

> *"I have never seen this in practice, so it is difficult to imagine how voice decisions would work. Perhaps you can consider transcribing the decisions automatically so that you also have them as text".*

While another respondent expressed:

> *"Writing down decisions is favorable over voice decisions, because it forces the author into a certain thought process and proper formulation of her thoughts. Voice decisions may be easy to record, but that does not save much time in my opinion".*

Our perception is that architects still think of documenting decisions as the main way to capture architecture knowledge, and they hardly perceive that ADDsV can be useful too. Another respondent said:

> *"The vast majority of decisions are not worthwhile to share; as the necessary context is not made explicit that allows the understanding of the decision".*

We believe that decisions made for a specific software project are bounded by the context of the project, so all stakeholders should be aware of the context boundaries where decisions are meaningful for all of them. However, capturing the rationale by voice can help to make decisions more understandable.

Finally, one respondent mentioned:

> *"distributed teams collaborating over Jira or chat platforms (e.g. Slack) mostly document decisions in tickets. That approach works very well for them. I see the benefit of the tool, not to capture the decision itself, but rather for capturing alternatives and pro/con arguments of the decision".*

We are in favor of capturing details as much as possible about the rationale of design decisions, including the pros and the cons of these decisions. ADDsV files can be augmented with more details on the pros and cons of the decisions. But overloading these files with extra details reduces the agility of the approach, as additional details on the pros and cons can be found in the detailed textual documentations.

From the qualitative analysis of the comments provided by the subjects interviewed we come up with the following issues. As capturing and documenting decisions by voice is a new practice for software architects our prototype plans to produce text strings of the voice decisions that can be documented in the traditional way, as searching for decisions in voice files could be complex. Also, the opinion expressed by one subject that voice decisions won't save much time could be partially true but in remote teams where users can use also their mobile phones to remotely capture a decision can ease the collaborative part in distributed teams, instead of having a UML tool to depict architecture artifacts. Therefore, capturing decisions more agile is not only a matter of time-saving. We agree that sharing a decision requires a context, but this can be added as an extension in the grammar. However, we believe all relevant stakeholders should know the context or specific project where decisions are made.

6 Limitations

Although this is an early experience for capturing design decisions by voice while modeling UML diagrams, our approach still has some limitations for practical use of the proposed solution. First, the proposed grammar can recognize commands for handling some decisions using voice, but the captured voice design decision is recorded as a voice stream, so currently, we cannot recognize pronounced sentences. One way to mitigate this factor is to integrate our solution with existing voice recognition software to provide extended capabilities. Second, extracting specific elements from the recorded voice is not possible at present, so we need to provide advanced capabilities to recognize concrete parts (e.g., the selection of a specific design pattern). Third, our current solution does not translate design designs by voice to textual forms and does not provide mechanisms to manipulate these forms. However, this feature is planned for implementation in the future.

7 Conclusion and Research Challenges

Compared to previous approaches using text templates, we offer an approach that combines a multi-modal interface for capturing the relevant Architecture knowledge and depicting software architectures using the touch screen with voice for capturing the relevant AK. This approach provides an agile way that is suitable for agile development teams to capture design decisions in a non-intrusive way while designing or discussing an architecture.
We used a survey to assess the usefulness of capturing design decisions by voice during a collaborative architecture design meeting. The respondents perceived capturing design decisions by voice as useful and indicated the potential of our approach in supporting architecture knowledge management. From our initial experience, software architects need some training to capture design decisions by voice together with their rationale using concise sentences. In addition, we trust on the reliability of the survey as we asked experts in architectural knowledge and software architecture as well, even if we only got 17 responses.
Currently, the grammar for assigning names is a subset of the full English grammar, as most of the keywords are based on computer science terms. In this direction, there are some interesting experiences choosing a suitable NLP (natural language processing) dictionary for analyzing documents [2]. In addition, our approach goes beyond [15] as we use one single tool for architecture modeling as well as for capturing the design decisions by voice and not only capturing the voice in design meetings. Hence, software architects can model using a UML tool and capture the decisions at the same time they depict their architecture models.

Below, we provide a list of the research challenges that will guide the next steps of future work:

- Extend the grammar to support additional voice-based interaction functionality.

- Provide support for group-decision making.
- Support versioning of decisions to track history.
- Display voice decisions in the smart touch screen so users can easily find and replay the decisions linked to design artifacts.
- Explore the use of mobile devices for capturing voice in an agile way.
- Add more semantics to the decisions captured.

References

1. ISO/IEC 42010:2011, Systems & Software Engineering — Architecture Description. `https://www.iso.org/standard/50508.html`, accessed: 2019.
2. Al Omran, F.N., Treude, C.: Choosing an NLP library for analyzing software documentation: a systematic literature review and a series of experiments. In: 14th Conference on Mining Software Repositories. pp. 187–197. IEEE Press (2017)
3. Baltes, S., Schmitz, P., Diehl, S.: Linking sketches and diagrams to source code artifacts. In: 22nd ACM SIGSOFT Symposium on Foundations of Software Engineering. pp. 743–746. ACM (2014)
4. Bosch, J.: Software architecture: The next step. In: European Workshop on Software Architecture. pp. 194–199. Springer (2004)
5. Capilla, R.: Embedded design rationale in software architecture. In: 2009 Working IEEE/IFIP Conference on Software Architecture & European Conference on Software Architecture. pp. 305–308. IEEE (2009)
6. Capilla, R., Jansen, A., Tang, A., Avgeriou, P., Babar, M.A.: 10 years of software architecture knowledge management: Practice and future. Journal of Systems and Software **116**, 191–205 (2016)
7. Ding, W., Liang, P., Tang, A., Van Vliet, H., Shahin, M.: How do open source communities document software architecture: An exploratory survey. In: 2014 19th International conference on engineering of complex computer systems. pp. 136–145. IEEE (2014)
8. Farenhorst, R., Lago, P., Van Vliet, H.: Eagle: Effective tool support for sharing architectural knowledge. International Journal of Cooperative Information Systems **16**(3/4), 413–437 (2007)
9. Harrison, N.B., Avgeriou, P., Zdun, U.: Using patterns to capture architectural decisions. IEEE software **24**(4), 38–45 (2007)
10. Jansen, A., Avgeriou, P., van der Ven, J.S.: Enriching software architecture documentation. Journal of Systems and Software **82**(8), 1232–1248 (2009)
11. Jolak, R., Vesin, B., Chaudron, M.R.V.: Using voice commands for UML modelling support on interactive whiteboards: Insights and experiences. In: CIbSE. pp. 85–98 (2017)
12. Jolak, R., Wortmann, A., Chaudron, M.R.V., Rumpe, B.: Does distance still matter? revisiting collaborative distributed software design. IEEE Software **35**(6), 40–47 (2018)
13. Lahtinen, S., Peltonen, J.: Adding speech recognition support to UML tools. Jnl of Visual Languages & Computing **16**(1-2), 85–118 (2005)
14. Nowak, M., Pautasso, C.: Team situational awareness and architectural decision making with the software architecture warehouse. In: European Conf. on Software Architecture. pp. 146–161. Springer (2013)
15. Soria, A.M., van der Hoek, A.: Collecting design knowledge through voice notes. In: Proceedings of the 12th Int. Workshop on Cooperative & Human Aspects of Software Engineering. pp. 33–36. IEEE (2019)

16. Tang, A., Aleti, A., Burge, J., van Vliet, H.: What makes software design effective? Design Studies **31**(6), 614–640 (2010)
17. Tang, A., Razavian, M., Paech, B., Hesse, T.: Human aspects in software architecture decision making: A literature review. In: 2017 IEEE International Conference on Software Architecture, ICSA 2017, Gothenburg, Sweden, April 3-7, 2017. pp. 107–116 (2017)
18. Triglianos, V., Pautasso, C., Bozzon, A., Hauff, C.: Inferring student attention with asq. In: European Conference on Technology Enhanced Learning. pp. 306–320. Springer (2016)
19. Walker, W., Lamere, P., Kwok, P., Raj, B., Singh, R., Gouvea, E., Wolf, P., Woelfel, J.: Sphinx-4: A flexible open source framework for speech recognition (2004)
20. van der Werf, J.M.E., de Feijter, R., Bex, F., Brinkkemper, S.: Facilitating collaborative decision making with the software architecture video wall. In: 2017 IEEE International Conference on Software Architecture Workshops (ICSAW). pp. 137–140. IEEE (2017)