Gottfried Wilhelm
Leibniz Universität Hannover
Fakultät für Elektrotechnik und Informatik
Institut für Praktische Informatik
Fachgebiet Software Engineering

Entwicklung einer Webanwendung für Interaktionsanalyse in Meetings

Bachelorarbeit

im Studiengang Informatik

von

Kristian Enns

Prüfer: Prof. Kurt Schneider
Zweitprüfer: Dr. Jil Klünder
Betreuer: Nils Prenner

Hannover, 19.08.2021

Erklärung der Selbstständigkeit

Hiermit versichere ich, dass ich die vorliegende Bachelorarbeit selbständig und ohne fremde Hilfe verfasst und keine anderen als die in der Arbeit angegebenen Quellen und Hilfsmittel verwendet habe. Die Arbeit hat in gleicher oder ähnlicher Form noch keinem anderen Prüfungsamt vorgelegen.

Hannover, den 19.08.2021

Kristian Enns

Zusammenfassung

Act4teams-SHORT ist ein Kodierungsschema für die Interaktionsanalyse von Meetings in der Softwareentwicklung. Mit Hilfe von act4teams-SHORT können Aussagen über den Erfolg von Meetings getroffen werden. Das Kodierungsschema wird in Echtzeit verwendet, das heißt die Kodierung erfolgt während eines Meetings und die Auswertung kann direkt im Anschluss daran durchgeführt werden. In dieser Arbeit wird eine Webanwendung zur digitalen Unterstützung dieser Echtzeitkodierung und Auswertung mit act4teams-SHORT entwickelt. Die Nutzbarkeit mit Tabletcomputern hat bei der Entwicklung dieser Webanwendung eine hohe Priorität, da sie vor allem mit Microsoft Surface Geräten verwendet werden soll. Für die Entwicklung des User Interfaces werden, anhand von Mockups, verschiedene Anordnungen von Komponenten verglichen und bewertet. Außerdem werden verschiedene Möglichkeiten der Datenvisualisierung, für die Auswertung der Daten, erarbeitet. Abschließend wird eines der Interfaces mit Hilfe des Webframeworks React.js implementiert.

Abstract

Act4teams-SHORT is a coding scheme for interaction analysis in software developement meetings. It can be used to draw conclusions about the success of meetings. The coding scheme is used live, which means the coding is done during meetings and can be evaluated directly after. The goal of this thesis is to develope a web application that digitally supports this live coding and subsequent evalutation of meetings with act4teams-SHORT. An important factor during the developement of this web application is the usability with tablet computers because the target devices are microsoft surface computers. Mockups of the user interface are used to compare and evaluate the usability of different layouts. Furthermore different types of data visualizations for the evaluation of the coded data are developed. Finally one of the interfaces is implemented with the help of the web framework React.js.

Inhaltsverzeichnis

Kapitel 1

Einleitung

Interaktionsanalyse ist eine interdisziplinäre Methode für die empirische Untersuchung der Interaktionen von Menschen miteinander und mit Objekten in ihrer Umgebung. Sie untersucht menschliches Handeln, wie verbale und nonverbale Interaktionen, um Routinen und Probleme zu finden und mögliche Lösungen für diese zu erarbeiten ([17], wörtlich übersetzt). Für die Interaktionsanalyse werden oft Videoaufnahmen mit Hilfe von Kodierungsschemas kodiert. Diese Kodierungsschemas teilen das Verhalten der beobachteten Personen in bestimmte Kategorien (auch Codes genannt) ein. Beispiele für solche Interaktionsanalyse Kodierungsschemas in verschiedenen Bereichen sind "Brief Romantic Relationship Interaction Coding Scheme(BRRICS)" von Humbad et al. [16], welches zur Kodierung von Interaktionen zwischen Ehepaaren verwendet wird, das "Four Habits Coding Scheme (4HCS)" von Krupat et al. [24], welches das Verhalten von Krankenhausärzten kodiert oder das "Advanced Interaction Analysis for Teams(act4teams) Coding Scheme" von Kauffeld et al. [20], welches zur Kodierung von Meetings verwendet wird.

Act4teams-SHORT von Klünder et al. [23] ist ein Kodierungsschema, welches zur Kodierung von Meetings von Software-Projekten verwendet wird und dabei helfen soll, Aussagen über den Erfolg von Meetings zu treffen. Anders als die zuvor genannten Kodierungsschemas wird act4teams-SHORT in Echtzeit und nicht anhand von Videoaufnahmen kodiert. Um diese Echtzeitkodierung zu unterstützen, soll in dieser Arbeit eine Webanwendung entwickelt werden.

1.1 Problemstellung

Die Kodierung von Meetings benötigt eine hohe kognitive Leistung. Deshalb, und wahrscheinlich um die Beobachteten nicht zu beeinflussen, werden meistens Videoaufnahmen kodiert. Hier soll jedoch die Echtzeitkodierung von Meetings unterstützt werden. Daher muss die Webanwendung möglichst

einfach zu bedienen sein, um die kodierende Person nicht zu behindern. Für die Echtzeitkodierung mit act4teams-SHORT wurde bisher ein Java-Prototyp verwendet. Die Kodierungsansicht des Java-Prototypen bestand aus einem 3x3-Raster für die Kategorien von act4teams-SHORT und einer Stoppuhr. Die kodierten Daten bestanden aus Namen der Kategorien mit Zeitstempeln. Jedoch konnten die Teilnehmer, zu denen diese Aktionen gehörten, nicht erfasst werden. Diese Funktion soll in der hier zu entwickelnden Webanwendung zusätzlich implementiert werden.

1.2 Lösungsansatz

Es soll eine Webanwendung für die act4teams-SHORT Kodierung entwickelt werden, welche die Funktionen des Java-Prototypen erweitert. Bei der Entwicklung soll vor allem auf Usability, also die einfache Nutzbarkeit der Applikation geachtet werden. Dafür sollen anhand von Mockups, verschiedene Anordnungen der Kodierungsansicht gegeneinander abgewogen werden. Dadurch, dass es eine Webanwendung ist, wird sie auch noch plattformunabhängig (portabel), solange ein moderner Internetbrowser vorhanden ist.

1.3 Struktur der Arbeit

Diese Arbeit ist wie folgt strukturiert. In Kapitel 2 werden der Hintergrund und die Grundlagen des Kodierungsschemas act4teams-SHORT vorgestellt. Außerdem werden Grundlagen der modernen Webentwicklung erklärt. In Kapitel 3 werden Mockups von verschiedenen Layouts der Kodierungsansicht miteinander verglichen und bewertet. Es werden zusätzlich die Ergebnisse des Anforderungsgesprächs mit dem Kunden dieser Webanwendung dokumentiert. In Kapitel 4 wird das in Kapitel 3 erarbeitete Konzept der Webanwendung implementiert und dargestellt.

Kapitel 2

Grundlagen

In diesem Kapitel werden die Grundlagen erklärt, die nötig sind um die Funktionsweise von act4teams-SHORT zu verstehen. Außerdem wird die Nielsen Heuristik erklärt, welche im Usability Engineering zum Finden von Problemen in Nutzeroberflächen genutzt wird. Zuletzt werden noch Werkzeuge der modernen Webentwicklung, die in dieser Arbeit verwendet wurden, vorgestellt. Dazu gehören Webframeworks und spezifische Packages.

2.1 Grundlagen zu act4teams

Der folgende Abschnitt über act4teams basiert auf einer Erklärung von Kauffeld et al. [20].

Act4teams steht für advanced interaction analysis for teams. Es ist ein Kodierungsschema zur Interaktionsanalyse von Gruppeninteraktionen, wie zum Beispiel Team-Meetings in Unternehmen, Gruppentreffen zum Lösen von Problemen oder auch Treffen zur Förderung der Gruppenkreativität. Mit Kodierung ist die Zuweisung von Verhalten und Aussagen von Meetingteilnehmern zu bestimmten Verhaltenskategorien (auch Codes genannt) gemeint. Dies kann sowohl digital durch Software unterstützt, als auch per Hand gemacht werden. Im Kontext dieser Arbeit wird auch öfters von Aktionen der Meetingteilnehmer, die zu einer Kategorie gehören, gesprochen. Act4teams definiert für die Kodierung insgesamt 43 sich gegenseitig ausschließende Verhaltenskategorien. Beispielsweise würde das Nennen eines Problems zu der Verhaltenskategorie *Problem* gehören, das Erklären eines Problems dann aber zu der Kategorie *Problemerklärung*. Die meisten dieser Kategorien gehören zu einer der folgenden vier großen Oberkategorien, die auch Kompetenzfacetten genannt werden (offizielle Übersetzung [21]): Professionelle Kompetenz, Methodenkompetenz, Sozialkompetenz und Selbstkompetenz. Die beiden im Beispiel genannten Kategorien *Problem* und *Problemerklärung* gehören zu der Facette *Professionelle Kompetenz*. In den übrigen drei Facetten wird auch noch zwischen positiven und negativen

Kategorien unterschieden. Ein Beispiel für die Facette *Selbstkompetenz* wäre, dass das Übernehmen von Verantwortung positiv und das Gegenteil, also das Verleugnen von Verantwortung, negativ eingeordnet wird. Manchmal kann Verhalten nicht genau einer Facette zugeordnet werden und dafür gibt es auch Kategorien die zu keiner der vier Facetten gehören, zum Beispiel wenn pausiert oder gelacht wird. Die Kategorien heißen dann auch genauso, also *Pause* und *Lachen*.

Mit Hilfe von act4teams können Verhaltensmuster in Meetings erkannt und Aussagen über den Erfolg von Meetings abgeleitet werden, da in einer großen Studie von Kauffeld und Lehmann-Willenbrock [19] klare Beziehungen zwischen den act4teams Kategorien, der Zufriedenheit mit den Team-Meetings und der Team-Produktivität gefunden werden konnten. Diese Studie hatte 3 Kernergebnisse: Erstens, dass negatives Verhalten (z.B. das Kritisieren von anderen Teilnehmern und Beschweren) einen stärkeren Einfluss auf das Ergebnis des Meetings hat, als positives Verhalten (z.B. Loben oder das Äußern einer positiven Einstellung). Zweitens, dass positive Ergebnisse für die Teams und Unternehmen nicht davon abhängen, wie oft Probleme genannt werden, sondern wie oft Lösungen genannt und Probleme und Lösungen analysiert werden. Und Drittens, dass vor allem die konkrete Planung von Maßnahmenschritten, als Ergebnis der Diskussionen über Probleme und Lösungen während des Meetings, einen positiven Einfluss auf die Zufriedenheit der Teilnehmer mit dem Meeting und die Teamproduktivität hat.

Aufgrund der feingranularen Kodierung in act4teams, eignet sich das Verfahren nicht zur Echtzeitkodierung. Mit Echtzeitkodierung ist die Kodierung während eines Meetings gemeint. Stattdessen werden Videoaufnahmen von Meetings kodiert, nachdem das Meeting schon stattgefunden hat. Es wurden jedoch vereinfachte Versionen von act4teams entwickelt, welche die Echtzeitkodierung, während eines Meetings, ermöglichen.

2.2 Echtzeitkodierung mit act4teams-SHORT

Folgende Beschreibung von act4teams-SHORT beruht auf den Ausführungen von Klünder et al. [23].

Aufbauend auf act4teams wurde das Kodierungsschema act4teams-SHORT entwickelt, mit dem Ziel die Echtzeitkodierung von Team-Meetings in Softwareprojekten zu ermöglichen und daraus direkt den Erfolg eines Meetings bestimmen zu können. Der Erfolg der Meetings ist wichtig, da effektive Meetings auch verstärkt zu erfolgreichen Projekten [19] und zufriedenen und produktiven Entwicklern führen, wie Prenner et al.[33] zeigen. Da weniger erfolgreiche Meetings direkt erkannt werden können, dank der Echtzeitkodierung, können sofort Maßnahmen ergriffen werden, um den Erfolg der Projekte nicht zu gefährden.

Aus der act4teams Studie von Kauffeld und Lehmann-Willenbrock [19] ist bekannt, dass die Beziehung zwischen Problemen und Lösungen, sowie konkrete Maßnahmeschritte als Ergebnis der Meetings, ausschlaggebende Faktoren für den Erfolg der Meetings sind, weshalb bei act4teams-SHORT analysiert wird, ob Probleme in den Meetings nicht nur genannt werden, sondern ob auch wirklich versucht wird diese Probleme zu lösen. Dafür wird selektiv kodiert, das bedeutet, dass nur die wichtigsten Aktionen der Teilnehmer kodiert werden. Daher mussten die einflussreichsten Kategorien (für den Erfolg von Meetings) in act4teams identifiziert werden, um die Echtzeitkodierung in act4teams-SHORt zu ermöglichen, indem 34 Team-Meetings von Studenten mit act4teams kodiert wurden. Nach zwei Fallstudien [33][23] in der Industrie, in der verschiedene Versionen von act4teams-SHORT erprobt wurden, ergaben sich neun essentielle Kategorien, die Klünder [22] so erklärt (Erklärungen wörtlich übernommen):

- **Probleme:** Benennen und Erläutern von Problemen, Stellungnahmen zu Problemen, Ursachen und Folgen von Problemen, Vergleiche zwischen Problemen
- **Lösungen:** Nennen von Lösungsvorschlägen, Sammeln von Vorschlägen, Ausarbeitung eines Lösungsvorschlags, Folgen von Lösungen, Anforderungen an Lösungen, Vergleiche zwischen Lösungen, Vorteile einer Lösung
- **Verknüpfungen und Vernetzungen:** Verknüpfungen zwischen Problemen und Vorschlägen, Einwände und Bedenken gegen Lösungen
- **Destruktives Verhalten:** Tadeln/Lästern, Schuldigensuche, Jammern, Kein Interesse an Veränderungen, Seitengespräche
- **Proaktives Verhalten:** Engagement für Veränderungen, Maßnahmen planen, Verantwortung übernehmen, Interesse an Veränderungen signalisieren
- **Kollegiales Verhalten:** Humor, Andere einbinden, Wertschätzung, Auf Beiträge anderer eingehen
- **Methodisch-strukturierendes Verhalten:** Auf Ziele verweisen, Prioritäten setzen, Vorschläge zum Verfahren machen oder erfragen, Aufgaben verteilen, Zusammenfassen, Visualisieren, Konkretisieren
- **Informationsweitergabe und Wissenstransfer:** Wissensaustausch, Wissenstransfer, Weitergabe von Informationen oder Wissen
- **Sonstiges** Beiträge die zwar relevant sind aber nicht eingeordnet werden können (möglichst nur im Notfall verwenden da nicht interpretierbar)

Bei act4teams wurde für die Kodierung die Software INTERACT von Mangold International benutzt. Dieses Programm ist zu umfangreich für eine Echtzeitkodierung. Deshalb wurde für act4teams-SHORT eine Java-Applikation entwickelt, mit der das Verhalten der Teilnehmer in den Meetings kodiert werden konnte.

2.3 Nielsen Heuristik

Folgende Beschreibung der Nielsen Heuristik basiert auf einem Artikel von Nielsen und Molich [30].

Heuristische Evaluierung ist einer informelle Methode der Usability Evaluierung von User Interfaces. Dabei werden den Personen, welche die Evaluierung durchführen, indem sie ein User Interface betrachten und bewerten, einige Heuristiken zur Orientierung (für die Evaluierung) vorgegeben. Mit Hilfe dieser Heuristiken kann bereits eine kleine Menge von Personen (drei bis fünf werden genannt), den Großteil der Probleme (bis zu 90%) eines User Interfaces entdecken. Nielsen [29] stellt seine zehn Heuristiken so vor (wörtlich übersetzt):

1. **Visibility of system status:** Der Nutzer sollte immer über den aktuellen Status des Systems Bescheid wissen.

2. **Match between system and the real world:** Das Design sollte die Sprache des Nutzers sprechen. Informationen sollten in einer natürlichen und logischen Ordnung erscheinen und Konventionen der realen Welt folgen.

3. **User control and freedom:** Der Nutzer sollte immer die Möglichkeit haben aus einem ungewollten Systemzustand zu entkommen, ohne durch einen langen Dialog aufgehalten zu werden. Beispiele für solche Funktionen sind undo und redo.

4. **Consistency and standards:** Nutzer sollten sich niemals fragen müssen, ob verschieden Wörter, Situationen oder Aktionen das Selbe bedeuten, deshalb sollten Industrie- und Plattform-Konventionen eingehalten werden.

5. **Error prevention:** Fehler gar nicht erst zu ermöglichen, durch sogfältiges Design, ist besser, als gute Fehlermeldungen.

6. **Recognition rather than recall:** Der Nutzer sollte sich niemals Informationen zwischen verschiedenen Teilen des Interfaces merken müssen. Diese Informationen sollten deshalb immer leicht abrufbar sein.

7. **Flexibility and efficiency of use:** Erfahrene Nutzer sollten die Möglichkeit haben, Abkürzungen im System nutzen zu können.

8. **Aesthetic and minimalist design:** Interfaces sollten keine irrelevanten oder selten genutzten Informationen enthalten. Visuelle Layouts sollten den Prinzipien Kontrast, Wiederholung, Ausrichtung und Nähe folgen.

9. **Help users recognize, diagnose, and recover from errors:** Fehlermeldungen sollten in natürlicher Sprache sein und dem Nutzer eine Lösung des Fehlers anbieten.

10. **Help and documentation:** Es ist zwar besser, wenn ein System ohne Dokumentation genutzt werden kann, aber Hilfe sollte leicht zu finden sein.

2.4 Webframeworks

Das Frontend moderner Websites besteht aus einer Kombination aus HTML für die Struktur der Website, JavaScript um die Website interaktiv zu machen und Cascading Style Sheets (CSS) für Farben und Formen der einzelnen Elemente der Website [28]. Um die Arbeit mit diesen drei Sprachen zu vereinfachen werden Webframeworks verwendet. Moderne Webframeworks erleichtern das Erstellen von Websites, indem sie fertige Komponenten, wie zum Beispiel Buttons, Icons oder auch Tabellen, zur Verfügung stellen, welche innerhalb der Rahmenbedinungen des Frameworks zusammengesetzt werden können. Beliebte Frameworks haben aktive Open Source Communitys und viele Ressourcen und Tutorials, wodurch das Lernen und der Umgang mit diesen Frameworks vereinfacht wird. Diese und weitere Vorteile von Webframeworks können diesem Artikel von Pekarsky entnommen werden [31].

2.5 Das Webframework React.js

React.js ist eine Open Source JavaScript Bibliothek von Facebook Open Source [11], die zur Konstruktion von User Interfaces und Single-Page Applikationen verwendet wird. Das besondere an Single-Page Applikationen ist, dass alle Teile der Website, also HTML, JavaScript und CSS, genau einmal vom Server abgerufen und geladen werden [10]. Dadurch werden diese Anwendungen reaktionsfähiger, da sich nur die Ansicht des Nutzers im Webbrowser ändert. Das Framework hat eine einsteigerfreundliche Dokumentation [9] und unterstützt die Verwendung von JavaScript-Packages zum Importieren von fertigen Komponenten. Es ist außerdem auch TypeScript kompatibel. Typescript ist eine Open Source Programmiersprache

die auf Javascript aufbaut und um statische Typisierung erweitert, wodurch Javascript-Code verständlicher und wartbarer wird [27].

React.js hat eine große Community, die in den letzten Jahren auch immer weiter gewachsen ist. Nach der jährlichen stackoverflow.com Entwicklerumfrage von 2021 [35] ist React.js seit 2021 das am meisten genutzte Webframework der Umfrageteilnehmer. Auf bekannten Webseiten von Packageanbietern wie zum Beispiel Yarn [7] werden über 77.000 Packages mit dem Stichwort "react" angeboten (Stand: August 2021).

2.6 Packages

In diesem Teil werden die einzelnen Packages vorgestellt, die für die Erstellung der Webanwendung verwendet wurden. Für die Paketverwaltung wurde der Package Manger Yarn [7] verwendet. Dieser kümmert sich um das Herunterladen und Aktualisieren der Packages. Der Package Manager wird auch zum lokalen Hosten der Webanwendung verwendet.

2.6.1 Material-UI

Material-UI ist ein User Interface Package, welches Googles Material Design Richtlinien [12] für React umsetzt [25]. Es stellt eine Menge von User Interface Komponenten für verschiedene Zwecke zur Verfügung und wird auch von großen Firmen wie Netflix, Amazon oder auch Spotify benutzt.

2.6.2 Export-to-CSV

Ein allgemeines Datenformat, welches von den meisten Tabellenkalkulationsprogrammen akzeptiert wird ist CSV (Comma-Separated Values). Das Package export-to-csv [1] ist ein Package, welches den Export von CSV-Dateien in Webanwendungen ermöglicht. Außer einem Komma werden auch weitere Separatoren wie Semikolons, Leerzeichen oder Tabulatoren akzeptiert. Um die Daten exportieren zu können, müssen die Daten als Array von JSON(JavaScript Object Notation)-Objekten übergeben werden. Das Format der Objekte ist dem Nutzer überlassen, der Tabellenkopf in den CSV-Dateien besteht jeweils aus den Feldern des Objektes.

Zum Beispiel ist das hier [{id: 1, name:Max}] ein Array mit genau einem JSON-Objekt. Die Felder, "id" und "name", würden dann beim Export zum Tabellenkopf werden und die "1" und "Max" wären in der zweiten Zeile der Tabelle, zugeordnet zu ihrem Feld.

2.6.3 react-papaparse

React-papaparse [4] ist ein Package, welches den Import von CSV-Dateien in React-Anwendungen ermöglicht. Es bietet Funktionen zum Konvertieren

von Strings im CSV-Format in ein Array von JSON-Objekten, da es den Separator automatisch erkennt. Außerdem akzeptiert es sowohl das Hochladen von lokalen Dateien, als auch von Dateien hinter eine URL. Dafür stellt es eine Click and Drag Komponente zur Verfügung.

2.6.4 nivo

Nivo [32] bietet 26 verschiedene Arten von Diagrammen an und ist modular aufgebaut. Das heißt man muss zusätzlich zu dem Kernmodul noch die jeweiligen Diagrammarten, die verwendet werden sollen, herunterladen. Die Diagramme diese Packages sind visuell ansprechend und viele Teile der Diagramme (z.B. Daten, Achsenbeschriftung, Farben, Legenden, Tooltips) können angepasst werden. Es können auch eigene Komponenten eingebunden und Mausklick-Interaktionen mit den Diagrammen definiert werden, um zum Beispiel zwischen verschiedenen Datensätzen zu wechseln.

Kapitel 3

Konzept

In diesem Kapitel wird das Konzept für die Webanwendung erarbeitet. Dafür werden verschiedene Layout-Möglichkeiten der Kodierungsansicht diskutiert, die dem Kunden dieser Anwendung in einem Anforderungsgespräch vorgestellt wurden. Es werden auch die Ergebnisse des Anforderungsgesprächs vorgestellt und mögliche Diagrammarten für die Auswertung der erhobenen Daten gezeigt. Abschließend wird die Wahl der Farben für diese Webanwendung begründet.

3.1 Mockups und Layoutdiskussion

Eine Funktion, die in dieser Webanwendung implementiert werden soll, ist die Zuweisung von Aktionen zu bestimmten Teilnehmern. Da dies bereits vor dem Anforderungsgespräch mit den Kunden bekannt war, wurden als Vorbereitung einige Mockups der Kodierungsansicht erstellt. Die Kodierungsansicht wird hier auch *Counter*View genannt, weil Aktionen der Teilnehmer von bestimmten Kategorien *gezählt* werden. Die Mockups zeigen unterschiedliche Anordnungen der Teilnehmer und Aktionen, sowie verschiedene Arten von Buttons, und sollten als Visualisierung für den Kunden genutzt werden.

Ein wichtiger Faktor, der für den Entwurf dieser Anwendung beachtet werden muss, ist, dass der Java-Prototyp vor allem mit einem Microsoft Surface mit einem Touchscreen, bedient wurde. Es war also davon auszugehen, dass das Layout der Anwendung möglichst touch-kompatibel bleiben und deshalb mit großen, leicht auswählbaren Elementen konzipiert werden sollte.

In den Mockups für das Layout, die hier folgen, sind die Buttons für die Teilnehmer immer lila und Buttons für Aktionen immer hellblau gefärbt. Diese Färbung war nur als Orientierung zwischen den vielen Mockups gedacht.

3.1.1 Teilnehmer als Liste und Aktionen als 3x3-Raster

Diese Anordnung (Abbildung 3.1) soll so nah wie möglich am Java-Prototypen bleiben und ihn nur um eine Darstellung für die Teilnehmer erweitern. Der Grund dafür ist, dass erfahrene Nutzer der Java-Applikation sonst nicht nur das Auswählen der Teilnehmer, sondern auch das Kodieren selbst nochmal neu lernen müssten, da die Positionen der Aktionen nicht mit dem Muskelgedächtnis (muscle memory) der Kodierer übereinstimmen würde.

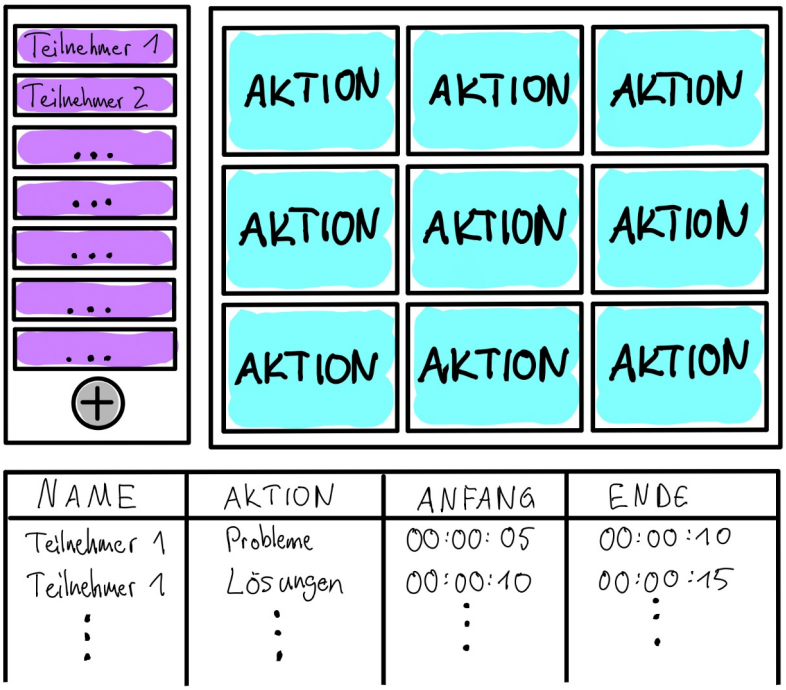

Abbildung 3.1: Mockup der Kodierungsansicht (CounterView)

Rechts ist ein 3x3-Raster aus Buttons mit den neun Kategorien (Aktionen) von act4team-SHORT zu sehen. Links daneben sieht man eine Liste aus Teilnehmern des Meetings, mit einem Plus-Button am unteren Ende, welcher die Liste erweitern sollte. Unter diesen beiden Elementen befindet sich ein Protokoll.

Viele Gründe sprechen für die 3x3 Anordnung der Buttons der Aktionen. Sie wurde bereits beim Java-Prototypen verwendet und ist zusätzlich eine Anordnung die oft im Alltag vorkommt (bspw. Taschenrechner, Telefontastaturen, Nummernblock von Computertastaturen). Der Bewegungsablauf bei der Nutzung einer 3x3 Anordnung sollte also sowohl geübten Nutzern der Java-Applikation als auch neuen Nutzern, natürlich vorkommen, da jede Bewegung von der Mitte aus in eine Richtung eine andere Aktion anzielt. Der Bewegungsablauf ist auch allgemein effizienter, als beispielsweise bei

einer Liste. Die Distanz vom mittleren Button zu einem der angrenzenden Buttons ist kürzer, als der Weg von der Mitte einer Liste zum ersten oder letzten Element in der Liste. Bei einer Liste mit drei oder weniger Elementen wäre das nicht der Fall, aber hier wäre eine Liste neun Elemente lang, da es neun Kategorien gibt. Da die Kodierung, also das Zählen von Aktionen, die wichtigste Funktion der Applikation ist, sollte sie auch im Zentrum der Applikation angezeigt werden. Dadurch können große Buttons, die leicht anzuklicken/zu berühren sind, verwendet werden.

Eine Liste als Datenstruktur für die Teilnehmer ist sinnvoll, weil die Anzahl der Teilnehmer in Meetings nicht immer dieselbe ist und eine Liste theoretisch immer erweitert werden kann, ohne das Layout zu verändern. Listen können auch leicht überflogen und ihre Elemente leicht gefunden werden. Deshalb sollten hier in der Praxis Grenzen gesetzt werden, um das Layout der Anwendung nicht unübersichtlich zu machen und die guten Eigenschaften einer Liste zu behalten. Der Nachteil einer Liste ist jedoch, dass keine direkte räumliche Verbindung zwischen den digitalen Teilnehmern in der Liste und den realen Teilnehmern, wie sie im Meetingraum sitzen, besteht.

Die Anordnung mit einer Liste auf der linken Seite und der Hauptfunktionalität (dem Body) in der Mitte/auf der rechten Seite ist ganz typisch für User Interfaces von Webanwendungen in Desktop-Ansicht (Beispiele: Youtube, Facebook, Wikipedia). Sie erlaubt außerdem die intuitive Bedienung mit beiden Händen, da die linke Hand sich nur auf die Teilnehmerzuweisung und die rechte Hand nur auf das Zählen der Aktionen konzentrieren muss.

Das Protokoll ist hier nur als Konzept angedacht und gehört nicht spezifisch zu dieser Anordnung. Der Gedanke des Protokolls war, dass, anders als beim Prototypen, nicht nur die Anzahl der Aktionen, sondern auch die Länge der Aktionen der Teilnehmer protokolliert werden und dem Kodierer direkt angezeigt werden könnte. Diese Funktion bietet die INTERACT Software an (die für act4teams verwendet wird) und hätte weitere Informationen über die kodierten Meetings geben können. Der Kodierer hätte dann zusätzlich über das Protokoll, falsch gezählte Aktionen wieder löschen können.

3.1.2 Teilnehmer räumlich am Rand angeordnet

Die vier Mockups in diesem Teil (Abbildung 3.2 und 3.3) unterscheiden sich nur in ihrer Anordnung der Aktionen, deshalb werden zuerst die Variationen der Aktionen in der Mitte und abschließend die Anordnung der Teilnehmer für alle zusammen diskutiert.

In Abbildung 3.2 sind zwei Mockups zu sehen, bei denen die Aktionen (in blau) zentral, in der Form eines runden Tisches positioniert sind. Die Teilnehmer (in lila) sind in beiden Mockups um den Tisch herum in einem Rechteck, als rechteckige Buttons, angeordnet.

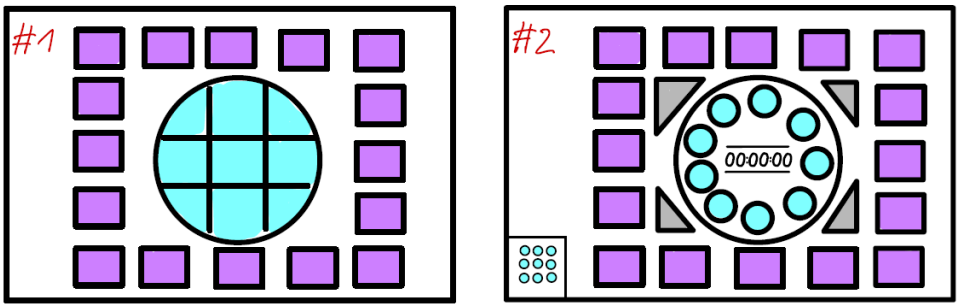

Abbildung 3.2: Mockups #1 und #2 mit Teilnehmern am Rand

In #1 füllen die Buttons für die Aktionen den Kreis voll aus und die meisten Aktionen haben eine große Fläche. Es gibt jedoch vier Buttons die durch diese Anordnung eher dreieckig sind und eine kleinere Fläche als die anderen Buttons haben, wodurch sie leichter verfehlt werden könnten. Es könnte auch unbewusst eine Hierarchie der Buttons wahrgenommen werden, da größere Elemente in der Regel wichtiger sein sollten. Diese beiden Punkte könnten zu einer Verfälschung der Ergebnisse beim Kodieren führen und das sollte so gut wie möglich verhindert werden.

In #2 werden, statt den üblichen eckigen, runde Buttons für die Aktionen verwendet und in einem Kreis angeordnet, ähnlich wie bei einem Telefon mit Wählscheibe. Diese Buttons könnten zum Beispiel Symbol-Buttons oder auch Buttons mit den Anfangsbuchstaben bzw. Abkürzungen der Aktionen sein. Ein Problem bei dieser Art von Buttons könnte sein, dass die Lernkurve für neue Nutzer zu steil wird, da sie genau Wissen müssten, wofür welches Symbol bzw. welche Abkürzungen stehen. Auch geübte Nutzer der Java-Applikation müssten sowohl die Buttons selbst als auch die Anordnung neu lernen. Um die Lernkurve für geübte Nutzer ein wenig abzuflachen, könnten die Buttons auch in dem gewohnten 3x3-Raster angeordnet werden (Skizze in #2 unten links). Allerdings würde dadurch viel unbenutzbare Fläche zwischen den Buttons entstehen. Besser als in #1 ist in #2, dass die Buttons alle dieselbe Fläche haben, jedoch ist die Fläche allgemein kleiner und dadurch schwieriger zu treffen als in #1. Die Fläche die durch die kleineren Buttons in runder Anordnung entsteht, kann aber auch produktiv genutzt werden. Hier zum Beispiel wird die Stoppuhr des Meetings in der Mitte angezeigt. In #2 kann man auch sehen, dass die äußeren Ecken (in grau) für weitere Buttons, zum Beispiel Start oder Stopp für die Stoppuhr, genutzt werden könnten.

Die beiden Mockups in Abbildung 3.3 (auf der nächsten Seite) sind ähnlich wie die Mockups in Abbildung 3.2 aufgebaut, also Teilnehmer außen (in lila) und Aktionen in der Mitte (in blau).

Mockup #3 zeigt eine Variation des Mockups #2. Hier werden statt den runden Buttons für die Aktionen, ovale bzw. längliche Buttons verwendet.

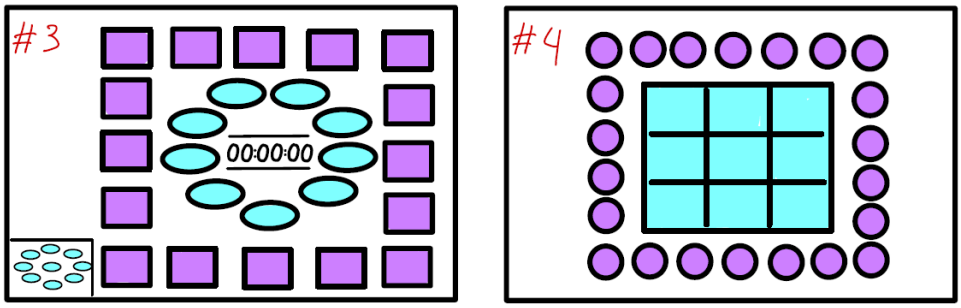

Abbildung 3.3: Mockups #3 und #4 mit Teilnehmern am Rand

Die Buttons sollen auch hier, durch ihre Anordnung, einen runden/ovalen Tisch darstellen, wie in #1 und #2. Ein klarer Vorteil gegenüber #2 ist, dass die Buttons durch ihre größere Fläche mehr Platz zum Klicken/Berühren und auch beschriften bieten. Dadurch könnten die Kategorien ausgeschrieben werden. Auch hier kann die Mitte der Fläche für die Stoppuhr genutzt werden, wenn die Buttons im Kreis angeordnet werden. Bei einer 3x3 Anordnung, wie in #2 unten links, würde mit den ovalen Buttons jedoch die Form des runden Tisches nicht mehr passen. Daher müssten die Buttons wie in #3 unten links angeordnet werden, um die Form beizubehalten. Der Nachteil bei dieser Anordnung ist, dass der Button in der Mitte wieder unbewusst als wichtiger wahrgenommen werden könnte, da er alleine dasteht und von den Anderen umrahmt wird, wie die Mitte einer Zielscheibe. Dieses Argument würde jedoch auch teilweise auf die anderen 3x3 Anordnungen zutreffen (bspw. in #4). Durch die Nähe der Buttons in den anderen 3x3 Anordnungen, ist das Argument in den Fällen nicht so stark, wie hier.

In #4 sind die Buttons für die Aktionen im gewohnten 3x3-Raster angeordnet und bilden so, einen rechteckigen Tisch. Die Teilnehmer sind hier nur als Beispiel runde Buttons und könnten genauso gut in den Mockups #1, #2 oder #3 verwendet werden. Sie bilden hier jedoch einen angenehmen Kontrast zu den eckigen Buttons in der Mitte und können dadurch leichter von diesen unterschieden werden. Runde Buttons für die Teilnehmer könnten sinnvoller als runde Buttons für die Aktionen sein. Die Tischformation der Buttons in der Mitte erzeugt eine räumliche Beziehung zwischen der Person in der Applikation und der Person im Meetingraum. Daher könnten die Initialen des Teilnehmers oder auch Pseudonyme (wie z.B. “Teilnehmer XY” oder auch abgekürzt “TN XY”, wobei XY eine Zahl ist) für eine korrekte Zuweisung bei der Kodierung ausreichen. Wenn die Namen der Teilnehmer doch ausgeschrieben werden sollten, dann würden sich die rechteckigen Buttons wiederum eher anbieten, da sie eine größere Schreibfläche anbieten. Die große Fläche hilft auch beim Anklicken/Berühren, erlaubt aber weniger Teilnehmer insgesamt, als die runden Buttons. Als Kompromiss zwischen mehr Fläche und mehr Teilnehmer könnten quadratische Buttons verwendet

werden, aber auch diese könnten bei zu langen Beschriftungen schnell überfüllen und unästhetisch werden.

Abschließend kann man sagen, dass das stärkste Argument für die Anordnung der Teilnehmer wie in #1 bis #4, die räumliche Beziehung der Teilnehmer zwischen der digitalen und der realen Welt ist, welche die Zuweisung der Aktionen zu Teilnehmern vereinfachen könnte und durch eine einfache Liste wie in Abbildung 3.1 nicht gegeben ist. Die Flächen der Buttons können auch größer gestaltet werden, als bei einer Liste, wodurch sie leichter anklickbar/berührbar werden. Die Fläche in der Mitte kann gut für die Buttons der Aktionen genutzt werden und könnte auch je nach Form des Tisches (rund oder eckig) im Meetingraum angepasst werden. Die Kodierung mit zwei Händen könnte hier jedoch schwieriger sein als in Abbildung 3.1, da man die Aufgaben der Auswahl der Teilnehmer und dem Kodieren der Aktionen nicht mehr auf die linke und rechte Hand aufteilen könnte. Man müsste also umgreifen und dies erfordert mehr Koordination als die einfache Aufteilung in linke und rechte Hand. Das größte Problem, welches bei dieser Anordnung der Teilnehmer entstehen könnte ist, dass die Teilnehmer während eines Meetings ihre Sitzposition ändern und dadurch die räumliche Beziehung verloren geht. Ein weiterer Fall der auch dazu gehört ist, dass der Meetingraum entweder gar keinen Tisch oder eine ganz andere Anordnung (z.B. U-Form oder Tischreihen) hat, wodurch erst gar keine räumliche Beziehung erstellt werden könnte. Diese beiden Fälle könnten durch die Option der Verschiebung der einzelnen Elemente nur teilweise gelöst werden, da die Form größtenteils durch die großen Buttons in der Mitte gegeben wird. Das User Interface würde dadurch auch unnötig komplex werden und könnten den Kodierer vom kodieren ablenken, was ohnehin bereits eine hohe kognitive Last für diese Person darstellt.

3.1.3 Teilnehmer im Body

In diesem Teil werden drei Mockups diskutiert, in denen die Aktionen als Liste am Rand angeordnet sind. Auch hier kommt die Diskussion zu der Liste selbst am Ende.

In Abbildung 3.4 sind zwei Mockups zu sehen, in denen die Buttons für die Aktionen als Liste am linken Rand angeordnet werden.

In #5 sieht man rechts von der Liste der Aktionen eine räumliche Anordnung der Teilnehmer, als runde Buttons, um einen ovalen Tisch herum. Die Fläche in der Mitte des Tisches wurde für die Darstellung der Stoppuhr genutzt. Ein großer Vorteil der räumlichen Darstellung hier, im Vergleich zu Mockups #1 bis #4, ist, dass die Form des Tisches bzw. die Anordnung der Teilnehmer nicht mehr durch die Buttons der Aktionen in der Mitte beeinflusst wird, da diese am Rand sind. Dadurch könnten viele verschiedene Teilnehmer-Tisch-Kombinationen möglich werden. Man könnte dem Kodierer vielleicht auch die Möglichkeit geben eine Form zu malen

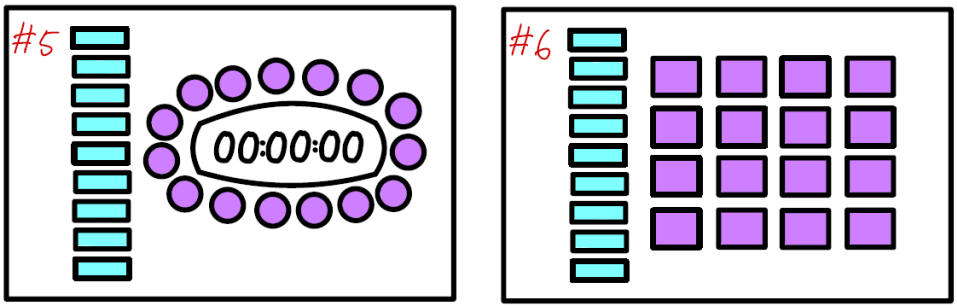

Abbildung 3.4: Mockups #5 und #6 mit den Aktionen als Liste und Teilnehmern im Body

und die Teilnehmer selbst zu positionieren. Die Nachteile der räumlichen Anordnung bestehen jedoch auch hier. Sobald das Meeting angefangen hat, sollten die Elemente eigentlich nicht mehr bewegbar sein, um nicht aus Versehen die Ordnung zu verlieren und den Kodierer vom kodieren abzulenken. Dadurch könnte wiederum die räumliche Beziehung verloren gehen, wenn Teilnehmer ihre Plätze wechseln.

In Mockup #6 sind die Buttons für die Teilnehmer als Raster angeordnet. Hier verliert man die Vorteile einer räumlichen Anordnung der Teilnehmer. Man gewinnt jedoch die Vorteile eines Rasters. In dem Mockup sind zwar nur 16 Teilnehmer eingezeichnet, es könnten jedoch sehr viel mehr Teilnehmer als in #5 dargestellt werden, da kein Platz für den Tisch verbraucht werden muss. Weil die Teilnehmer hier im Zentrum dargestellt werden, liegt der Fokus in dieser Anordnung auf den Teilnehmern, obwohl die Hauptaufgabe dieser Applikation, das Zählen der Aktionen sein sollte.

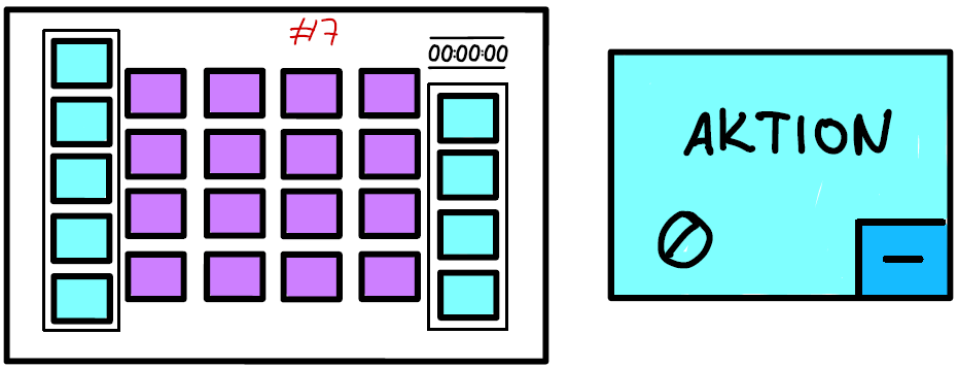

Abbildung 3.5: Mockup #7 und Button der Java-Applikation

In Abbildung 3.5 ist Mockup #7 zu sehen und der Button der Aktionen, wie er in der Java-Applikation verwendet wurde. Auf den Button wird in Abschnitt 3.1.4 eingegangen.

Mockup #7 teilt, anders als #5 und #6, die Liste der Aktionen in zwei Teile auf und ordnet diese jeweils links und rechts am Rand an. Statt fünf Aktionen links und vier Aktionen rechts wären auch andere Aufteilungen denkbar wie sechs links und drei rechts oder eine dieser beiden Varianten gespiegelt. Der freie Platz über der kleineren Liste der Aktionen kann zum Beispiel für die Stoppuhr verwendet werden (oben rechts). Die Teilnehmer sind hier in der Mitte als Raster angeordnet (wie in#5), es könnte hier aber auch die räumliche Anordnung wie in #6 verwendet werden. Hier können sehr viel größere Buttons für die Aktionen verwendet werden, als in einer einfachen Liste, aber durch die Aufteilung der Aktionen entsteht ein logischer Bruch. Die Aktionen müssten also entweder so aufgeteilt werden, dass Kategorien die einen logischen Bezug zueinander haben (bspw. Probleme, Lösungen und Verknüpfungen und Vernetzungen) in der selbe Gruppe sind oder diese Aufteilung wäre nicht nutzbar, da sie die Nutzer verwirren könnte.

Abschließend kann man sagen, dass die Anordnung der Buttons für die Aktionen als eine Liste, viele Möglichkeiten für die Anordnung der Teilnehmer eröffnet, da eine große Fläche in der Mitte frei wird. Jedoch werden dadurch die Teilnehmer zum Fokus der Applikation, obwohl das Zählen der Kategorien der Fokus der Anwendung sein sollte. Außerdem wurden in einer unveröffentlichten Eye-Tracking-Studie von Schween und Klünder [34] die Anordnungen der Aktionen als Liste und als 3x3-Raster getestet. Wie sich zeigte, mussten die Testpersonen sich bei der Liste stärker auf das Finden der Aktionen konzentrieren, als bei dem 3x3-Raster, wodurch sie mehr vom Meeting abgelenkt werden. Das 3x3-Raster für die Buttons der Aktionen, sollte also definitiv gegenüber der Listen-Anordnung, bevorzugt werden.

3.1.4 Mockups für Buttons

In diesem Abschnitt werden verschiedene Buttons für die Aktionen diskutiert. Auch hier sind die Farben nur als Platzhalter gedacht.

In der Java-Applikation wurde der Button, wie er in Abbildung 3.5 rechts zu sehen ist, verwendet. Der Button besteht aus insgesamt vier Teilelementen: Dem Schriftzug der Kategorie(Aktion), dem Zähler für diese Aktion (die Null unten links), einer großen Schaltfläche zum Hochzählen (die gesamte Fläche außer der unteren rechten Ecke) und einer kleinen Schaltfläche zum Herunterzählen (die kleine Fläche in der unteren Rechten Ecke mit dem Minus als Schriftzug). Dieser Buttons ist sehr gut, wenn der Button groß dargestellt werden kann, da man auf einen Blick den Status der Aktion erkennen kann. Man muss jedoch mindestens einmal ausprobieren oder raten, um herauszufinden, was die große Fläche macht. Wenn der Bildschirm kleiner ist, wie zum Beispiel bei einem Smartphone, dann bietet sich diese Art von Button weniger an, da der Button zum Herunterzählen sehr klein werden würde. Auch für eine Listenanordnung wäre dieser Button

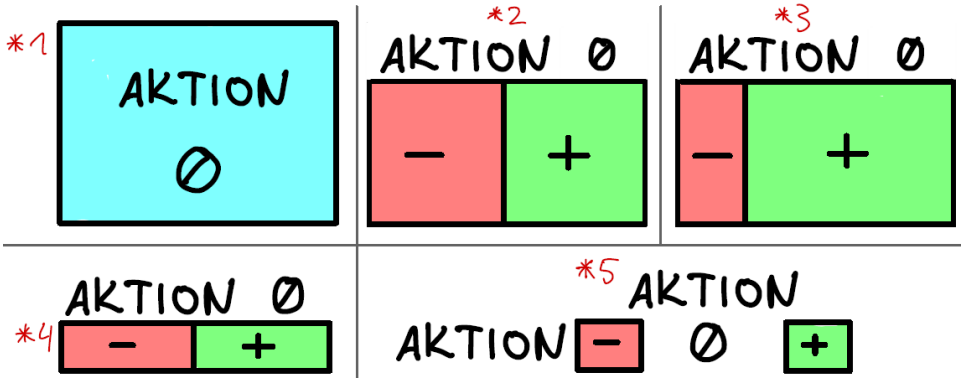

Abbildung 3.6: Fünf verschiedene Mockups für Buttons der Aktionen

nicht nutzbar. Deshalb werden in Abbildung 3.6 verschiedene Buttons gezeigt, die einzelne Aspekte des Buttons verbessern.

Mockup ∗1 entfernt die Fläche zum herunterzählen ganz, bleibt aber sonst genauso wie der Java-Applikations-Button. Das Herunterzählen könnte durch ein langes Klicken/Berühren (long press oder press and hold) oder auch Rechtsklick implementiert werden. Ein Vorteil, der durch das Weglassen des Minus-Buttons entsteht, ist, dass der Button sehr gut auf kleine Bildschirme herunterskaliert werden könnte. Der Nachteil ist, dass die Funktionsweise des Herunterzählens ohne Erklärung nicht klar ist.

Mockups ∗2 und ∗3 sind sich recht ähnlich. Bei beiden wurde die Beschriftung für die Aktion und den Zähler über die Buttons gelegt und die Fläche in einen Plus-Button zum Hochzählen und einen Minus-Button zum Herunterzählen eingeteilt. Bei ∗2 haben die Buttons die gleiche Fläche, bei ∗3 ist die Fläche für den Plus-Button größer. Bei ∗2 und ∗3 ist sofort klar, welche Funktion mit welcher Fläche verbunden ist, anders als bei der großen Fläche bei den zwei Buttons davor (Java-Applikation und ∗1). Sie wirken dadurch jedoch weniger elegant und eher funktional. Mockup ∗3 wäre für diese Applikation wahrscheinlich besser geeignet als ∗2, da bei ∗2 der Minus-Button genauso groß ist, wie der Plus-Button und es dadurch öfter zu falschen klicks kommen könnte.

Mockup ∗4 ist eine schmalere Variante der Buttons ∗2 und ∗3 und würde sich, an deren Stelle, für eine Listenanordnung eignen.

Bei Mockup ∗5 sind der Minus- und der Plus-Button getrennt. Die Fläche zwischen den Buttons wurde für den Zähler genutzt. Die Beschriftung für die Aktion kann hier sowohl an der Seite als auch darüber platziert werden. Die visuelle Klarheit ist in diesem Mockup wahrscheinlich besser als in allen Mockups davor, da der Zähler eine zentrale Position hat. Auch dieses Mockup würde sich gut für einer Listenanordnung eignen, vor allem mit der Beschriftung an der Seite. Ein großer Nachteil ist hier, dass die Buttons sehr klein und deswegen schwer zu treffen sein könnten.

Insgesamt wurden in diesem Abschnitt viele verschiedene Layouts und Buttons mit ihren Vor- und Nachteilen vorgestellt und eine Nutzerstudie für den Vergleich zwischen den Layout- und Button-Kombinationen würde sich anbieten. Jedoch hängt die Entscheidung für ein bestimmtes Layout am Ende vor allem von den Wünschen der Kunden ab, da sie die Applikation nutzen wollen.

3.2 Anforderungserhebung

Um das Ziel dieser Arbeit zu Erreichen, wurde ein Anforderungsgespräch mit den Kunden, Psychologen der Technischen Universität Braunschweig, geführt. Die Psychologen sind Teil des Fachgebietes für Arbeits-, Organisations- und Sozialpsychologie der Technischen Universität Braunschweig, welches von Professor Kauffeld, einer der Autorinnen von act4teams, geführt wird. Act4teams-SHORT wurde in Kooperation mit dem Fachgebiet für Software Engineering der Leibniz Universität Hannover entwickelt.

Anforderungsgespräche werden üblicherweise zu Beginn eines Software-Projektes mit den Kunden geführt, um die Anforderungen der Kunden, an die zu entwickelnde Software, zu erheben und zu dokumentieren. Das konkrete Ziel dieses Anforderungsgespräches war, herauszufinden, welche Funktionen des Java-Prototypen modifiziert werden sollten und welche weiteren Funktionen gewünscht sind, außer der Zuweisung der Aktionen eines Teilnehmers zu einer bestimmten Kategorie. Außerdem wurden die Mockups aus Abschnitt 3.1 vorgestellt, um mit den Kunden das Layout festzulegen. Das Mockup welches die Kunden am meisten überzeugt hat, ist in Abbildung 3.1 zu sehen. Es handelt sich dabei um das Mockup mit den Teilnehmern als Liste an der linken Seite und den Aktionen als 3x3-Raster in der Mitte. Der Grund dafür ist, dass die Kunden sich nur minimale Veränderungen an der Anordnung der Kodierungsansicht und explizit eine Liste für die Teilnehmer gewünscht haben. Das Log wurde nicht erwünscht, da die Information über die Länge einer Aktion nicht benötigt wird. Auch die Buttons sollten so bleiben, wie bei der Java-Applikation. Ein Wunsch der Kunden für die neue Applikation war, weitere Darstellungsmöglichkeiten für die Daten, die beim Kodieren erhoben werden, da diese sehr bei der Auswertung und Kommunikation mit den Teilnehmern, am Ende eine Meetings, helfen. Hier ist eine Liste der Anforderungen, die sich aus dem Anforderungsgespräch ergaben:

R01: Der Benutzer kann auf einen Button drücken und dadurch eine Aktion zählen, welche mit einem Zeitstempel markiert wird

R02: Der Benutzer kann die Aktionen einem Teilnehmer zuweisen

R03: Nachdem einem Teilnehmer eine Aktion zugewiesen wurde, soll kein Teilnehmer ausgewählt sein

R04: Dem Benutzer werden neun Aktionen zum Zählen zur Verfügung gestellt

R05: Die Zeit wird während des Meetings gemessen

R06: Teilnehmer können zur Liste hinzugefügt und von ihr entfernt werden, bevor das Meeting startet

R07: Der Benutzer kann die Daten des Meetings als Excel-Datei exportieren

R08: Die Anwendung soll dem Nutzer ermöglichen die Daten auszuwerten

R09: Alte Meetings können wieder visualisiert werden

R10: Vor dem Beginn des Meetings kann der Benutzer die Anzahl der Teilnehmer, die Namen der Teilnehmer und den Namen des Meetings festlegen

R11: Die Anwendung soll als Webanwendung in einem Browser laufen

3.3 Auswertung der Daten

Die Kunden hatten sich im Anforderungsgespräch weitere Darstellungsmöglichkeiten für die Auswertung der Daten gewünscht. Diese Möglichkeiten werden hier erarbeitet. Die Farben in Abbildung 3.7 sind nur für das Mockup, zur besseren Unterscheidung, gedacht. Auch die Zahlen und Beschriftungen dienen nur zur besseren Kommunikation.

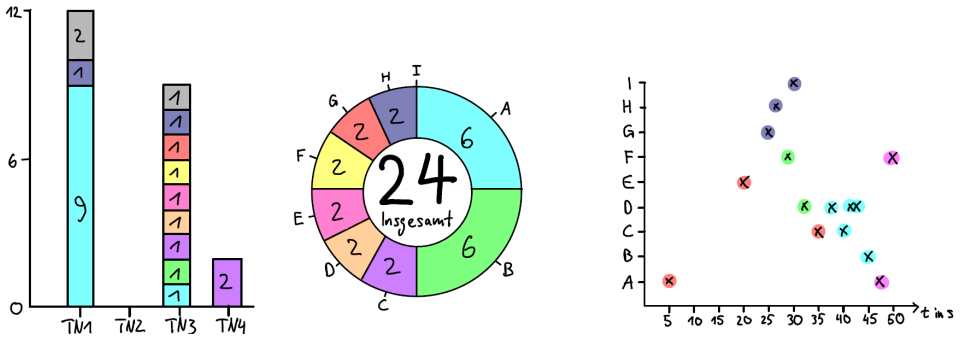

Abbildung 3.7: Mockups für drei mögliche Arten von Diagrammen, für die Visualisierung der Daten

Die Daten die bei der Kodierung eines Meetings erfasst werden bestehen aus drei Teilen: Einer Person, welche die Aktion ausgeführt hat, der Kategorie der Aktion, die ausgeführt wurde, und dem Zeitpunkt, an dem

die Aktion ausgeführt wurde. Es ist also naheliegend, den Fokus bei der Auswertung auf jeweils einen der drei Teile zu setzen.

Für die Visualisierung und Auswertung der Personen (im Bezug zu ihren Aktionen) bietet sich ein Säulendiagramm an. In Säulendiagrammen kann man Häufigkeiten sehr gut darstellen ([14], S.102). Man kann also sehr gut erkennen, wie oft eine Person eine bestimmte Aktion ausgeführt hat. Auch der Vergleich der Summen (der Aktionen) zwischen den Teilnehmern lässt sich sehr gut mit einem Säulendiagramm visualisieren. Noch besser als nur die Summen zu vergleichen, wäre ein Stapeldiagramm, welches die Summe durch ihre Höhe visualisiert und innerhalb der Säule die Anteile der jeweiligen Aktionen an der Summe zeigt. Ein Beispiel dafür ist in Abbildung 3.7 links zu sehen. Auf der x-Achse stehen die Namen der Teilnehmer (hier abgekürzt mit TN1 - TN4) und auf der y-Achse Zahlen (hier von null bis zwölf). Die einzelnen gefärbten Bereiche der Stapel repräsentieren Kategorien. Man kann auf einen Blick sehr viel erkennen, bspw., dass TN2 absolut keine Aktion in diesem Meeting gemacht hat oder dass TN1 sich am meisten beteiligt hat, aber dafür besonders von einer Kategorie. Bei TN3 konnte jede Kategorie genau einmal und bei TN4 nur eine Kategorie zweimal gezählt werden.

Kreisdiagramme eignen sich gut für die Darstellung von relativen Anteilen im Bezug auf eine Gesamtmenge ([14], S.106). Wenn das Meeting als Gesamtmenge betrachtet wird, dann kann man mit Hilfe eines Kreisdiagrammes, den Anteil der einzelnen Kategorien an einem Meeting, visualisieren. Man kann auch eine einzelne Kategorien als Gesamtmenge nehmen und die relative Verteilung, der Teilnehmer, an dieser Kategorie, zeigen. Kreisdiagramme eignen sich in diesem Fall eher als Stapeldiagramme, da diese, im Bezug auf das gesamte Meeting, nur eine einzelne riesige Säule darstellen würden (im Prinzip ist ein Kreisdiagramm auch nichts anderes als ein Stapeldiagramm, bei dem beide Enden verbunden wurden). Das klassische Kreisdiagramm hat eigentlich eine ausgefüllte Mitte, diese bietet jedoch keinen Mehrwert. Deshalb sollen hier Ringdiagramme (auch donut diagram) verwendet werden, da somit die Mitte für weitere Funktionen genutzt werden kann. Ein Beispiel ist in der Mitte von Abbildung 3.7 zu sehen. Das Mockup zeigt ein Ringdiagramm mit neun Kategorien (beschriftet A bis I). Man sieht sofort, dass zwei Kategorien (A und B) jeweils ein Viertel aller Aktionen des Meetings ausgemacht haben. Die andere Hälfte besteht aus sechs Kategorien die jeweils den gleichen Anteil haben (C bis H, jeweils 2 mal) und Kategorie I kam nicht einmal vor. Die Mitte des Diagramms wurde genutzt, um die Gesamtzahl der Aktionen des Meetings anzuzeigen (24 Insgesamt).

Das rechte Diagramm in Abbildung 3.7 zeigt, wie die Kategorien (auf der y-Achse) über der Zeit (auf der x-Achse) dargestellt werden können. Jedes Kreuz steht für eine Aktion eines Teilnehmers und jede Farbe für einen Teilnehmer. Man kann hier sehr gut erkennen, welche Aktion auf welche gefolgt ist, was wichtig sein könnte, um zu zeigen, dass auf ein Problem auch

mit einer Lösung eingegangen wurde. Man sieht hier auch sehr gut, wenn für eine längere Zeit nichts passiert (zwischen Sekunde fünf bis zwanzig) oder wenn in kurzer Zeit sehr viel passiert (ab 25 Sekunden).

3.4 Farben für die UI und die Datenauswertung

Damit Text auf einem Bildschirm lesbar ist, ist es wichtig einen hohen Kontrast zwischen dem Hintergrund und dem Text zu haben ([2], S. 73). Viele Websites haben dafür einen hellen Modus, also dunkle Schrift auf hellem Hintergrund, oder dunklen Modus, also helle Schrift auf dunklem Hintergrund. Der helle Modus soll besser lesbar sein [38] als der dunkle Modus. Jedoch soll das stark abhängig von der Helligkeit des Raumes und der Größe der Schrift sein [6] (dieser Artikel geht noch näher darauf ein [3]). Der dunkle Modus ist angenehmer für die Augen und kann in Geräten mit OLED-Bildschirmen Batterie sparen[5]. In dieser Umfrage[36] haben ca. 82%(von 243) der Teilnehmer angegeben, dass sie den dunklen Modus ihres Betriebssystems nutzen, und in dieser Umfrage[37] waren es auch 81.9% (von 2514 Teilnehmern) für Android-Geräte. Es wollen also viele Nutzer einen dunklen Modus. Deshalb wird diese Applikation als Schema helle Schrift auf dunklem Grund nutzen. Jedoch sollte, für die beste User Experience, dem Nutzer die Möglichkeit gegeben werden, zwischen dunklem Modus und hellem Modus zu wechseln. Weiße Schrift auf schwarzem Hintergrund wirkt sehr intensiv, daher wird meistens ein dunkleres Grau als Hintergrundfarbe verwendet und soll es auch in dieser Anwendung.

Für die Buttons der Aktionen wurde in einer unveröffentlichten Nutzerstudie von Schween und Klünder [34](die gleiche Studie wie für den Vergleich Liste und 3x3-Raster) eine Farbkodierung mit den ursprünglichen Farben der Kategorien für act4teams getestet (Dunkelblau für Professionelle Kompetenz, Hellblau für Methodenkompetenz, Rot für Sozialkompetenz, Gelb für Selbstkompetenz und Grau für keine Zuweisung [18]). Diese Farbkodierung soll das 3x3-Raster der Buttons für die Aktionen unübersichtlich gemacht und die Studienteilnehmer verwirrt haben. Deshalb soll eine einheitliche, möglichst neutrale, Farbe für diese Buttons gewählt werden, die sich vom dunklen Hintergrund abhebt. Für die Buttons der Aktionen wurde ein Hellblau und für die restlichen Buttons der UI (auch die Liste der Teilnehmer) ein helles Grau gewählt. Die Buttons der Aktionen haben ihre eigene Farbe, damit sie sich noch besser von dem Rest der UI abheben, da sie die wichtigste Komponente der UI sind.

Um die Kategorien in den Diagrammen der Datenauswertung gut unterscheiden zu können, werden neun verschiedene Farben benötigt. Die Farben die gewählt wurden, richten sich nach den Diagrammen von Klünder et al. [23]. Jedoch werden sie als Pastellfarben verwendet, da diese heller und dadurch besser erkennbar sind, vor dem dunklen Hintergrund. Für Dia-

gramme, in denen Teilnehmer visualisiert werden, benötigt man mindestens 20 verschieden Farben. Wenn man hier nur auf Pastellfarben zurückgreifen würde, dann könnte man die Teilnehmer nicht sehr gut unterscheiden. Deshalb werden für die Teilnehmer 20 möglichst unterschiedliche Farben gewählt, die auf dem Hintergrund noch erkennbar sind.

Kapitel 4

Implementierung

In diesem Kapitel soll die Implementierung des Konzeptes aus Kapitel 3 und der Endzustand der Webanwendung gezeigt werden.

4.1 Framework

In Kapitel 2.3 wurde gezeigt, dass das Nutzen von Webframeworks für die Entwicklung von Webanwendungen empfehlenswert ist. Für diese Webanwendung wurde das React.js Framework, da es sehr verbreitet ist und sich gut für diese Art von Anwendungen eignet. React bietet außerdem auch eine gute Grundlage für mögliche zukünftige Erweiterungen dieser Webanwendung. Beispiele für Erweiterungen könnten weitere Diagramme zur Auswertung, ein Smartphone User Interface oder auch eine Datenbankanbindung, für die Auswertung mehrerer Meetings, sein.

Als Programmiersprache wurde TypeScript [27] statt normalem JavaScript gewählt, da es den Code übersichtlicher und wartbarer macht.

4.2 Datenverwaltung

React.js ist ein Front-End-Framework und wird normalerweise in Kombination mit einem Back-End (wie zum Beispiel einer Datenbank) verwendet. Jedoch würde eine Datenbank diese Anwendung weniger portabel machen, da die Anwendung öfter, als nur beim ersten Laden der Anwendung, mit dem Server kommunizieren müsste. Bei Serverproblemen könnte die Anwendung auch langsamer werden. Außerdem konnten die Kunden nicht klar sagen, ob die Anwendung am Ende lokal ausgeführt oder über einen Server gehostet werden soll. Deshalb wurde der React-Context [8] für die Datenverwaltung gewählt, um die Anwendung möglichst portabel zu machen. Variablen in React sind eigentlich nur lokal verfügbar, das heißt Komponenten haben nur Zugriff auf Variablen die ihnen von ihrem Parent übergeben werden oder Variablen die in der Komponente selbst erstellt werden. Mit Hilfe des

React-Context kann man Variablen und auch Funktionen für ausgewählte Komponenten global verfügbar machen. In dieser Anwendung speichert der Kontext nicht nur die Daten des Meeting, sondern kann auch die Daten umformen, sodass sie mit den Packages, für die Visualisierung und den Export/Import der Daten, verwendet werden können.

4.3 Ansichten (Views)

Die vier Ansichten der Anwendung entsprechen drei Stufen eines Meetings: Vor, während und nach dem Meeting. Vor dem Meeting sollen in der InitView die Parameter des Meetings festgelegt werden. Während des Meetings sollen die Aktionen der Teilnehmer in der CounterView kodiert werden. Nach dem Meeting können die kodierten Daten in der DataView visualisiert und ausgewertet werden. Über die UploadView kann die 2. Stufe (während eines Meetings) übersprungen und direkt zur Auswertung in der DataView vorgegangen werden. Die Zustandsübergänge zwischen den Ansichten der Anwendung sind in Abbildung 4.1 zu sehen. In den folgenden Abschnitten werden die Funktionen der einzelnen Ansichten näher erklärt.

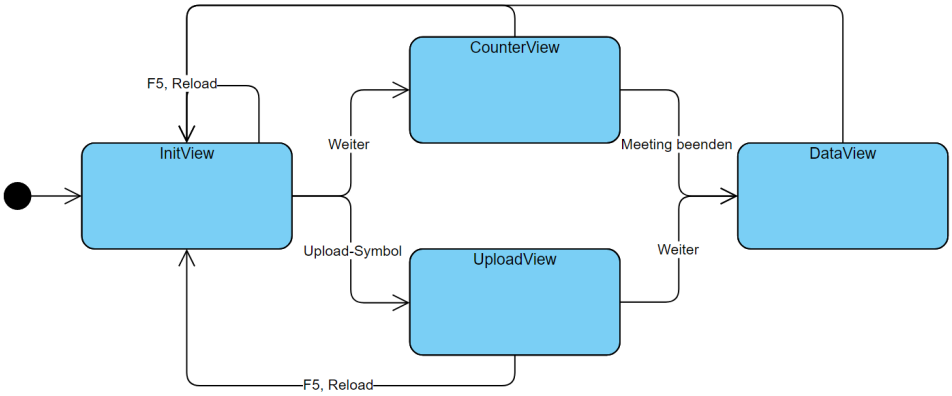

Abbildung 4.1: Zustandsübergangsdiagramm der Ansichten

4.3.1 InitView

Die InitView (Initialization View) ist die Startseite der Anwendung (Abbildung 4.2). Auf der linken Seite der Ansicht ist eine editierbare Liste aus Teilnehmern (hier dargestellt mit der maximalen Teilnehmerzahl). Rechts daneben ist ein Textfeld für den Meetingnamen und ein Textfeld für die Anzahl der Teilnehmer. Die editierbare Liste passt ihre Größe an, wenn die Anzahl der Teilnehmer verändert wird. Falls die Zahl kleiner als 0 oder größer als 20 ist, weist eine Fehlernachricht den Nutzer darauf hin, dass nur Zahlen von 0 bis 20 erlaubt sind. Per Default sind die Namen “Teilnehmer XY”,

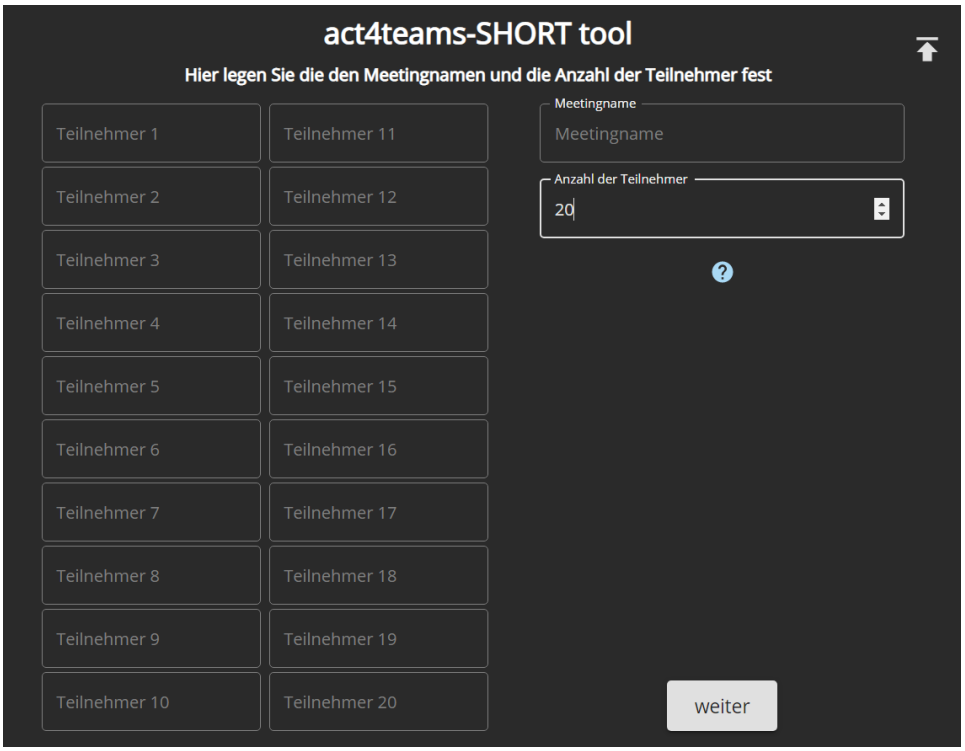

Abbildung 4.2: Screenshot der InitView

wobei XY eine Zahl von eins bis zwanzig ist. Bereits editierte Namen werden nur zurückgesetzt, wenn die Größe der Liste auf einen Wert gesetzt wird, der kleiner als die Position des editierten Namen ist. Unter dem Textfeld für die Anzahl der Teilnehmer ist ein Fragezeichen-Symbol. Beim Klick auf dieses Symbol poppt eine Karte auf, in der die einzelnen Elemente der InitView und ihre Grenzwerte erklärt werden. Wenn ein neues Meeting kodiert werden soll, dann kommt man über den "Weiter"-Button, in der unteren rechten Ecke, zur CounterView. Wenn jedoch ein altes Meeting nochmal visualisiert werden soll, dann muss das Upload-Symbol in der oberen rechten Ecke geklickt werden. Dies führt den Nutzer zur UploadView weiter.

4.3.2 CounterView

In der CounterView (Abbildung 4.3) findet die Kodierung eines neuen Meetings statt. Diese Ansicht sollte das Layout des Mockups umsetzen (siehe Abbildung 3.1), das von den Kunden gewünscht wurde.

Am oberen Rand ist der Meetingname dargestellt, der vorher in der InitView eingestellt wurde. Auf der linken Seite ist die Liste mit den Teilnehmern (hier mit 20 Teilnehmern, also der maximalen Anzahl) und rechts davon das 3x3 Raster mit den Buttons für die Aktionen zu sehen.

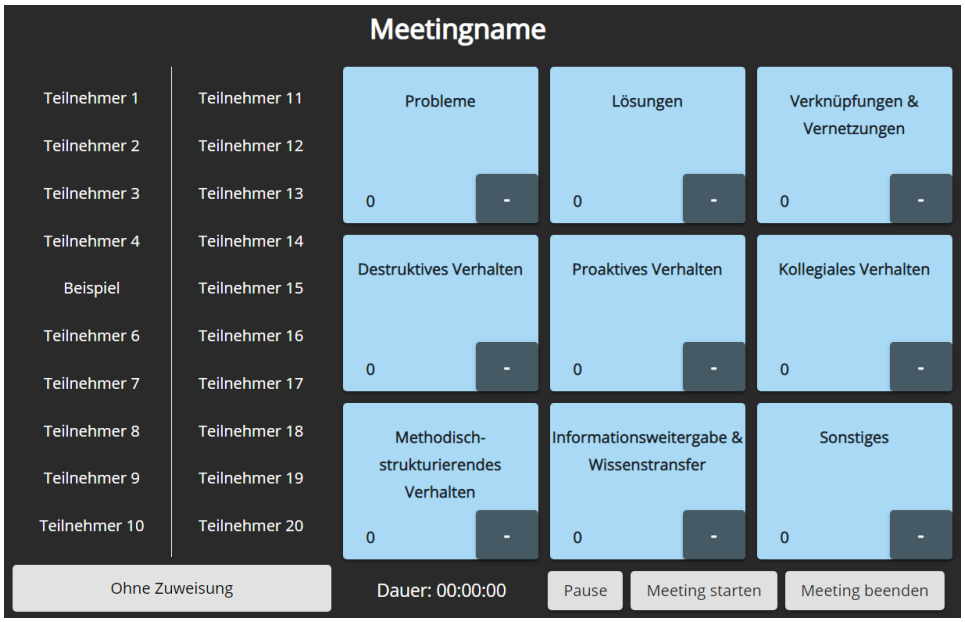

Abbildung 4.3: Screenshot der CounterView

Unter dem 3x3-Raster befindet sich die Stoppuhr für das Meeting.

Die Liste passt ihre Größe an die Anzahl der Teilnehmer aus der InitView an und erweitert diese immer um den Eintrag “Ohne Zuweisung”, auch, wenn null Teilnehmer ausgewählt werden. Dieser Eintrag wird immer in der unteren linken Ecke angezeigt. Wenn einem Teilnehmer eine Aktion zugewiesen wurde, dann ist danach immer der Eintrag “Ohne Zuweisung” ausgewählt. Diese Funktion wurde explizit vom Kunden gewünscht (**R03**). Die Namen der Teilnehmer passen sich an die geänderten Namen (aus der InitiView) an, als Beispiel wurde hier der Name des fünften Teilnehmers in den Namen “Beispiel” geändert.

Die Buttons für die Aktionen funktionieren exakt so, wie in der Java-Applikation (siehe 3.1.4), das heißt die große Fläche zählt den Zähler (jeweils in der unteren linken Ecke des Buttons) hoch und der Minus-Button (jeweils in der unteren rechten Ecke des Buttons) zählt herunter.

Die Stoppuhr besteht aus der Anzeige für die Uhr im Format hh:mm:ss, einem “Pause”-Button, einem “Meeting starten”-Button und einem “Meeting beenden”-Button. Die Funktionsweisen des “Pause”-Buttons ist selbsterklärend. Der Button “Meeting starten” ändert seine Aufschrift in “Meeting fortsetzen”, sobald er einmal betätigt wurde und ab diesem Zeitpunkt zählt die Uhr hoch. Solange dieser Button nicht betätigt wurde, ist es nicht möglich eine Aktion zu zählen. Wenn es doch versucht wird, weist eine Fehlermeldung den Nutzer darauf hin, indem sie den Nutzer darum bittet das Meeting zu starten, damit mit dem Zählen begonnen werden kann. Wenn man auf den

"Meeting beenden"-Button drückt, kommt man zur DataView.

Wenn in dieser Ansicht versucht wird die Seite zu schließen oder neu zu laden, indem die F5-Taste oder der Reload-Button des Browsers betätigt wird, dann weist eine Meldung den Nutzer darauf hin, dass beim Verlassen dieser Seite, die Daten des Meetings verloren gehen.

4.3.3 UploadView

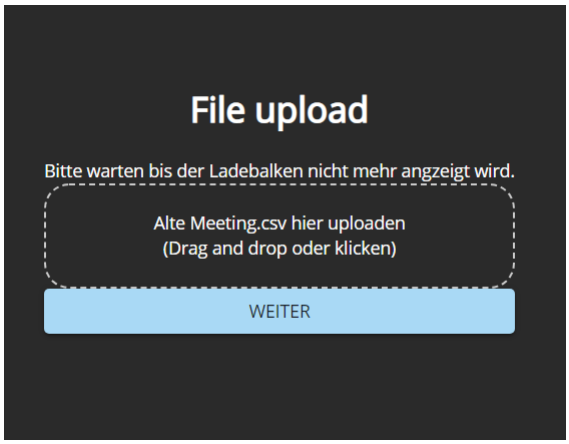

Abbildung 4.4: Screenshot der UploadView

In der UploadView (Abbildung 4.4) können Nutzer alte CSV-Dateien hochladen. Hierfür wird die Click and Drag Komponente des Packages react-papaparse verwendet. Wenn auf das gestrichelte Feld geklickt wird, dann öffnet sich ein Dateibrowser, in dem die CSV-Datei ausgewählt werden muss. Bei Drag and Drop wird die Datei direkt hochgeladen. Sobald der Ladebalken abgelaufen ist, kann man über den "Weiter"-Button zur DataView gelangen.

4.3.4 DataView

Die DataView ist die Ansicht, in der die kodierten Daten des Meetings ausgewertet werden können. Die Diagramme sind alle Teil des Packages nivo. Abbildung 4.4 zeigt die DataView, wenn der Tab für Personen ausgewählt ist. Ganz oben ist der Name des Meetings abgebildet. Darunter sieht man eine Tab-Leiste mit den vier Aufschriften "Aktionen", "Personen", "Aktionen und Personen" und "Zeit". Rechts daneben befindet sich ein Download-Button. Unter der Tab-Leiste sieht man eine Dropdown-Liste mit der Aufschrift "Alle Teilnehmer" und rechts daneben einen Symbol-Button der aus einem Symbol für ein Säulendiagramm und einem Symbol für ein Kreisdiagramm besteht. Unter diesen Beiden Komponenten ist ein Kreisdiagramm, mit einer Legende

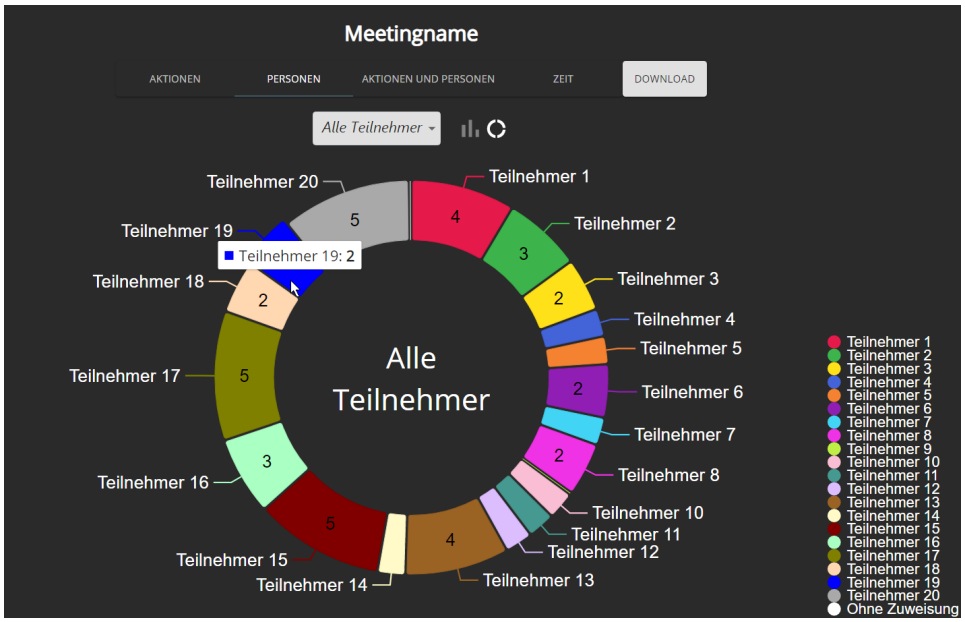

Abbildung 4.5: Screenshot der DataView mit den Personen als Fokus

rechts daneben, zu sehen. Die Maus hovert über dem Teilnehmer 19, wodurch ein Tooltip über der Maus sichtbar wird.

Der Download-Button dient zum Export der Daten des Meetings als CSV-Datei. Dafür wurde das Package Export-to-CSV verwendet. In den CSV-Dateien dieser Anwendung werden alle gezählten Aktionen des Meetings mit Reihenfolge (in Form einer ID), Zeitstempel (in Sekunden) und Person abgespeichert. Zusätzlich werden auch der Meetingname, das Datum des Meetings, Anfangs- und Endzeit des Meetings und eine Liste der Teilnehmer mit angegeben. Die Liste der Teilnehmer ist relevant, da es auch Meetings geben könnte, in denen Teilnehmer nichts sagen und diese würden ohne diese Liste nicht dokumentiert werden. Als Separator für die Einträge werden Semikolons verwendet. Der Name der exportierten Datei ist der Meetingname und kann ohne Probleme verändert werden, der Rest der Datei sollte jedoch möglichst nicht modifiziert werden, falls das Meeting nochmal visualisiert werden soll, da es sonst zu Fehlern kommen könnte.

Die Tab-Leiste dient zur Navigation zwischen den vier verschiedenen Arten der Datenauswertung, welche jeweils ihre eigenen Arten von Diagrammen zeigen. In dem Tab “Personen” (im Screenshot zu sehen) werden sowohl Kreisdiagramme als auch Säulen-/Stapeldiagramme verwendet. Das Kreisdiagramm hier zeigt die Summe der Aktionen eines Teilnehmers in dem kodierten Meeting. Das Kreisdiagramm wird als Ring-/Donut-Variante verwendet, da das Verhältnis trotzdem gut erkennbar bleibt und man zusätzlich die Mitte als Anzeige für den derzeitigen Fokus des Diagramms

nutzen kann. Das Säulendiagramm hat auf der x-Achse die Personen und auf der y-Achse Zahlen (von null bis zur größten Summe der Aktionen von allen Teilnehmern) als Beschriftung. Um zwischen den Diagrammarten zu wechseln, muss der Symbol-Button gedrückt werden. Wenn der Fokus des Diagramms gewechselt werden soll, dann kann entweder die Person in der Dropdown-Liste ausgewählt werden oder die Person direkt im Diagramm angeklickt werden. Das funktioniert bei beiden Diagrammarten und das Diagramm zeigt dann nur noch die Daten für die ausgewählte Person an, aufgefächert nach den Kategorien von act4teams-SHORT.

In dem Tab "Aktionen" wird nur ein Kreisdiagramm (als Ring-/Donut-Variante) verwendet. Hier wird per Default das Verhältnis der Summen der jeweiligen Kategorien angezeigt, also wie oft eine Kategorie in einem Meeting insgesamt vorkam. Auch hier kann, über eine Dropdown-Liste oder das Anklicken der jeweiligen Kategorie, der Fokus des Diagramms gewechselt werden. Das Kreisdiagramm zeigt dann das Verhältnis aller Personen an der jeweiligen Kategorie an.

Der Tab "Aktionen und Personen" zeigt ein Blasendiagramm (bubble chart) an. Auf der x-Achse stehen die Namen der Teilnehemr und auf der y-Achse die Kategorien. Der Wert der Blasen hängt von der Anzahl der Aktionen (der jeweiligen Kategorie) des Teilnehmers ab und kann über den Tooltip herausgefunden werden. Die Blasen skalieren automatisch im Verhältnis zueinander und zeigen einen guten Gesamtüberblick.

In dem Tab "Zeit" wird ein Zeitstrahl der Aktionen der Teilnehmer angezeigt. Auf der x-Achse steht die Zeit von Anfang bis Ende des Meetings (in Sekunden) und auf der y-Achse die Kategorien. In der Legende des Diagramms steht, welche Farbe für welchen Teilnehmer steht, aber das kann auch über den Tooltip herausgefunden werden.

4.3.5 Bewertung der Usability

Hier soll die Usability des User Interfaces mit Hilfe der Nielsen Heuristik abschließend beurteilt werden.

- **Visibility of system status:** Es gibt in jeder Ansicht klare Indikatoren dafür, welche Komponente derzeit aktiv ist, beispielsweise durch farbliche Hervorhebung in der InitView oder einen Rahmen für den ausgewählten Teilnehmer in der CounterView.

- **Match between system and the real world:** Dieser Punkt ist bei der Bewertung der Mockups schon aufgekommen. Die umgesetzte Kodierungsansicht nutzt eine Liste für die Teilnehmer, die ihre eigenen Vorteile bietet. Jedoch gibt es eben keine direkte Verbindung zwischen der Liste der Teilnehmer und der realen Welt, weshalb der Kodierer sich seine eigene logische Verbindung erstellen muss.

- **User control and freedom:** Die Freiheit für das Wechseln zwischen den Ansichten ist in eine Richtung beschränkt, um Fehler vorzubeugen. Es ist zwar immer möglich zu der InitView zurückzukehren, durch die F5-Taste oder den Reload-Button des Browsers, dabei gehen jedoch die Daten verloren, wenn sie nicht vorher heruntergeladen wurden. Dafür wurde das Setup der Anwendung so konzipiert, dass man schnell wieder zur Kodierungsansicht vorgehen kann. Undo wird durch den Minus-Button in der Kodierungsansicht unterstützt.

- **Consistency and standards:** In der CounterView ist die Anordnung der Liste auf der linken Seite und dem zentralen Body, der hier durch die Buttons der Aktionen repräsentiert wird, ein typisches Layout von Webseiten. Es wurde außerdem darauf geachtet, dass Elemente mit den gleichen Funktionen gleich gefärbt sind und ungefähr an den gleichen Positionen bleiben. Beispiele dafür sind die Positionen der Buttons für den Wechsel der Ansichten oder auch die Positionen der Listen zwischen der InitView und der CounterView.

- **Error prevention:** Es ist schwierig in dieser Webanwendung Fehler zu verursachen, da dem Nutzer viele Grenzen gesetzt wurden, um Fehler nicht zu ermöglichen. Ein gutes Beispiel hierfür ist, dass das Aktualisieren der Seite in der Kodierungsansicht erst noch bestätigt werden muss, um den Verlust von Daten zu verhindern.

- **Recognition rather than recall:** Das User Interface ist ziemlich flach, das heißt es gibt kaum versteckte Funktionen und für den Nutzer sollten alle seine Optionen erkennbar sein. Die einzige Einschränkung ist der Wechsel des Fokus durch das Anklicken der Diagramme. Jedoch wurde diese Funktion deshalb durch die Dropdown-Listen ergänzt.

- **Flexibility and efficiency of use:** Der Wechsel des Fokus der Diagramme durch das Anklicken ist die einzige Abkürzung für erfahrene Nutzer.

- **Aesthetic and minimalist design:** Mit der Wahl der Farben wurde versucht ein ästhetisches User Interface zu erstellen, jedoch ist es schwierig dies objektiv zu bewerten. Es wurde auch minimalistisch gehalten.

- **Help users recognize, diagnose, and recover from errors:** Die Fehlernachrichten sind immer eindeutig und in einfacher Sprache geschrieben.

- **Help and documentation:** Mit dem Fragezeichen-Button in der InitView, der alle Funktionen erklärt, wird versucht neuen Nutzern zu Helfen, sich mit dem User Interface zurechtzufinden.

Kapitel 5

Verwandte Arbeiten

Die hier entwickelte Webanwendung dient zur manuellen Echtzeitkodierung mit act4teams-SHORT. Jedoch wird derzeit auch an Lösungen geforscht, um solche Kodierungen zu automatisieren.

In seiner Masterarbeit entwickelte Horstmann [15] ein Konzept für die Analyse der textuellen digitalen Kommunikation von Softwareentwicklerteams, außerhalb von Meetings, und konnte es erfolgreich an einem Team von 80 Personen anwenden. Mit Hilfe von Natural Language Processing werden Textnachrichten analysiert und in drei Kategorien eingeteilt: Positiv, Neutral und Negativ. Das Konzept wurde mit einer manuellen Klassifizierung durch eine Person verglichen und war nur 4% schlechter als diese, hatte jedoch nur eine Genauigkeit von 63%.

Aufbauend auf der Arbeit von Horstmann entwickelte Meyer[26] in seiner Bachelorarbeit eine Verbesserung der Klassifizierungsgenauigkeit für dieses Konzept. Das Konzept von Horstmann wurde an einem Goldstandard-Datensatz getestet, an dem es eine Genauigkeit von 82% erzielen konnte. Anhand dieses Tests wurden verschiedene Anpassungen für die Vorverarbeitung der Daten, die Klassifizierungsalgorithmen und die Merkmalsselektion entwickelt. Dadurch konnte das Konzept an dem gleichen Goldstandard-Datensatz eine Genauigkeit von 87% bei der Klassifizierung von Nachrichten erreichen.

Aufbauend auf diesen zwei Arbeiten wurde von Herrmann[13] eine Möglichkeit entwickelt, diesen Klassifikationsalgorithmus auf gesprochene Sprache statt auf Text anzuwenden. Dafür wird der Ton von Meetings aufgezeichnet, mit einer Sprachengine in Text umgewandelt und an diesem Text der Klassifikationsalgorithmus von Horstmann und Meyer angewandt. Bei einem Vergleich mit einer manuellen Kodierung, konnte diese Verfahren eine Übereinstimmung von 88% erreichen. Hermann gibt auch den Ausblick, dass der nächste Schritt dieses Klassifikationsalgorithmus, die automatische Kodierung von act4teams-SHORT wäre.

Kapitel 6

Zusammenfassung und Ausblick

In diesem Kapitel wird die Arbeit nochmal zusammengefasst. Außerdem wird eine mögliche Erweiterung der hier entwickelten Webanwendung vorgestellt.

6.1 Zusammenfassung

Die Entwicklung einer Webanwendung für das Kodierungsschema act4teams-SHORT war das Ziel dieser Arbeit. Die Grundlage für die hier entwickelte Webanwendung bildete ein Java-Prototyp, der zuvor für die Kodierung mit act4teams-SHORT genutzt wurde. Dieser Prototyp konnte, bei der Kodierung von Meetings, nur Aktionen einer Kategorie und ihre Zeitstempel festhalten. Die Kodierung sollte, in dieser Arbeit, zusätzlich um den Namen der Teilnehmer erweitert werden. Außerdem sollte der Export der kodierten Daten als CSV-Datei ermöglicht werden.

Anhand von Mockups der Kodierungsansicht, wurden verschiedene Möglichkeiten der Anordnung der Teilnehmer diskutiert. Die Nutzbarkeit mit Tabletcomputern war ein wichtiger Faktor, auf den bei der Bewertung der Mockups geachtet wurde. In einem Anforderungsgespräch wurden die Mockups vorgestellt und mit den Kunden dieser Anwendung ein Layout festgelegt. Das festgelegte Layout besteht aus einer Liste für die Teilnehmer eines Meetings und einem 3x3-Raster für die Buttons der Kategorien von act4teams-SHORT. Das Layout der Buttons für die Aktionen eignet sich gut, da das 3x3-Raster eine bekannte Anordnung aus dem Alltag ist, neue Nutzer es also schnell erlernen können und erfahrene Nutzer es zusätzlich vom Java-Prototypen kennen. Eine Liste als Anordnung für die Teilnehmer ist sinnvoll, da Listen übersichtlich gestaltet werden können und das Layout insgesamt die beidhändige Bedienung vereinfacht. Von den Kunden wurden weitere Auswertungs- und Darstellungsmöglichkeiten der kodierten Meetings gewünscht, welche mit Hilfe von Mockups dargestellt wurden. Das erarbeitete

Konzept wurde mit dem Webframework React.js implementiert. React ermöglicht die Entwicklung von reaktionsschnellen Webanwendungen und ist gut erweiterbar. Bei der Implentierung des erarbeiteten Konzeptes wurde, außer auf Usability, auch auf Portability geachtet, damit die Webanwendung sowohl lokal als auch auf einem Server gehostet werden kann. Mit Hilfe eines Packages wurde der Export der Meetingdaten als CSV-Datei ermöglicht.

6.2 Ausblick

Das User Interface dieser Webanwendung ist vor allem auf die Nutzung mit Tabletcomputern ausgelegt und ist auch auf Desktop-Computern und Laptops nutzbar. Auf Smartphones ist es nicht nutzbar, da diese einen kleineren Bildschirm haben und deshalb ihr eigenes Layout benötigen. Ein Layout für Smartphones wäre sinnvoll, da wahrscheinlich die meisten Softwareentwickler ein Smartphone täglich mit sich tragen und act4teams-SHORT gerade darauf ausgelegt ist, leichter zugänglich zu sein. Ein Beispiel für ein mögliches Layout der Kodierungsansicht ist in Abbildung 6.1 zu sehen. Links ist ganz oben die Liste der Teilnehmer (in lila), ausgelegt als runde Buttons in einem Raster, darunter das 3x3-Raster der Buttons für die Aktionen und ganz unten die Stoppuhr. Rechts zeigt die Teilnehmer als aufgeklappte Dropdown-Liste.

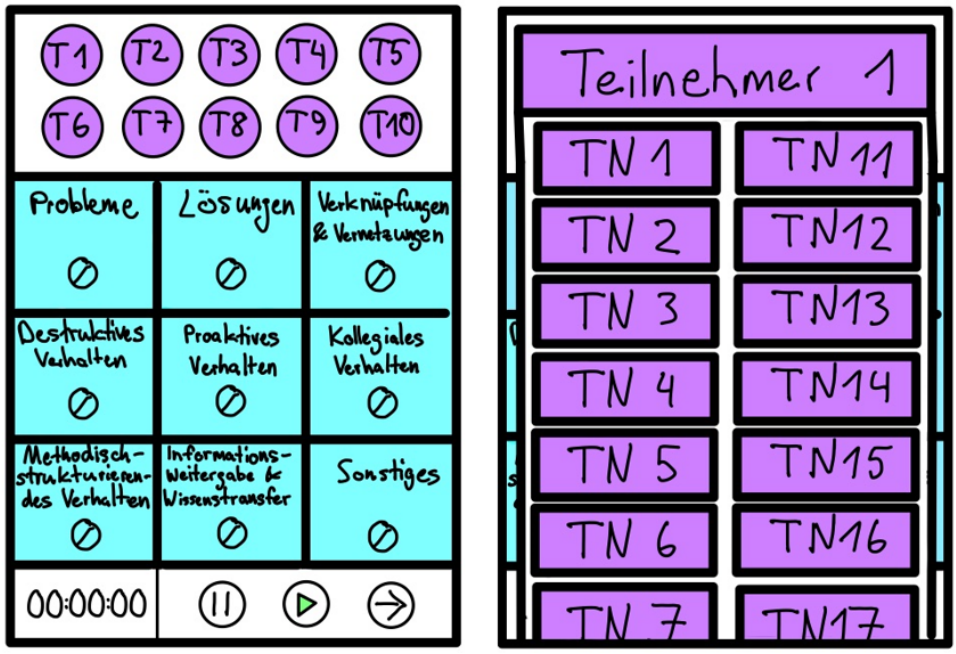

Abbildung 6.1: Mockups für Smartphone Layout

Die Liste der Teilnehmer kann nicht so dargestellt werden, wie auf einem

großen Bildschirm in horizontaler Ausrichtung, und muss deshalb verkleinert werden. Hier werden dafür zwei Möglichkeiten gezeigt, die man kombinieren könnte, je nach Anzahl der Teilnehmer in einem Meeting. Das linke Layout zeigt ein festes Raster aus runden Buttons für die Teilnehmer und kann für Meetings mit einer geringen Anzahl an Teilnehmern (bis zu zehn) genutzt werden. Das rechte Layout ersetzt die runden Buttons der Teilnehmer durch eine Dropdown-Liste. Ganz oben kann man sehen, dass genug Platz für das Ausschreiben von längeren Zeichenketten möglich wäre ("Teilnehmer 1"). Innerhalb der aufgeklappten Dropdown-Liste werden die Namen abgekürzt (Teilnehmer wird zu TN), damit man zwei Spalten von Teilnehmern anzeigen kann. TN7 und TN17 sind unten abgeschnitten, was zeigen soll, dass man in der Liste scrollen kann. Dieses Layout der Teilnehmer könnte für Meetings mit mehr als zehn Teilnehmern genutzt werden.

Auch die Buttons für die Aktionen müssten angepasst werden, da es nicht genügend Platz für den Minus-Button gäbe. Ein Button der sich anbieten würde und hier dargestellt ist, ist auch schon im Konzept vorgestellt worden (siehe Abbildung 3.6, Button $*1$). Der Button ist gut skalierbar und die Funktionalität des Minus-Buttons, kann durch ein langes Berühren ersetzt werden. Ein Problem gäbe es jedoch bei der Darstellung des Textes der Kategorien auf den Buttons. Da manche Kategorien einen sehr langen Namen haben, müsste man diese wahrscheinlich abkürzen. Man könnte die Namen auch herunterskalieren, aber dann könnten sie unlesbar werden (sieht man auch im Mockup).

Bei der Stoppuhr sollten, so wie hier dargestellt, statt einfachen Buttons mit Text, Symbol-Buttons verwendet werden, da diese weniger Platz einnehmen und die Funktionalität der Buttons trotzdem erkennbar bleibt.

Mit dem hier gewählten Webframework React ist so eine Erweiterung der Webanwendung gut umsetzbar. Es könnte jedoch Probleme bei der Auswertung und Visualisierung der Daten geben, wenn es in den Meetingräumen keine Beamer oder Fernseher mit Smartphone-Kompatibilität gibt.

Literaturverzeichnis

[1] alexcaza. export-to-csv - GitHub. https://github.com/alexcaza/export-to-csv. letzer Zugriff: August 2021.

[2] C. M. Brown. *Human-computer interface design guidelines.* Intellect Books, 1999.

[3] R. Budiu. Dark Mode vs. Light Mode: Which Is Better? https://www.nngroup.com/articles/dark-mode/. letzer Zugriff: August 2021.

[4] Bunlong. react-papaparse - GitHub. https://github.com/Bunlong/react-papaparse. letzer Zugriff: August 2021.

[5] P. Dash and Y. C. Hu. How much battery does dark mode save? an accurate oled display power profiler for modern smartphones. In *Proceedings of the 19th Annual International Conference on Mobile Systems, Applications, and Services*, pages 323–335, 2021.

[6] J. Dobres, N. Chahine, and B. Reimer. Effects of ambient illumination, contrast polarity, and letter size on text legibility under glance-like reading. *Applied Ergonomics*, 60:68–73, 2017.

[7] Facebook. Yarn. https://yarnpkg.com/. letzer Zugriff: August 2021.

[8] Facebook Open Source. Context - React. https://reactjs.org/docs/context.html. letzer Zugriff: August 2021.

[9] Facebook Open Source. Getting Started - React. https://reactjs.org/docs/getting-started.html. letzer Zugriff: August 2021.

[10] Facebook Open Source. Glossary of React Terms - React. https://reactjs.org/docs/glossary.html#single-page-application. letzer Zugriff: August 2021.

[11] Facebook Open Source. React | Facebook Open Source. https://opensource.fb.com/projects/react. letzer Zugriff: August 2021.

[12] Google. Introduction - Material Design. https://material.io/design/introduction. letzer Zugriff: August 2021.

[13] M. Herrmann. Automatische Klassifikation von Aussagen in Meetings von Entwicklungsteams. `https://www.pi.uni-hannover.de/fileadmin/pi/se/Stud-Arbeiten/2021/Herrmann2021.pdf`, 2021. Bachelor's thesis, Leibniz Universität Hannover.

[14] B. Hey. *Präsentieren in Wissenschaft und Forschung.* Springer, 2018.

[15] J. Horstmann. Computer-gestützte Analyse des Kommunikationsverhaltens in Entwicklerteams unter Berücksichtigung digitaler Medien. Master's thesis, Leibniz Universität Hannover, 2019.

[16] M. N. Humbad, M. B. Donnellan, K. L. Klump, and S. A. Burt. Development of the brief romantic relationship interaction coding scheme (brrics). *Journal of Family Psychology*, 25(5):759, 2011.

[17] B. Jordan and A. Henderson. Interaction analysis: Foundations and practice. *The Journal of the Learning Sciences*, 4(1):39–103, 1995.

[18] S. Kauffeld. Kauffeld: Arbeits-, Organisations- und Personalpsychologie für Bachelor. `https://lehrbuch-psychologie.springer.com/sites/default/files/atoms/files/web-exkurs.007.04.pdf`. letzer Zugriff: August 2021.

[19] S. Kauffeld and N. Lehmann-Willenbrock. Meetings matter: Effects of team meetings on team and organizational success. *Small group research*, 43(2):130–158, 2012.

[20] S. Kauffeld, N. Lehmann-Willenbrock, and A. Meinecke. *The Advanced Interaction Analysis for Teams (act4teams) coding scheme*, chapter 21. Cambridge University Press, 06 2018.

[21] S. Kauffeld, G. Tiscar-Lorenzo, K. Montasem, and N. Lehmann-Willenbrock. act4teams®: Die nächste generation der teamentwicklung. *Handbuch kompetenzentwicklung*, pages 191–215, 2009.

[22] J. Klünder. Vorlesungsfolien: Sozio-technische Aspekte des Software Engineering, 2020/2021.

[23] J. Klünder, N. Prenner, A.-K. Windmann, M. Stess, M. Nolting, F. Kortum, L. Handke, K. Schneider, and S. Kauffeld. Do You Just Discuss or Do You Solve? Meeting Analysis in a Software Project at Early Stages. In *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops*, ICSEW'20, page 557–562, New York, NY, USA, 2020. Association for Computing Machinery.

[24] E. Krupat, R. Frankel, T. Stein, and J. Irish. The four habits coding scheme: validation of an instrument to assess clinicians' communication behavior. *Patient education and counseling*, 62(1):38–45, 2006.

[25] Material-UI. About Us - Material-UI. `https://material-ui.com/company/about/`. letzer Zugriff: August 2021.

[26] C. Meyer. Verbesserung von evolutionären Algorithmen zur Klassifikation von schriftlicher Kommunikation in Entwicklungsteams. `https://www.pi.uni-hannover.de/fileadmin/pi/se/Stud-Arbeiten/2020/Meyer2020.pdf`, 2020. Bachelor's thesis, Leibniz Universität Hannover.

[27] Microsoft. What is TypeScript? `https://www.typescriptlang.org/`. letzer Zugriff: August 2021.

[28] Mozilla Developer Network. Getting started with the web. `https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web`, 2021. letzter Zugriff: August 2021.

[29] J. Nielsen. Ten usability heuristics. `https://www.nngroup.com/articles/ten-usability-heuristics/`, 1994. letzer Zugriff: August 2021.

[30] J. Nielsen and R. Molich. Heuristic evaluation of user interfaces. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 249–256, 1990.

[31] M. Pekarsky. Does your web app need a front-end framework? `https://stackoverflow.blog/2020/02/03/is-it-time-for-a-front-end-framework/`, 2020. letzer Zugriff: August 2021.

[32] pluoc. nivo - GitHub. `https://github.com/plouc/nivo`. letzer Zugriff: August 2021.

[33] N. Prenner, J. Klünder, and K. Schneider. Making meeting success measurable by participants' feedback. *SEmotion '18: Proceedings of the 3rd International Workshop on Emotion Awareness in Software Engineering*, page 25–31, 2018.

[34] M. Schween and J. Klünder. Eye-Tracking-Studie zur Verwendung von Farben und zur Anordnung von Kategorien bei act4teams-SHORT. unpublished, 2019.

[35] Stack Overflow. Developer Survey 2021. `https://insights.stackoverflow.com/survey/2021#web-frameworks`, 2021. letzer Zugriff: August 2021.

[36] T. Steiner. Let there be darkness! Maybe.... `https://medium.com/dev-channel/let-there-be-darkness-maybe-9facd9c3023d`. letzer Zugriff: August 2021.

[37] J. Westenberg. We asked, you told us: Just about everyone uses dark mode. `https://www.androidauthority.com/dark-mode-poll-results-1090716/`. letzer Zugriff: August 2021.

[38] S. Zuffi, C. Brambilla, G. Beretta, and P. L. Scala. Human computer interaction: Legibility and contrast. In *Proceedings - 14th International conference on Image Analysis and Processing, ICIAP 2007*, pages 241–246, 10 2007.