TECHNISCHE UNIVERSITÄT DRESDEN

Fakultät Informatik Institut für Systemarchitektur, Professur für Rechnernetze

Großer Beleg

Umsetzbarkeitsuntersuchungen zu Zeit-Metaphern und Nutzungskonzepten in Graphicuss

Matti Leydecker
Geboren am: 27.09.1990 in Berlin
Matrikelnummer: 3824144

Betreuer
Dr. Tenshi Hara
Betreuender Hochschullehrer
Herr Prof. Dr. rer. nat. habil. Dr. h. c. Alexander Schill

Eingereicht am: 27.04.2018

TECHNISCHE UNIVERSITÄT DRESDEN

Fakultät Informatik Institut für Systemarchitektur, Professur für Rechnernetze

Zusammenfassung

Graphicuss ist eine Mischung aus virtuellem Whiteboard und Forensystem, welches an der TU Dresden, Professur für Rechnernetze entwickelt wurde. In dieser Arbeit wird der bestehende Prototyp von Graphicuss analysiert und neue Anforderungen werden definiert. Unter Nutzung der Ergebnisse der Arbeit von Natalia Iljassova wird ein neuer Prototyp des Frontends vorgestellt. Der Entwicklungsprozess wird dabei anschaulich dargestellt und dokumentiert. Anschließend wird in einer Nutzerstudie die Bedienbarkeit des Prototyps bewertet. Es folgt ein Ausblick über zukünftige Eigenschaften und Merkmale von Graphicuss.

Selbstständigkeitserklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit mit dem Titel *Umsetzbarkeitsuntersuchungen zu Zeit-Metaphern und Nutzungskonzepten in Graphicuss* selbstständig und ohne unzulässige Hilfe Dritter verfasst habe. Es wurden keine anderen als die in der Arbeit angegebenen Hilfsmittel und Quellen benutzt. Die wörtlichen und sinngemäß übernommenen Zitate habe ich als solche kenntlich gemacht. Es waren keine weiteren Personen an der geistigen Herstellung der vorliegenden Arbeit beteiligt. Mir ist bekannt, dass die Nichteinhaltung dieser Erklärung zum nachträglichen Entzug des Hochschulabschlusses führen kann.

Dresden, 27.04.2018

Matti Leydecker

TECHNISCHE UNIVERSITÄT DRESDEN

Fakultät Informatik Institut für Systemarchitektur Professur Rechnernetze

AUFGABENSTELLUNG FÜR DEN GROSSEN BELEG

THEMA: Umsetzbarkeitsuntersuchungen zu Zeit-Metaphern und Nutzungskonzepten in Graphicuss

Name:	Leydecker, Matti	Studiengang:	Diplom Informatik (PO 2010)
Matrikel-Nummer:		Projekt/Fokus:	AMCS/Graphicuss
verantwortlicher Hochschullehrer:	Prof. Dr. rer. nat. habil. Dr. h. c. Alexander Schill		
involvierte Mitarbeiter:	Dr.-Ing. Tenshi Hara, Dr.-Ing. Iris Braun		
Beginn am:		einreichen bis:	

ZIELSTELLUNG

Am Lehrstuhl Rechnernetze wurde mit Graphicuss eine Kombination aus klassischen Forensystemen und interaktiven, virtuellen Whiteboards entwickelt. Darauf basierend sind mehrere Abschlussarbeiten entstanden. Diese machen Vorschläge zu Nutzungskonzepten (bspw. Deanonymisierungskonzept) und zu Darstellungsmöglichkeiten durch Zeit-Metaphern. Eine Untersuchung am Prototypen erfolgte bisher nicht.

Da der grundlegende Prototyp von Graphicuss sich nicht gut mit dem ebenfalls am Lehrstuhl implementierten Audience Response System „Auditorium Mobile Classroom Service“ (AMCS) integriert, ist generell eine Neuimplementierung von Graphicuss auf Basis der neuen Erkenntnisse und dem aktuellen Stand der Technik wünschenswert. Zudem gibt es neuerliche Erkenntnisse aus dem Bereich der Lehr-/Lern-Psychologie, die vorschlagen die Zeit-Metaphern auf ein Zustandskonzept mit einer Phasen-Basis zu stellen.

Im Rahmen dieses Großen Beleges soll untersucht werden, ob und wie gut sich die in den bisherigen Abschlussarbeiten erarbeiteten Ergebnisse mit den aktuellen Web-Technologien implementieren lassen. Dabei sollen auch die neusten Vorschläge aus dem Bereich der Lehr-/Lern-Psychologie einfließen. Ebenfalls soll Graphicuss auf die gleiche Nutzerbasis wie AMCS zugreifen können (Single Sign-on). Das Implementationsergebnis ist im Anschluss gegen die gängigen technischen Metriken zu validieren, bevor eine Nutzer-basierte Evaluation, insbesondere des Workflows und der Verständlichkeit der Metaphern, erfolgt.

Prof. Dr. rer. nat. habil. Dr. h. c. Alexander Schill
(verantwortlicher Hochschullehrer)

SCHWERPUNKTE

- Analyse und Auswahl bestehender Ansätze und Lösungen,
- Definieren von Anforderungen,
- Definieren von Bewertungskriterien für die Anforderungserfüllung,
- prototypische Umsetzung des Konzepts,
- Evaluation und Auswertung der Ergebnisse.

Inhaltsverzeichnis

Zusammenfassung **2**

1. Einleitung **11**
1.1. Motivation . . . 11
1.2. Graphicuss . . . 11
1.3. Ziel dieser Arbeit . . . 11

2. Grundlagen und verwandte Arbeiten **13**
2.1. Forensysteme . . . 13
2.2. Virtuelle Interaktive Whiteboards (V-IWB) . . . 13
2.2.1. AwwApp . . . 14
2.3. Diskussionssysteme . . . 14
2.3.1. Auditorium Mobile Classroom (AMCS) . . . 14
2.3.2. auditorium . . . 15

3. Analyse des aktuellen Standes **16**
3.1. Graphicuss (Chen) . . . 16
3.1.1. Kursverwaltung (course management) . . . 16
3.1.2. Frageverwaltung (question management) . . . 16
3.1.3. Antwortverwaltung (answer management) . . . 16
3.1.4. Grafischer Editor (drawing tool) . . . 17
3.1.5. Echtzeitfähigkeit (realtime) . . . 17
3.1.6. Aktueller Stand des Antwort-Editors . . . 17
3.1.7. Zitierfunktion . . . 18
3.2. Analyse der Bedienungsmetaphern für den grafischen Editor . . . 18
3.2.1. Nutzungskontexte in Graphicuss . . . 19
3.2.2. Metaphern für den Aktionsverlauf . . . 19
3.2.3. Evaluation der Diagramme . . . 21

4. Anforderungsanalyse **23**
4.1. Funktionale Anforderungen . . . 23
4.1.1. Funktionale Anforderungen Graphicuss allgemein . . . 23
4.1.2. Funktionale Anforderungen für den grafischen Editor . . . 23
4.2. Nicht-funktionale Anforderungen . . . 23

5. Konzepte zur Erweiterung des bisherigen Standes **26**
5.1. Erweiterung der Historie . . . 26
5.1.1. Speichern der Historie an der Frage oder Antwort . . . 26

5.1.2. Zitieren beliebiger Zustände aus der Bildhistorie . . . 26
5.1.3. Thumbnails . . . 26
5.1.4. Löschen von Zuständen . . . 26
5.1.5. Animationen . . . 28
5.2. Phasenkonzept . . . 28
5.2.1. Markieren von Zuständen als Phasen . . . 28
5.2.2. Springen zwischen Phasen . . . 28

6. Implementierung **29**
6.1. Technologieauswahl . . . 29
6.2. Aktueller Stand Web Technologien . . . 29
6.3. Implementierung des Frontends . . . 29
6.3.1. Implementierung des grafischen Editors . . . 30
6.3.2. Implementierung des Phasenkonzeptes . . . 31
6.4. Anpassungen am Backend . . . 33
6.5. Technische Details des SSO . . . 35

7. Evaluation **37**
7.1. Usability . . . 37
7.1.1. System Usability Scale (SUS) . . . 37
7.1.2. Fragebogen zur Usability . . . 38
7.1.3. Think-Aloud-Methode . . . 38

8. Zusammenfassung und Ausblick **40**

A. Anhang **42**
A.1. Usability Fragebogen . . . 43
A.2. System Usability Scale . . . 44

Abbildungsverzeichnis

2.1. Die Werkzeugpalette in AwwApp 14

3.1. Der alte Texteditor in Graphicuss 17
3.2. Der alte grafische Editor in Graphicuss 18
3.3. Die von Iljassova gesammelten Diagramme zur Darstellung des zeitlichen Ablaufs. 20
3.4. Die Auswertung zu den UI Skizzen 22

4.1. Das Use-Case Diagram für Graphicuss 24
4.2. Use-Case Diagramm für den grafischen Editor 25

5.1. Die überarbeitete Historie in Graphicuss 27
5.2. Ein einzelner Zustand der Historie 28

6.1. Das Mockup für den Editor von Chen 30
6.2. Der fertige grafische Editor von Graphicuss 32
6.3. Das Datenmodell in Chens Graphicuss 34
6.4. Das erweiterte Datenmodell für die Neuimplementierung 35

Tabellenverzeichnis

3.1. Die Kandidatenmetaphern für Graphicuss 19
3.2. Die vordefinierten Fragen der Auswertung 21

4.1. Funktionale Anforderungen für Graphicuss 23
4.2. Anforderungen für den grafischen Editor 24

7.1. Die Ergebnisse der System Usability Scale 38
7.2. Die Auswertung der Interviewfragen 39

Quelltextverzeichnis

6.1. Das Datenmodell für den HistoryEntry 30
6.2. Die Funktion zur Bewegung innerhalb der Historie 33
6.3. Das neue Schema für einen Historieneintrag 34
6.4. Das modifizierte Schema für die Antwort 35
6.5. Eine beispielhafte Implementierung einer Registrierungsfunktion für Graphicuss 36

1. Einleitung

1.1. Motivation

Die Benutzung von Online Plattformen nimmt in verschiedensten Bereichen zu. Auch in der Lehre ist dieser Trend bemerkbar. Beispielsweise steigt die Anzahl unterschiedlicher für das Studium genutzter Plattformen, wie Portale zur Organisation der Lehre oder Antwortcommunities immer weiter an.

Gerade in der universitären Lehre treffen sehr unterschiedliche Lerntypen aufeinander. Manche Studenten beginnen bereits während des Semesters mit ausführlichen Vorbereitungen, andere warten bis kurz vor den Prüfungen und benötigen dann viele Informationen in kurzer Zeit. Um dennoch zwischen beiden Gruppe von Studenten einen Wissensaustausch zu gewährleisten, wird eine Plattform benötigt, über die asynchron - also zeitversetzt - kommuniziert werden kann.

Gerade im studentischen Kontext sind Diagramme und Abbildungen von besonderer Bedeutung. Es kann vorkommen, dass eine ganze Übungsstunde über eine besonders komplexe schematische Darstellung diskutiert wird. Im virtuellen Raum fehlen dafür einige Hilfsmittel. Es kann nicht auf die Grafik gezeigt werden und über einzelne, klar definierte Abschnitte diskutiert werden. Online erscheint eine Grafik im Ganzen, während offline die Erstellung mitverfolgt werden kann. An diesem Punkt wird ein Werkzeug benötigt, welches den Austausch unter Studenten ermöglicht und dabei die Einschränkungen der Onlinekommunikation verringert.

1.2. Graphicuss

Graphicuss ist ein grafisches Diskussionssystem, eine Zusammensetzung aus virtuellem Forum und grafischem Editor. Es wurde erstmals von Chen im Rahmen seiner Masterarbeit an der TU Dresden vorgestellt (Chen 2016). Studenten können Fragen stellen und Antworten geben. Dabei können sie Grafiken erstellen und anhängen. Das System stellt Werkzeuge zur Selbstregulierung wie ein Voting-System und das Markieren nützlicher Antworten zur Verfügung.

1.3. Ziel dieser Arbeit

In dieser Arbeit wird eine Neuauflage von Graphicuss entworfen, dokumentiert und implementiert. In der neuen Version sollen die Ergebnisse aktueller Abschlussarbeiten sowie

neue Erkenntnisse aus der Lehr-/Lernpsychologie einfließen. Zusätzlich soll eine technologische Anpassung an das ebenfalls an der TU Dresden entwickelte AMCS Tool vorgenommen werden. Dieses wird in Unterabschnitt 2.3.1 näher beschrieben.

2. Grundlagen und verwandte Arbeiten

In diesem Kapitel soll auf ausgewählte Grundlagen eingegangen sowie der aktuelle Stand verwandter Webanwendungen untersucht werden.

2.1. Forensysteme

Internet Forensysteme dienen dem Informationsaustausch verschiedener Nutzer. Häufig sind die Foren einem Überthema zugeordnet. Innerhalb des Forums können Nutzer Beiträge zu Themen posten oder eigene Themen erstellen. Die Kommunikation in Foren ist asynchron, d.h. Nutzer lesen oder posten Beiträge nicht zwingend sofort, sondern können dies in ihrem eigenen Rhythmus ausführen.

2.2. Virtuelle Interaktive Whiteboards (V-IWB)

Hara beschreibt virtuelle interaktive Whiteboards (V-IWB) als interaktive Whiteboards, welche sich eines virtuellen Displays statt eines physischen Displays bedienen (T. Hara 2016). Dabei wird zwischen Arten der Kollaboration $m : n$ unterschieden. m ist die Anzahl der Ersteller des Inhalts und n die Anzahl der nur lesenden Konsumenten.

1. Sei $m = 1$ und $n = 0$, dann existiert ein Ersteller von Inhalt der diesen nur mit sich selbst teilt. Das Whiteboard dient dann eher als persönliches Notizbuch.
2. Sei $m = 1$ und $n \in \mathbb{N}_{\neq 0}$, dann teilt der Ersteller den Inhalt mit mindestens einer weiteren Person. Das V-IWB fungiert als persönlicher Blog oder schwarzes Brett.
3. Sei $m \in \mathbb{N}_{>1}$ und $n = 0$, dann arbeiten mehrere Personen parallel oder seriell an Inhalt den sie nur unter sich teilen. Das Whiteboard funktioniert als geteiltes Notizbuch.
4. Sei $m \in \mathbb{N}_{>1}$ und $n \in \mathbb{N}_{\neq 0}$, dann arbeiten mehrere Personen parallel oder seriell und teilen den Inhalt mit mindestens einer Person. Das Whiteboard fungiert als ein geteiltes schwarzes Brett.

Bei Graphicuss trifft für die initiale Erstellung einer Grafik der zweite Fall mit einem Ersteller und potenziell mehreren Empfängern zu. Allerdings ist es jedem Empfänger einer Grafik möglich, diese in seine Zeichenfläche zu kopieren und modifiziert zurückzusenden.

Im Verlauf der Nutzung wird Graphicuss zu einer Art schwarzem Brett. Fragen und Antworten können beliebig oft zitiert und dabei modifiziert werden. Der Unterschied zu einem schwarzen Brett ist, dass Nutzer nicht exakt den selben Inhalt bearbeiten, sondern jeweils an ihrer Kopie.

2.2.1. AwwApp

AwwApp[1] ist ein visuelles Online-Tool zur Kollaboration mehrerer Nutzer. Es lassen sich Whiteboards anlegen, welche über einen Link zur Verfügung stehen und auf mehreren Geräten angezeigt werden können. Desktop sowie mobile Clients werden unterstützt. Auf dem Whiteboard gibt es die für Bildbearbeitung üblichen Werkzeuge wie Auswahlwerkzeug, Rückgängig machen, Farbauswahl, Freihandzeichnen, Radiergummi und geometrische Formen (siehe Abbildung 2.1).

Abbildung 2.1.: Die Werkzeugpalette in AwwApp

Um andere Nutzer zur Kollaboration einzuladen, muss nur der Link zum entsprechenden Whiteboard geteilt werden. Dadurch ist es auch möglich, mehrere Geräte zur Bearbeitung zu verwenden. So kann beispielsweise ein Tablet zur Eingabe und ein Laptop mit Beamer zum Anzeigen des Whiteboards verwendet werden. Für die Kommunikation der Nutzer ist ein einfacher Chatroom in AwwApp integriert. Sendet ein Nutzer eine Nachricht über das Eingabefeld, wird diese an alle anderen Nutzer, welche das Whiteboard sehen, weitergeleitet.

AwwApp eignet sich, um mit wenig Aufwand online Skizzen kollaborativ zu erstellen und zu teilen.

2.3. Diskussionssysteme

2.3.1. Auditorium Mobile Classroom (AMCS)

AMCS ist eine Softwarelösung des Lehrstuhls Rechnernetze der TU Dresden. Es kann zur Unterstützung von Lehrveranstaltungen eingesetzt werden um die Kommunikation zwischen Lehrenden und Lernenden zu verbessern (Kubica, I. T. Hara und I. I. Braun 2016). Dazu können vom Lehrenden Fragen definiert werden, welche die Lernenden beantworten können. In AMCS werden einzelne Lehrveranstaltungen zu Kursen zusammengefasst. Ein Kurs kann zum Beispiel eine Vorlesungsreihe über ein Semester sein. Eine einzelne Vorlesung ist dann eine Lehrveranstaltung. Über mobile Endgeräte können sich die Lernenden während der Lehrveranstaltung in einen Kurs einschreiben. Hierfür benötigen sie eine PIN, welche der Lehrende für den Kurs definiert.

Lernende können:

- Fragen beantworten
- Nachrichten erhalten
- Ihr Profil bearbeiten

Lehrende können:

[1] https://awwapp.com/, zuletzt abgerufen am 19.04.2018

- Kurse und Fragen erstellen und auswerten
- Den Zustand (vor, nach oder während der Veranstaltung) und die aktive Folie der Veranstaltung setzen
- Nachrichten definieren
- Ergebnisse des Kurses / der Lehrveranstaltung exportieren
- Assistenten (besitzen die selben Berechtigungen wie der Lehrende) verwalten
- Ihr Profil bearbeiten

AMCS unterstützt den Lernprozess durch weitere Merkmale wie das Abfragen persönlicher Ziele, interaktive Lernaufgaben und das Bereitstellen von weiterführendem Material (I. Braun u. a. 2015).

2.3.2. auditorium

Bei auditorium handelt es sich um ein klassisches Internetforum, welches intern für die Fakultät Informatik der TU Dresden verwendet wird (Beier, I. Braun und T. Hara 2014). Nach einer Registrierung hat der Nutzer die Möglichkeit, Gruppen zu folgen oder selbst eine Gruppe zu erstellen. Diese Gruppen können Lehrveranstaltungen, Themengruppen oder Studiengruppen repräsentieren. Innerhalb einer Gruppe können Ankündigungen versendet werden, diese werden an alle Nutzer weitergeleitet die dieser Gruppe folgen. Zusätzlich können Fragen gestellt werden oder Themen erstellt werden. Hierbei gibt es die Möglichkeit private Fragen zu stellen, welche nur von den Gruppenadministratoren gelesen werden können. Fragen und Themen können mit Tags versehen werden. Über Tags ist es möglich verwandte Fragen und Themen auch über Gruppen hinweg zu finden. Auditorium setzt bei der Verwaltung der Inhalte hauptsächlich auf die Mitwirkung der Community. So kann jeder Nutzer Gruppen anlegen und Beiträge erstellen. Zusätzlich werden Tools zur Verfügung gestellt, mit deren Hilfe die Community erstellte Inhalte selbst regulieren kann:

- Jeder Nutzer kann Beiträge bewerten und damit über die Nützlichkeit abstimmen.
- Antworten können als ’hilfreich’ markiert werden. Dies soll Lesern dabei helfen, schnell zu entscheiden welche Antworten die Frage des Autors sinnvoll beantwortet haben und welche nicht.
- Moderatoren und Mitglieder einer Gruppe helfen bei der Regulierung, indem sie Beiträge verwalten und Ankündigungen verfassen.

3. Analyse des aktuellen Standes

3.1. Graphicuss (Chen)

In seiner Arbeit beschreibt und implementiert Chen das grafische Diskussionssystem Graphicuss(Chen 2016). Es handelt sich um eine Kombination aus einem herkömmlichen Forensystem und einem grafischen Editor. Ein einfaches Rollenkonzept, welches zwischen Studenten, Tutoren und Administratoren unterscheidet, ist integriert. Die Basisfunktionalität deckt die im nachfolgenden beschriebenen Bereiche ab.

3.1.1. Kursverwaltung (course management)

Ein Nutzer in der Tutorenrolle ist in der Lage Kurse anzulegen und bestehende Kurse zu modifizieren. Beim Anlegen des Kurses kann der Name und eine Beschreibung festgelegt werden, sowie zur besseren Erkennung ein Hintergrundbild hochgeladen werden. Beim Anlegen eines Kurses wird ein eindeutiger Code generiert, durch welchen er von den Studenten gefunden werden kann. Falls ein Student an einem Kurs besonders interessiert ist, kann er diesen zu seinen Favoriten hinzufügen, um ihn später leicht wiederzufinden.

3.1.2. Frageverwaltung (question management)

Ein Student der eine Frage zu einem bestimmten Kurs hat, kann innerhalb dieses Kurses eine Frage erstellen. Der erstellende Student hat die Möglichkeit seine Frage zu editieren. Sollte es noch keine Beiträge zu seiner Frage geben, kann sie ganz zurückgezogen werden. Eine Frage bewerten zu können ist entscheidend um eine bessere Community zu bilden. Um dies zu realisieren kann ein Nutzer eine Frage als hilfreich (+1) oder nicht hilfreich (-1) markieren. Das absolute Abstimmungsergebnis wird neben jeder Frage angezeigt. Auch Fragen können, analog zu Kursen, favorisiert und später leicht wiedergefunden werden.

3.1.3. Antwortverwaltung (answer management)

Ein Nutzer kann zu jeder Frage eine Antwort hinzufügen. Später kann er diese Antwort verändern oder löschen. Antworten können nach dem selben Prinzip wie Fragen bewertet werden. Die Antwort mit dem höchsten Abstimmungsergebnis wird oben in der Liste angezeigt. Antworten können außerdem zitiert werden. So können Nutzer weitere Informationen zusätzlich zur ursprünglichen Antwort hinzufügen oder auf Fehler in einer Antwort hinweisen.

Um das System an die komplexen Anforderungen moderner Lernmethoden anzupassen, wurde die Basisfunktionalität um zwei weitere Aspekte erweitert.

3.1.4. Grafischer Editor (drawing tool)

Um bei komplexeren Diskussionen nicht auf reinen Text angewiesen zu sein, wurde ein grafischer Editor entwickelt. Mit Hilfe dieses Editors können Nutzer zu ihren Antworten vorgegebene Elemente wie Rechtecke, Kreise, Linien usw. hinzufügen. Diese Funktionalität soll den Nutzer unterstützen seine Gedanken besser ausdrücken zu können. Dabei sind die folgenden Punkte besonders entscheidend.

Unterscheidung der Elemente. Alle Elemente können in Position, Farbe, Form und Größe frei angepasst werden. Dadurch können Nutzer wichtige Teile ihrer Zeichnung hervorheben

Zeichenhistorie. Um Fehler beim Zeichnen abzufangen sowie die flexible Änderung bereits hinzugefügter Elemente zu gewährleisten, gibt es eine Historie aller Änderungen. Über diese kann mit „undo“-/ und „redo“-Funktionalität ein früherer Zustand wiederhergestellt werden.

3.1.5. Echtzeitfähigkeit (realtime)

Die Interaktivität der Anwendung zu steigern und damit das Engagement der Benutzer zu erhöhen ist ein wichtiges Designziel von Chen. Um dieses Ziel zu erreichen schlägt er die zwei folgenden Echtzeitfunktionalitäten vor.

Echzeit Fragenliste. Die Frageliste muss nicht angefragt werden, stattdessen werden neue Fragen automatisch in die Liste eingefügt (push).

Echtzeit Antwortsortierung. Ohne die Seite erneut laden zu müssen werden die Antworten neu sortiert sobald eine Bewertung abgegeben wurde.

3.1.6. Aktueller Stand des Antwort-Editors

Chens Antwort-Editor verfügt über die Funktionalität welche man von einem Texteditor erwarten würde. Der Text kann fett, kursiv, unter- und durchgestrichen dargestellt werden. Verschiedene Blocksätze und das Formatieren von Quellcode sind möglich.

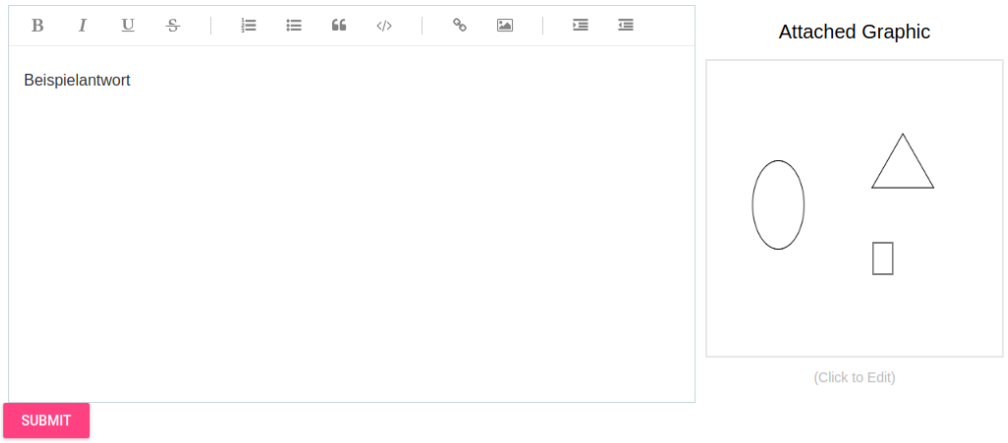


Abbildung 3.1.: Der alte Texteditor in Graphicuss

Eine Grafik kann hinzugefügt werden, indem auf ein Vorschaubild geklickt wird. Danach öffnet sich der grafische Editor. Hierbei ist zu Beachten, dass der grafische Editor den Platz

des Texteditors einnimmt. Während eine Zeichnung bearbeitet wird, lässt sich der Text nicht editieren. Im grafischen Editor können Rechtecke, Kreise und Ovale hinzugefügt werden. Diese lassen sich in Farbe, Form und Position beliebig anpassen. Besonders fällt auf, dass jedes Objekt sich auch nach dem Platzieren auf dem Canvas noch verändern lässt.

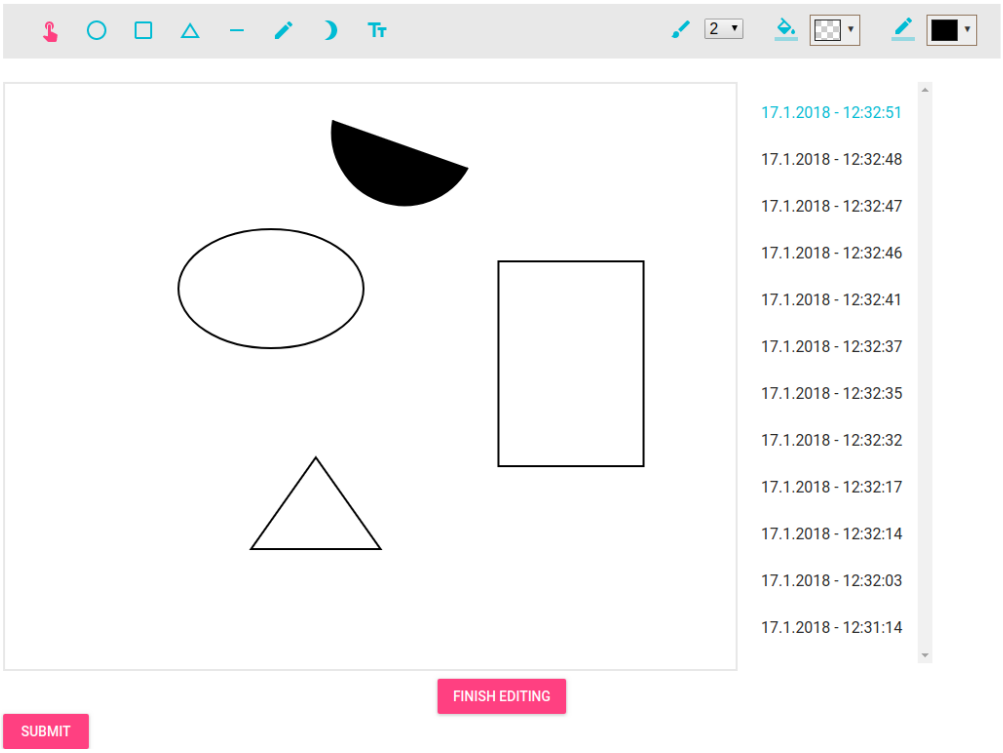


Abbildung 3.2.: Der alte grafische Editor in Graphicuss

Für jede Veränderung des Canvas' wird ein Eintrag in der Historie angelegt. Jeder Eintrag ist mit einem Zeitstempel versehen. Durch den Klick auf einen Zeitstempel wird der Zustand des Editors zum Zeitpunkt des Zeitstempels wiederhergestellt. Ist die Zeichnung zufriedenstellend, kann mit einem Klick auf „Finish Editing" zum Texteditor zurückgekehrt werden. Das Vorschaubild zeigt nun die angefertigte Grafik an.

3.1.7. Zitierfunktion

In der letzten Version von Graphicuss ist es möglich andere Antworten zu zitieren. Diese Funktionalität ist aus klassischen Foren bekannt. Dabei wird an der eigenen Antwort vermerkt, auf welche vorherige Antwort man sich bezieht. In Chens Graphicuss hat dies keine weiteren Auswirkungen auf die Grafik oder den Inhalt des Texteditors.

3.2. Analyse der Bedienungsmetaphern für den grafischen Editor

Iljassova konzipiert in ihrer Arbeit die Anforderungen an den grafischen Editor von Graphicuss und gibt Empfehlungen für Metaphern zum Aktionsverlauf (Iljassova 2017).

3.2.1. Nutzungskontexte in Graphicuss

Um die Bedienungsmetaphern bewerten zu können, müssen zunächst die Nutzungskontexte in Graphicuss beschrieben werden. Anhand der Kontexte kann dann eine Einordnung der Metaphern erfolgen, bzw. können sich Metaphern als ungeeignet erweisen.

Im Folgenden werden die laut Iljassova wichtigsten Nutzungskontexte definiert.

Textlastige Kommunikation. Text ist einer der natürlichsten Wege, beliebige Informationen zu übermitteln. In V-IWB wird Text hauptsächlich genutzt um Elemente zu beschriften oder Kommentare einzufügen.

Diagrammlastige Kommunikation. Mithilfe von geometrischen Formen lassen sich hervorragend Diagramme und Graphen erstellen. Sie werden von fast jedem Whiteboard angeboten. Diese Art der Kommunikation wird im Bildungsbereich eingesetzt, sie zeichnet sich durch wiedererkennbare Formen und ihre Übersichtlichkeit aus. Auch bei der Erstellung von Diagrammen nutzt man Text für die Beschriftung.

Skizzenlastige Kommunikation. Eine der häufigsten Einsatzmöglichkeiten von V-IWB ist die Erstellung von Skizzen mit Hilfe des Stift-Werkzeugs. Mit einer Skizze können Ideen anschaulich dargestellt werden, diese Art der Kommunikation ähnelt dem Arbeiten an einer Tafel, einem klassischen Whiteboard oder einer Skizze auf Papier.

Bildlastige Kommunikation. Bilder sind ein potentes Kommunikationsmittel. Sie werden schneller erfasst und übermitteln Inhalte direkter als andere Formen der Kommunikation. In V-IWBs werden Bilder meist direkt eingefügt oder in Kombination mit anderen Zeichenwerkzeugen genutzt.

Es ist wichtig anzumerken, dass in einem V-IWB all diese Formen der Kommunikation in beliebiger Kombination genutzt werden können.

3.2.2. Metaphern für den Aktionsverlauf

Iljassova beschreibt die Auswahl der Kandidatenmetaphern als ersten Schritt des metapherbasierten Entwurfsprozesses nach (Erickson 1995) und (Madsen 1994). Folgende Kandidatenmetaphern wurden als Teil des Entwurfsprozesses ausgewählt:

Metapher	Beschreibung
Film	Die Bedienelemente haben Pendants zu Videoplayern und Recordern aus der echten Welt. Zumindest Play- und Pauseknöpfe müssen vorhanden sein.
Karussel	Hierzu zählen bspw. Slider und Slideshow. Inhalte werden in einer Liste (vorzugsweise mit Bildern) dargestellt. Die Darstellung kann sowohl horizontal als auch Vertikal geschehen. Die Darstellungsform eignet sich, um Bildergalerien oder auch Nachrichten kompakt und geordnet darzubieten.
Kartenstapel	Diese Metapher kann teilweise als Karussel-Metapher betrachtet werden. Sie besitzt ein einfaches Grundprinzip: auf jeder Karte kann Inhalt dargestellt werden. Durch das Klicken einer Schaltfläche kann zwischen den einzelnen Karten gewechselt werden.

Tabelle 3.1.: Die Kandidatenmetaphern für Graphicuss

In Schritt Zwei des metapherbasierten Ansatzes müssen die Kandidaten-Metaphern hinsichtlich ihrer Bekanntheit und Eignung bewertet werden. Iljassova kommt zu dem Schluss,

dass alle drei Kandidaten-Metaphern für die Nutzungskontexte in Graphicuss geeignet wären.

In einem ’blank paper’-Ansatz bat Iljassova im Anschluss 5 Nutzer ein Diagramm zu skizzieren, in welchem der zeitlichen Ablauf in Graphicuss darstellbar wäre. Die dabei entstanden Diagramm sind in Abbildung 3.3 zu sehen.

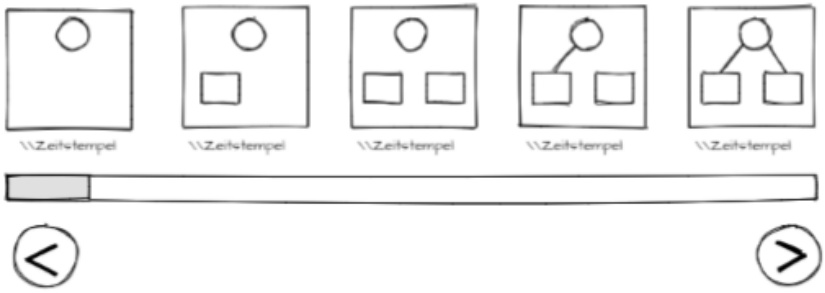


Abbildung 5.1: Skizze zum Slider-UI mit 2 Navigationsvarianten

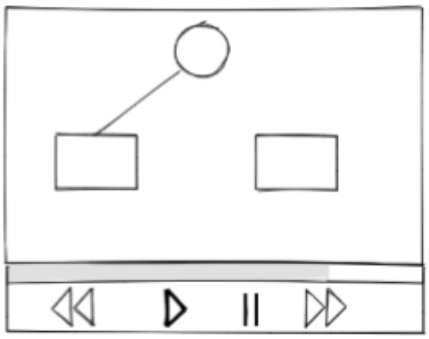


Abbildung 5.2: Skizze zum VideoPlayer-UI

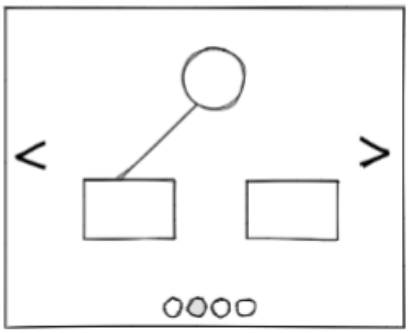


Abbildung 5.3: Skizze zum Slideshow-UI

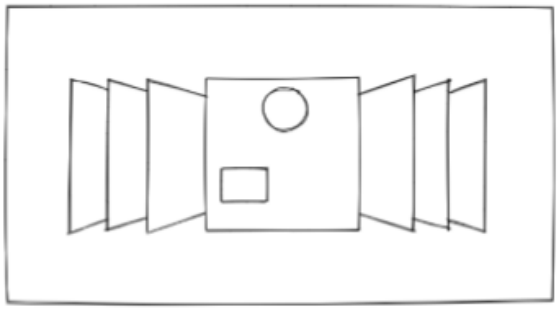


Abbildung 5.4: Skizze zum Karussell-UI

Abbildung 5.5: Skizze zum Kartenstapel-UI

Abbildung 3.3.: Die von Iljassova gesammelten Diagramme zur Darstellung des zeitlichen Ablaufs.

Danach wurden die erstellten Diagramme in Bezug auf ihre Erfüllung der Anforderungen an Graphicuss bewertet. Dabei fiel auf, dass keine eindeutige Lösung existiert. Allerdings wurden nur **Slider**, **VideoPlayer** und **Slideshow** in der Evaluation berücksichtigt. Karussel und Kartenstapel-UIs ließen sich auf Grund der Gesamtbewertung ausschließen.

3.2.3. Evaluation der Diagramme

In der Evaluation wurden 15 Probanden gebeten eine Testaufgabe (siehe Unterabschnitt 3.2.3) zu erfüllen, dabei ihre Gedanken laut auszusprechen und hinterher eine Reihe vordefinierter Fragen (siehe Tabelle 3.2) zu beantworten.

> **Testaufgabe:** Während einer Prüfungsvorbereitung nutzt ihr und eure Kollegen AMCS, um Fragen zu klären. Gestern hat ein Student einen Beitrag mit einer Frage gepostet. Heute früh habt ihr gesehen, dass jemand die Frage beantwortet hat. Die Antwort wurde mit Hilfe von Graphicuss erstellt und beinhaltet sowohl geometrische Primitive als auch Text und Formeln. Ihr habt einen Fehler in dieser Antwort gefunden und wollt ihn korrigieren ohne alles neu zeichnen zu müssen. Ihr wisst, dass Graphicuss die Möglichkeit anbietet, den Zeichenverlauf eines erstellten Beitrags zu verfolgen und einen gewissen Zustand zu zitieren, um diesen als Ausgangspunkt euer Korrekturen zu verwenden. Eure Aufgabe besteht darin, die Nutzerschnittstelle zu bedienen und einen beliebigen Zustand auszusuchen. Während des Versuchs ist es wünschenswert, laut zu denken und eure Handlungen zu kommentieren. Nach dem Versuch werden anschließend ein paar Fragen gestellt.
>
> Anmerkung: Mögliche Fragen bezüglich der Funktionsweise von AMCS und Graphicuss werden vom Moderator beantwortet.

Abkürzung	**Frage**
F-1.	Ist die Darstellungsform und die zugrundeliegende Metapher passend und verständlich?
F-2.	Sind die Icons verständlich?
F-3.	Ist die Richtung des zeitlichen Verlaufs intuitiv nachvollziehbar?
F-4.	Könnten Sie sich diese Design-Variante nicht nur auf dem PC/Laptop sondern auch auf dem Smartphone/Tablet vorstellen?
F-5.	Sind zusätzliche Informationen zu dem Zustand notwendig bzw. wünschenswert?
F-6.	Ist der Unterschied zwischen zwei benachbarten Zuständen erkenntlich?

Tabelle 3.2.: Die vordefinierten Fragen der Auswertung

Iljassova kommt bei der Auswertung der Ergebnisse (siehe Abbildung 3.4) zu dem Schluss, dass alle verbleibenden UI-Skizzen grundsätzlich geeignet sind. Sie empfiehlt das Slideshow-UI für die Darstellung des zeitlichen Verlaufs von Graphicuss besonders auf Grundlage der möglichen Geräteunabhängigkeit (siehe Tabelle 3.2, F-4).

	F-1	F-2	F-3	F-4	F-5	F-6
Slider-UI	++	++	++	-	-	-
	++	+	++	+	++	-
	++	++	++	-	-	-
	++	+	++	-	-	-
	++	+	++	-	-	-
Slideshow-UI	++	++	++	++	-	++
	++	+	++	++	++	-
	++	++	++	++	-	+
	++	+	++	++	++	-
	++	+	++	++	-	-
VideoPlayer-UI	++	++	++	++	-	-
	++	++	++	+	-	-
	++	+	++	+	-	-
	++	+	++	+	-	-
	++	++	++	-	-	-

*Legende: **++** = ja, **+** = ja, unter der Voraussetzung, **-** = nein*

Abbildung 3.4.: Die Auswertung zu den UI Skizzen

4. Anforderungsanalyse

In der Anforderungsanalyse werden die funktionalen und nicht-funktionalen Anforderungen an die zu erstellende Anwendung zusammengetragen. Dabei sollen sowohl technische Aspekte als auch Anforderungen des Kunden, in diesem Fall des Lehrstuhls für Rechnernetze, einfließen.

4.1. Funktionale Anforderungen

Funktionale Anforderungen geben an, was ein System leisten muss.

4.1.1. Funktionale Anforderungen Graphicuss allgemein

Tabelle 4.1 listet die funktionalen Anforderungen für die Neuimplementierung von Graphicuss.

Abkürzung	Anforderungsbeschreibung
A-1	Nutzer kann eine Liste aller Kurse aufrufen.
A-2	Nutzer kann eine Liste aller Fragen abrufen, sie sind nach Bewertung sortiert.
A-3	Nutzer kann eine neue Frage erstellen, zusätzlich eine Zeichnung erstellen und anhängen.
A-4	Nutzer kann alle bisherigen Antworten zu einer Frage abrufen, sie sind nach Bewertung sortiert. Enthalten sie eine Zeichnung, wird eine Vorschau dargestellt.
A-5	Nutzer kann eine Antwort erstellen, zusätzlich eine Zeichnung erstellen und anhängen.

Tabelle 4.1.: Funktionale Anforderungen für Graphicuss

4.1.2. Funktionale Anforderungen für den grafischen Editor

4.2. Nicht-funktionale Anforderungen

Die nicht-funktionalen Anforderungen beschreiben Anforderungen welche über die rein funktionalen Merkmale hinausgehen. Dazu zählen beispielsweise Zuverlässigkeit, Bedienbarkeit oder Korrektheit.

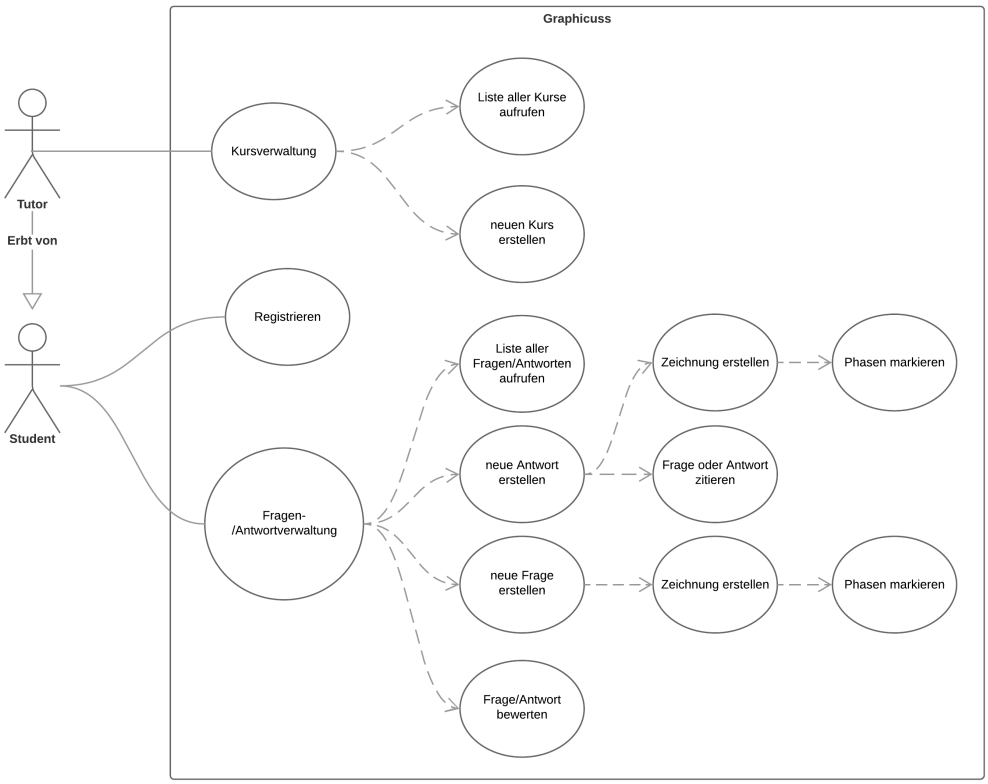


Abbildung 4.1.: Das Use-Case Diagram für Graphicuss

Für Graphicuss ist eine wichtige nicht-funktionale Anforderung die intuitive Bedienbarkeit. Diese wird im Rahmen der Evaluation überprüft.

Abkürzung	**Anforderungsbeschreibung**
AGE-1	Beim Zitieren einer Frage oder Antwort wird die gesamte Bearbeitungshistorie der zu zitierenden Frage oder Antwort in den Editor des Nutzers geladen.
AGE-2	Nutzer kann in seiner Bearbeitungshistorie Zustände löschen.
AGE-3	Nutzer kann in seiner Bearbeitungshistorie Zustände als Phasen markieren.
AGE-4	Nutzer kann in seiner Bearbeitungshistorie vor und zurück springen, sowohl in einzelnen Schritten als auch nur zwischen Phasen.

Tabelle 4.2.: Anforderungen für den grafischen Editor

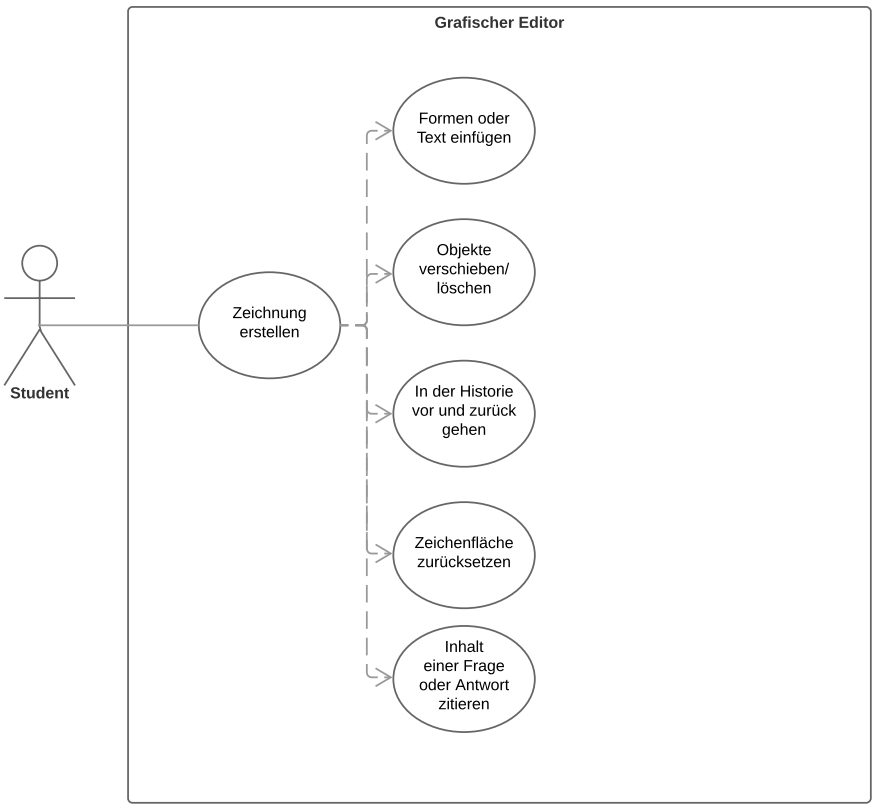


Abbildung 4.2.: Use-Case Diagramm für den grafischen Editor

5. Konzepte zur Erweiterung des bisherigen Standes

5.1. Erweiterung der Historie

Um die Benutzerfreundlichkeit und Benutzbarkeit von Graphicuss weiter zu erhöhen werden die nachfolgenden Punkte vorgeschlagen.

5.1.1. Speichern der Historie an der Frage oder Antwort

Im ersten Entwurf von Graphicuss durch Chen wird die Historie nur auf dem Client zwischengespeichert. In dieser Arbeit wird ein Konzept aufgestellt, welches ermöglicht, zu jeder Frage oder Antwort die gesamte Bearbeitungshistorie mit an den Server zu senden. Dadurch ist die Bearbeitungshistorie Teil jeder Zeichnung und kann zu jedem Zeitpunkt wieder in den Editor geladen werden.

5.1.2. Zitieren beliebiger Zustände aus der Bildhistorie

Bei Chen war es bereits möglich, beliebige Antworten zu zitieren. Allerdings bezog sich dabei das Zitat nur auf den Text. Mithilfe des vorangegangenen Punktes ist es in der überarbeiteten Version von Graphicuss möglich auch in der Historie einer anderen Antwort zu navigieren. Wird eine Antwort zitiert, so wird die Historie dieser Antwort automatisch in den grafischen Editor des Nutzers geladen.

5.1.3. Thumbnails

Thumbnails sind kleine Vorschaubilder zu einer eigentlich größeren Grafik. Sie können schneller verarbeitet werden als größere Darstellungen und nehmen in der Benutzeroberfläche weniger Platz weg. In der Historie ermöglichen sie das schnelle Navigieren zwischen verschiedenen Zuständen. So kann ein Nutzer schon vor dem Anklicken entscheiden, ob es sich um den von ihm gesuchten Zustand handelt.

5.1.4. Löschen von Zuständen

Bei der Erstellung einer komplexen Grafik entstehen viele Zustände. Dies wirkt sich zum Einen negativ auf den benötigten Speicherplatz aus, da je Zustand eine Grafik abgespeichert werden muss. Zum Anderen erschwert es das Zitieren eines bestimmten Zustandes für

Abbildung 5.1.: Die überarbeitete Historie in Graphicuss

andere Nutzer, da dieser zunächst in einer großen Menge an Zuständen gefunden werden muss. Eine Möglichkeit diesem Problemen zu begegnen ist es, dem Nutzer zu ermöglichen Zustände zu löschen. Ist ein Nutzer unzufrieden mit einem Zustand, hat er die Möglichkeit diesen per Klick auf das Abfallsymbol aus der Historie zu entfernen (Siehe Abbildung 5.2).

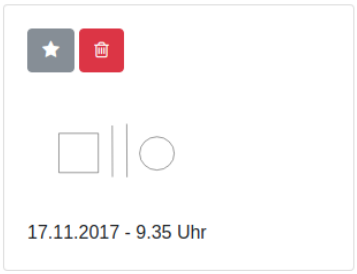


Abbildung 5.2.: Ein einzelner Zustand der Historie

5.1.5. Animationen

Animationen können wichtige Konzepte spielerisch begreifbar machen. Außerdem lockern sie die Benutzerinteraktion auf. Im Falle der Historie gibt es zwei Animationen:

Fade in. Kommt ein Zustand in der Historie hinzu, gleitet dieser von links hinein

Fade out. Wird ein Zustand gelöscht, gleitet er nach rechts aus der Historie

5.2. Phasenkonzept

Als Phase bezeichnet man einen wichtigen Zustand in der Bearbeitungshistorie. Diese können durch den Nutzer selbst festgelegt werden. Phasen helfen dem Nutzer seine erstellte Grafik vor dem Abschicken selbst zu Prüfen. Außerdem erlauben sie es anderen Nutzern beim Zitieren schnell zwischen wichtigen Zuständen zu wechseln.

5.2.1. Markieren von Zuständen als Phasen

Durch einen Klick auf das Sternsymbol kann ein Zustand in der Historie als Phase festgelegt werden (Siehe Abbildung 5.2). Der Stern wird daraufhin golden hinterlegt. Wenn die Antwort abgeschickt wird, werden auch die markierten Phasen als Teil der Historie mitgeschickt. Andere Nutzer, welche eine Frage zitieren, können die Phasen als Orientierung nutzen.

5.2.2. Springen zwischen Phasen

In der Benutzeroberfläche gibt es zwei Knöpfe mit jeweils dem Phasen-Stern und einem Richtungspfeil. Darüber kann zwischen Phasen in der Historie gewechselt werden. Diese Knöpfe ermöglichen es dem Nutzer schnell zur nächsten oder vorherigen Phase zu wechseln und diese nicht manuell suchen zu müssen. Über ein Plugin wurde außerdem eine Scroll-Funktionalität eingebaut, welche in der Liste automatisch zur nächsten gefundenen Phase scrollt, so dass nach dem Klicken der Knöpfe immer der aktuell aktive Zustand in der Historie zu sehen ist.

6. Implementierung

Das vorliegende Kapitel beschreibt die gewählte Vorgehensweise bei der Neuimplementierung von Graphicuss.

Wie in Kapitel 3 gezeigt, basiert Chens Graphicuss auf einer klassischen Server-Client Architektur. Diese wurde in der Neuimplementierung beibehalten. Dabei wurden am Serverteil einige Anpassungen gemacht um neue Features zu ermöglichen. Diese werden in Abschnitt 6.4 beschrieben. Das Frontend wurde von Grund auf neu implementiert. Dieser Prozess ist in Abschnitt 6.3 erläutert.

6.1. Technologieauswahl

Die vorherige Implementierung von Graphicuss durch Chen nutzte das JavaScript Frontend Framework React.js. Diese Arbeit stellt eine Implementierung in Angular.js vor. Dies geschieht um eine bessere Anbindung an AMCS zu gewährleisten. AMCS nutzt Angular.js für das Frontend. Daher kann eine Graphicuss-Angular Implementierung einfacher integriert werden und die Wartbarkeit wird erhöht, da der Technologiestack einfacher gehalten wird.

6.2. Aktueller Stand Web Technologien

Angular.js wird von Google betreut und ist in der Javascript-Welt weit verbreitet. Besonders für komplexe Single Page Applications (SPAs) ist es sehr beliebt. Die Aufteilung der Logik in Komponenten erlaubt es Funktionalität zu kapseln und Änderungen an großen Webapplikationen auch in Teams reibungslos zu gestalten. Ein weiterer Vorteil ist die sehr ausführliche Dokumentation. Sie erlaubt einen schnellen Einstieg durch ein einfaches Tutorial, welches dennoch sämtliche Funktionalität einer modernen Webanwendung beinhaltet.

6.3. Implementierung des Frontends

Das Frontend wurde von Grund auf in Angular.js implementiert. Hierbei wurde die ausgezeichnete Dokumentation von Google zu Hilfe genommen[1].

Angular.js baut auf TypeScript auf. Dieses ermöglicht es unkompliziert Klassen und Vererbungen in JavaScript zu benutzen. Außerdem findet eine Kompilierung in JavaScript statt, was eine statische Überprüfung des Codes vor der Ausführung ermöglicht. Diese Eigenschaft von TypeScript erwies sich als sehr nützlich, da viele Ungenauigkeiten und Fehler

[1] https://angular.io/docs, zuletzt aufgerufen am 25.04.2018

bereits vor der Ausführung durch den Compiler gemeldet wurden und damit nicht zur Laufzeit debuggt werden mussten.

6.3.1. Implementierung des grafischen Editors

Für den ersten Entwurf des Editors wurde das Mockup von Chen verwendet (siehe Abbildung 6.1), welches er in seinem Prototypen letztendlich nur abgeändert umsetzte. Er entschied sich für eine Variante der Historie, in der statt einer Vorschaugrafik Zeitstempel an den einzelnen Zuständen zu sehen sind.

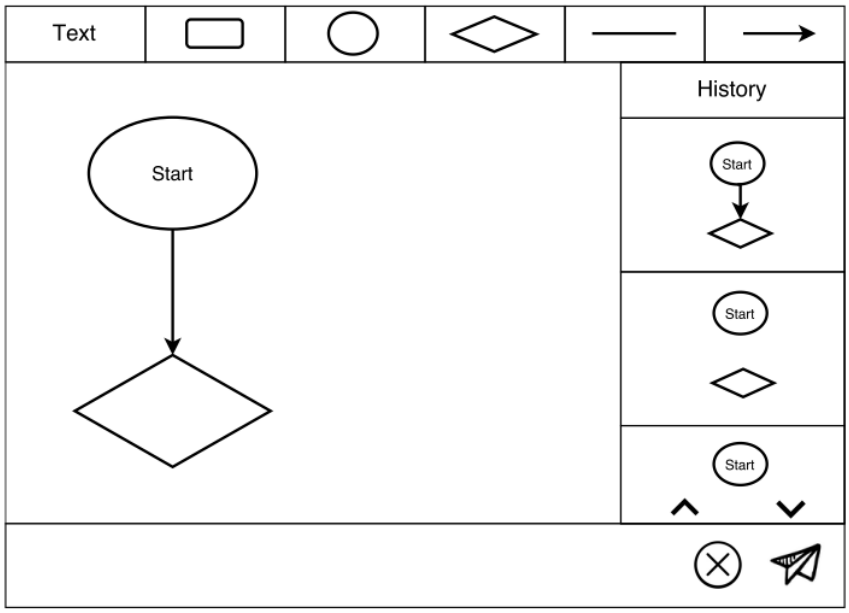


Figure 3.7.: Mockup: Drawing editor with drawing history

Abbildung 6.1.: Das Mockup für den Editor von Chen

Um der Historie des Editors Vorschaugrafiken wie in Abbildung 6.1 hinzuzufügen, wurde das Datenmodell des Frontends angepasst. Es wurde ein Objekt *HistoryEntry* angelegt, welches einen einzelnen Eintrag in der Historie repräsentiert (siehe Quelltext 6.1). An diesem Objekt kann nun die zugehörige Zeichnung, sowie die Information, ob es sich um eine Phase handelt, gespeichert werden. Mit der angehängten Zeichnung ist es möglich, ein Vorschaubild für jedes Historienobjekt zu generieren. Das Historienobjekt ermöglicht mit Angular.js außerdem eine einfachere Verarbeitung im Frontend, da eine typsichere Liste dieser Objekte leicht angezeigt werden kann.

```
1  export class HistoryEntry {
2    created: Date
3    data: string
4    isPhase: boolean
5    timestamp: number
6    _id: number
7
8    constructor(created: Date, data: string, isPhase: boolean) {
9      this.created = created
10     this.data = data
11     this.isPhase = isPhase
12     this.timestamp = created.getTime()
```

```
13   }
14 }
```

Quelltext 6.1: Das Datenmodell für den HistoryEntry

Ein weiterer Vorteil von Angular kam bei der Implementierung des Datenmodells auf Backendseite zum Tragen. Durch die Kapselung als *HistoryEntry* kann das Objekt einfach in JSON umgewandelt und an den Server gesendet werden. Im konkreten Beispiel wird eine Liste von *HistoryEntrys* an eine Frage (*Question*) oder Antwort (*Answer*) gehängt. Dazu wurden die Objekte *Question* und *Answer* um das Attribut *history* erweitert. Siehe Datenmodell in Abbildung 6.4. Dies ermöglicht das Zitieren eines Bildes, wobei auf die gesamte Historie des Bildes zugegriffen werden kann. Die notwendingen Anpassungen am Backend werden in Abschnitt 6.4 näher erläutert.

Da Angular.js gut dokumentierte Animationen standardmäßig anbietet, konnten diese leicht für die Historie implementiert werden. Beim Erstellen eines Historieneintrages gleitet dieser scheinbar von links außerhalb der Liste herein, was dem Nutzer das visuelle Feedback der Änderung geben soll. Wird ein Eintrag gelöscht, gleitet er nach rechts aus der Liste. Auch hiermit werden dem Nutzer die Konsequenzen seiner Handlung (in diesem Fall des Löschens) über die Animation verdeutlicht. Über die Pfeiltasten und Buttons oberhalb der Historie kann zwischen vorherigen und nächsten Zuständen gesprungen werden. Außerdem gibt es Buttons um zum nächsten oder vorherigen Phasenzustand zu springen. Auf dieses Konzept wird in Unterabschnitt 6.3.2 näher eingegangen.

Da im Historienfenster immer nur eine begrenzte Anzahl Einträge gezeigt werden kann war es notwendig den Nutzer immer wissen zu lassen wo er sich gerade befindet. Dies gilt im besonderen wenn er über die Pfeiltasten oder die Phasensuche zu bestimmten Zuständen springt. Dafür wurde ein Scrollplugin[2] genutzt, welche es ermöglicht zu bestimmten HTML Elementen zu scrollen. Den einzelnen Historieneinträgen wird bei der Erstellung als HTML-id ihr Erstellungszeitstempel gesetzt. Dieser wird auch im Angular-Objekt gespeichert. Wird über die Sprungfunktion ein neuer Zustand gesetzt, kann an das Scrollplugin die id übergeben werden, sodass dieses den entsprechenden Zustand findet und dorthin scrollt. Dadurch sieht der Nutzer wo in der Historie er sich gerade befindet, falls er die List einmal von Hand gescrollt haben sollte.

6.3.2. Implementierung des Phasenkonzeptes

Wie für die Historie musste auch für das Phasenkonzept das Datenmodell angepasst werden. Ein Historieneintrag (*HistoryEntry*) erhielt ein Attribut *isPhase* welches signalisiert, ob es sich um eine Phase handelt. Im Frontend wurde eine Methode **setPhase** implementiert, welche an einem Historieneintrag dieses Attribut setzen kann. Ist ein Zustand bereits als Phase markiert, kann dies durch nochmaligen Aufruf der Funktion rückgängig gemacht werden. Dies wird als 'Toggle' bezeichnet, da ein festes Set aus Zuständen zur Verfügung steht, wobei immer einer der Zustände aktiv sein muss. Über einen Button an jedem Historieneintrag kann diese Funktion aufgerufen werden. Wird die Historie an den Server gesendet, wird das *isPhase* Attribut automatisch mitgesendet. Ein Historieneintrag der als Phase markiert ist, lässt sich im Frontend durch einen goldenen Stern erkennen.

Da die Historie schnell aus sehr vielen Zuständen bestehen kann, ist es wichtig Phasen einfach zu finden. Hierfür wurde eine Suchfunktion implementiert, welche schnell zur nächsten Phase springt. Analog zu den Pfeilbuttons mit denen zum nächsten oder vorherigen Zustand gesprungen werden kann, gibt es Buttons auf denen ein Pfeil und das Phasensymbol zu sehen ist. Mit diesen kann zur nächsten oder vorherigen Phase gesprungen werden. Intern wird dies über eine einzige Funktion realisiert, welche als Parameter mitbekommt ob alle Zustände oder nur Phasen gesucht werden sollen (siehe Quelltext 6.2.

[2] https://www.npmjs.com/package/ng2-page-scroll, zuletzt aufgerufen am 20.04.2018

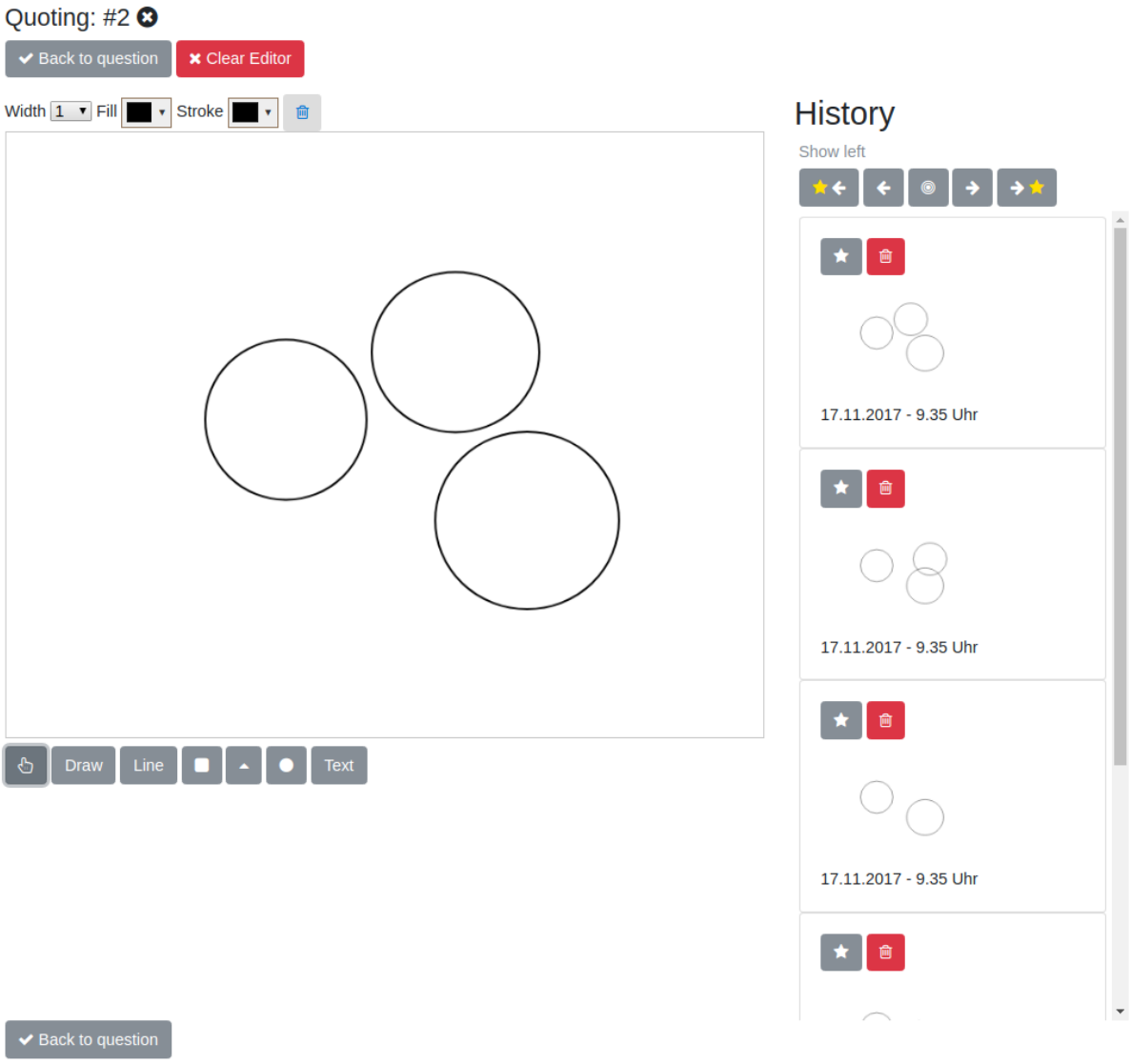


Abbildung 6.2.: Der fertige grafische Editor von Graphicuss

```
jumpInHistory(steps: number, onlyPhases = false) {
    // here we set currentWhiteboardData, which is an input for the whiteboard
    let history = this.history
    let currentState = this.exportCanvas()
    let indexOfCurrentState = history.findIndex(x => x.data === currentState)

    if (indexOfCurrentState === -1) {
      console.log('couldnt find current state in history')
      return
    }
    let pickedState = undefined
    if (!onlyPhases) {
      pickedState = history[indexOfCurrentState + steps]
    } else {

      let index
      if (steps < 0) {
        // search to the left
        for (var i = indexOfCurrentState - 1; i >= 0; i--) {
          if (history[i].isPhase) {
            index = i
            break
          }
        }
      } else {
        index = history.findIndex((elem, idx) => elem.isPhase && idx >
            indexOfCurrentState)
      }
      if (index !== undefined) {
        pickedState = history[index]
      }
      //pickedState = ;
    }
    if (!pickedState) {
      console.log('couldnt jump to state')
      return
    } else {
      let newData = pickedState.data
      // scrolling
      let pageScrollInstance: PageScrollInstance = PageScrollInstance.
          newInstance({
        document: this.document,
        scrollTarget: '#' + pickedState.timestamp,
        scrollingViews: [this.container.nativeElement]
      })
      this.pageScrollService.start(pageScrollInstance)
      this.data = newData
      this.selectedHistory = pickedState
    }

}
```

Quelltext 6.2: Die Funktion zur Bewegung innerhalb der Historie

6.4. Anpassungen am Backend

Wie in Abbildung 6.3 zu sehen ist, besteht das alte Datenmodell von Chen auf einer hierarchischen Struktur aus Kursen (*Course*), welche Fragen (*Question*) enthalten, welche wiederum Antworten (*Answer*) haben können. Diese Struktur wurde übernommen, allerdings mussten einige Anpassungen gemacht werden um die geplanten Verbesserungen der Historie und des Zitierens zu ermöglichen.

Das erweiterte Datenmodell ist in Abbildung 6.4 dargestellt. Zur Implementierung wurde mongoose[3] benutzt, eine JavaScript Bibliothek um leicht Modelle für mongoDB zu erstellen. MongoDB[4] ist die zugrundeliegende Dokumenten-orientierte Datenbank.

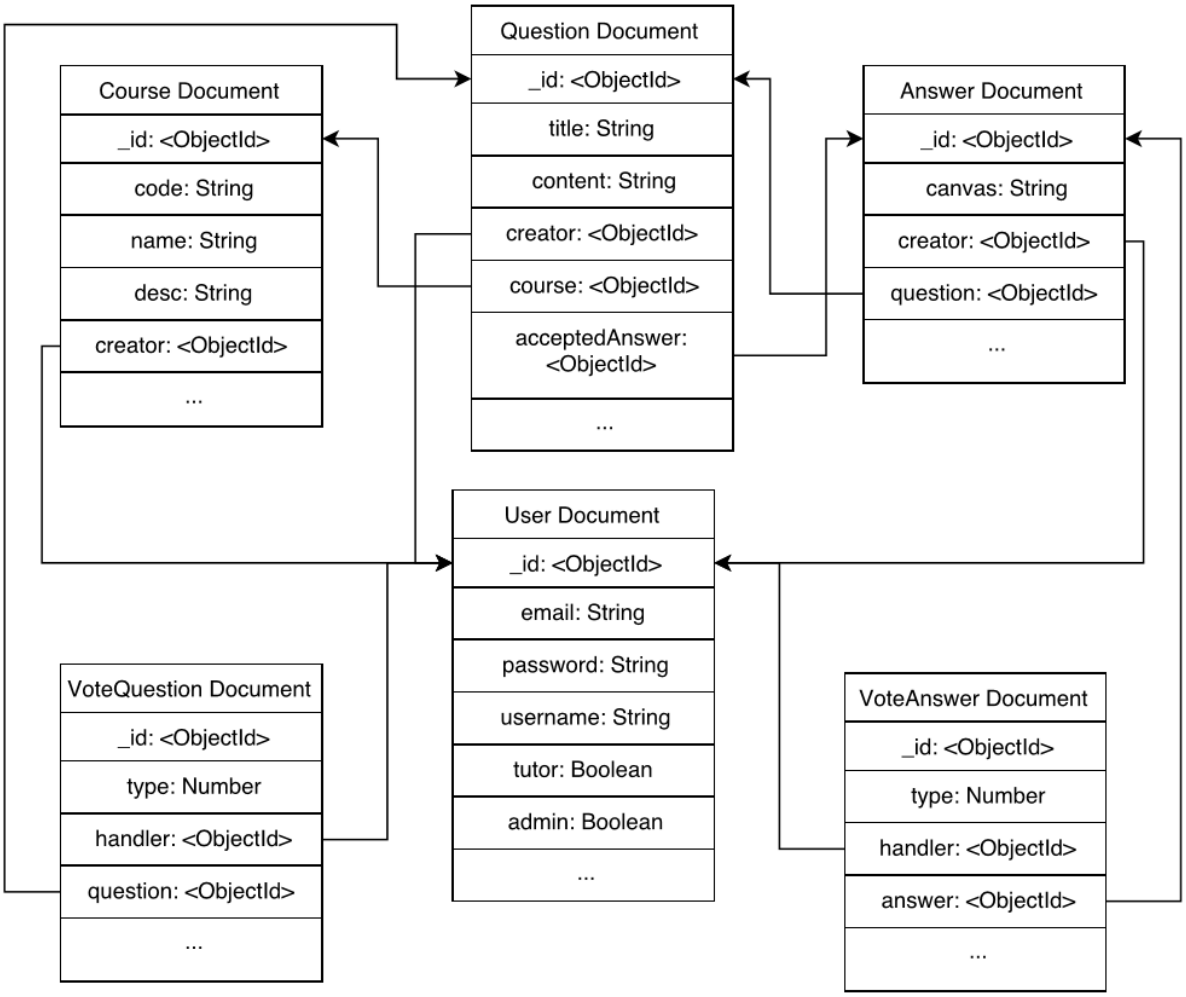


Abbildung 6.3.: Das Datenmodell in Chens Graphicuss

Zunächst musste ein neues Objekt *HistoryEntry* (Quelltext 6.3) hinzugefügt werden, welches einen einzelnen Eintrag in der Historie repräsentiert. Anschließend wurde jeweils der Frage (*Question*) und Antwort (*Answer* eine Liste dieser Objekte hinzugefügt.

```
1  import mongoose from 'mongoose'
2  const Schema = mongoose.Schema
3
4  var historyEntrySchema = mongoose.Schema({
5    data: String,
6    created: Date,
7    isPhase: Boolean
8  });
9
10 // methods ======================
11
12 export default historyEntrySchema
```

Quelltext 6.3: Das neue Schema für einen Historieneintrag

[3]http://mongoosejs.com/, zuletzt abgerufen am 20.04.2018
[4]https://www.mongodb.com/, zuletzt abgerufen am 20.04.2018

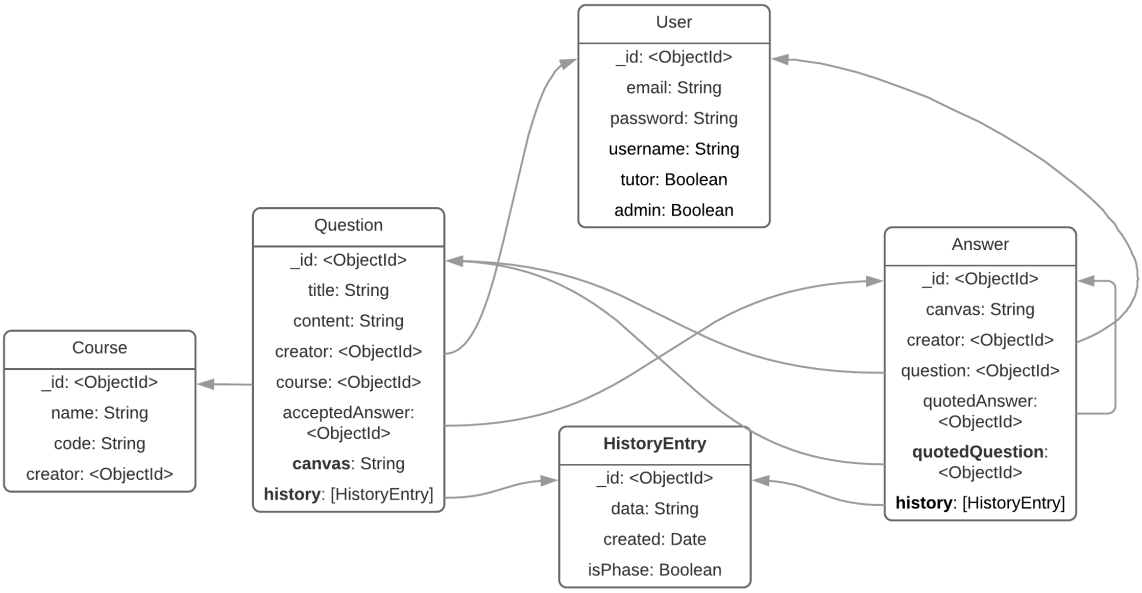


Abbildung 6.4.: Das erweiterte Datenmodell für die Neuimplementierung

Quelltext 6.4 zeigt dies Beispielhaft für das Modell der Antwort. Das neu erstellte Schema *HistoryEntry* wird im Schema der Antwort genutzt um eine Liste der Historieneinträge abzuspeichern.

```
let answerSchema = mongoose.Schema({
  content: String,
  canvas: String,
  // this is a subschema
  history: [historyEntry],
  creator: {type: Schema.Types.ObjectId, ref: 'User'},
  question: {type: Schema.Types.ObjectId, ref: 'Question'},
  quotedAnswer: {type: Schema.Types.ObjectId, ref: 'Answer'},
  quotingAnswers: [{type: Schema.Types.ObjectId, ref: 'Answer'}],
  quotedQuestion: {type: Boolean, default: false},
  deleted: {type: Boolean, default: false},
  vote: {type: Number, default: 0},
}, {timestamps: true})
```

Quelltext 6.4: Das modifizierte Schema für die Antwort

6.5. Technische Details des SSO

Das Single Sign-on Konzept (SSO) soll dafür Sorge tragen, dass die Nutzeraccounts von Graphicuss und AMCS homogen sind. Wenn sich ein Nutzer bei Graphicuss registriert, wird dieser Nutzer auch automatisch in AMCS angelegt, sofern er dort noch nicht existiert. Wenn er bereits in AMCS existiert, wird er dort eingeloggt. Dafür wird das Backend von AMCS mit den Nutzerdaten aufgerufen, welche während der Registrierung in Graphicuss angegeben wurden. Dabei unterscheidet das AMCS Backend zwischen den HTML Antwort Codes 200 für eine neue Accounterstellung und 201 für einen existierenden Account, in welchen sich nur eingeloggt wurde. Quelltext 6.5 zeigt die Implementierung einer Registrierungsfunktion in Graphicuss, welche das Backend von AMCS auf vorhandene Nutzer prüft.

```
export const register = (req, res, next) => {
  res.json(req.user)
}

// check the AMCS login
  let myHeaders = new Headers({
    "Content-Type": "application/json",
    "X-AMCS-API": 3,
  });
  let url = "https://mobileclassroom.inf.tu-dresden.de/api/auth/authenticate"
  let data = {
    "auth": {
      "username": req.body.username,
      "password": password
    }
  }

  fetch(url, {
    method: 'POST', // or 'PUT'
    body: JSON.stringify(data),
    headers: myHeaders
  }).then(res => res.json())
  .catch(error => console.error('Error:', error))
  .then(response => console.log('Success:', response));

  // create the user
  let newUser = new User();
  newUser.email = email;
  newUser.password = newUser.generateHash(password);
  newUser.username = req.body.username;
  newUser.faculty = req.body.faculty;
  newUser.save((err) => {
    if (err) return done(err);
    newUser = newUser.toObject()
    req.user = newUser
    return done(null, newUser);

});
console.log('created a new user in AMCS: ', newUser);
```

Quelltext 6.5: Eine beispielhafte Implementierung einer Registrierungsfunktion für Graphicuss

Vom AMCS Backend[5] wird ein Token mitgesendet, welcher für die Authentifizierung aller Aktionen in der AMCS App genutzt wird.

Da Graphicuss laut Anforderung als eigenständige App laufen soll, ist es nicht sinnvoll, dass beide Apps das selbe Backend benutzen. Momentan ist noch nicht möglich sich nur in AMCS oder Graphicuss anzumelden und in der jeweils anderen App auch angemeldet zu werden. Dafür müsste das AMCS Backend um einen Identitätsprovider erweitert werden, welcher eine Identität zur Verfügung stellt. Diese müsste dann von Graphicuss verifiziert und zur Authentifizierung genutzt werden.

In einer zukünftigen Version wäre es denkbar, AMCS und Graphicuss unter der selben Hauptdomain laufen zu lassen, so dass Cookies oder der LocalStorage von beiden Apps gelesen werden können. Bei einem erfolgreichen Login könnte ein Cookie mit einem JSON-Webtoken gesetzt werden, welcher für alle Aktionen in den Apps zur Authentifizierung genutzt werden würde. Somit wäre es möglich sich nur noch in einer App anmelden zu müssen.

[5] https://amcs.website/apidoc/, zuletzt aufgerufen am 20.04.2018

7. Evaluation

Nachdem das System in den vorherigen Kapiteln beschrieben, entworfen und implementiert wurde, erfolgt in diesem Kapitel die Evaluation der Benutzbarkeit (Usability). Hierbei soll der Fokus auf dem grafischen Editor liegen.

7.1. Usability

Das Testen der Usability eines Systems bedeutet eine empirische Auswertung der Nutzbarkeit mit potenziellen Nutzern. Das Ziel ist das Sammeln von Daten und deren Auswertung zur Behebung von Usability-Fehlern des Systems.

7.1.1. System Usability Scale (SUS)

Die Usability kann immer nur innerhalb eines Kontextes existieren. Das bedeutet, dass es keine absolute Usabilitybewertung geben kann. Dennoch besteht Bedarf für eine breite und generelle Usabilityeinschätzung. Die System Usability Scale (SUS) ermöglicht eine schnelle und günstige, aber dennoch verlässliche Möglichkeit die Usability eines Systems in verschiedenen Kontexten zu testen. Die SUS besteht aus 10 Fragen, welche einen Gesamtsicht auf die subjektive Benutzbarkeit eines Systems geben (Brooke 1996).

Um die SUS auszuwerten muss von der Antwort auf Fragen mit geradem Index 1 subtrahiert werden und für Fragen mit ungeradem Index der Wert von 5 subtrahiert werden. Die resultierenden Werte werden aufsummiert und mit 2,5 multipliziert. Der entstehende Wert ist das Ergebnis der System Usability Scale.

Für die Auswertung zur Usability der Neuimplementierung von Graphicuss wurden 7 Teilnehmer befragt. Der genutzte Fragebogen befindet sich im Anhang. Jeder der Teilnehmer gab eine Bewertung zwischen 1 (Starke Ablehnung) und 5 (Starke Zustimmung) ab.

Folgende Aufgaben sollten ausgeführt werden:

- Stellen einer Frage (Titel: Kreis oder Viereck? Zeichnung: Kreis, Dreieck) dabei ersten Zustand als Phase markieren
- Beantworten dieser Frage mit Zeichnung (Zitieren, ursprüngliche Zeichnung verbessern: Ersetze Dreieck durch Viereck)
- Korrigieren der Zeichnung: Zwei Kreis einfügen
- Zurücksetzen der Zeichenfläche

- Die Antwort upvoten.
- Eigene Antwort zitieren: zwei Dreiecke hinzufügen, finden über Phase

Aus den einzelnen Antworten wurde der Durchschnitt berechnet, mit dem der finale SUS Score ermittelt wurde. Für die vorliegende Umfrage liegt der SUS Score bei 75,4. Laut einem Artikel zur Übertragung des SUS Scores auf eine greifbarere Bewertung liegt dieses Ergebnis zwischen *GUT* und *EXZELLENT* (Bangor, Kortum und Miller 2009). Dieser Wert ähnelt dem von Chen ermittelten SUS Score für die vorherige Version von Graphicuss, welcher bei 73,5 liegt. Nutzer können das System auch nach der Neuimplementierung schnell verstehen und ohne große Hilfe von außen intuitiv benutzen.

Item	A	B	C	D	E	Durchschnitt
1	4	4	4	3	4	3,8
2	2	2	2	2	2	2
3	4	4	4	3	4	3,8
4	2	1	1	1	1	1,2
5	4	4	4	4	4	4
6	1	2	3	2	1	1,8
7	4	3	3	3	5	3,6
8	1	2	2	3	1	1,8
9	4	3	4	3	4	3,6
10	1	3	2	3	1	2

Tabelle 7.1.: Die Ergebnisse der System Usability Scale

7.1.2. Fragebogen zur Usability

Um genauer auf die exakten Anforderungen der vorliegenden Anwendung einzugehen wurde ein zusätzlicher Fragebogen entworfen, der sich spezieller auf Graphicuss bezieht. Um es den Probanden leichter zu machen ihre Bewertung abzugeben, wurde die Skala der System Usability Scale beibehalten: von 1 (starke Ablehnung) bis 5 (starke Zustimmung). Die Auswertung des Fragebogens fand direkt im Anschluss an den SUS Test statt. Der Fragebogen befindet sich im Anhang.

Es fiel auf, dass Fragen zur Bedienung des Editors wie 1 und 6 mit jeweils 4,4 Punkten sehr gut bewertet wurden. Potenzial besteht noch bei der Echtzeitfunktion. Diese wird als generell nützlich angesehen (Frage 8), allerdings wurde das automatische Sortieren der Antworten nicht als besonders intuitiv eingeschätzt (Frage 7).

Die gesamten Ergebnisse der Befragung können der Tabelle 7.2 entnommen werden.

7.1.3. Think-Aloud-Methode

Während der durchzuführenden Aufgaben waren die Teilnehmer angehalten, mittels der Think-Aloud-Methode, welche vorher erklärt worden war, ihre Gedanken zu verbalisieren. Die Methode sieht vor, dass Teilnehmer einer Studie bei der Benutzung eines Systems ihre Gedanken und dabei besonders aufkommende Probleme, Unstimmigkeiten oder positive Eigenschaften, beschreiben.

Während der Studie wurde so äußerst nützliches Feedback gesammelt. Dieses lässt sich grob in vier Kategorien einteilen.

Feedback zu Interface-Komponenten. Drei Nutzer konnten die Funktion einiger Bedienelemente nicht direkt erkennen und wünschten sich Tooltips an den jeweiligen But-

tons. Durch das Bewegen des Mauszeigers auf das Element sollte eine kurze Beschreibung in einer Sprechblase angezeigt werden.

Der Zitieren-Button war einigen Teilnehmern der Studie nicht sichtbar genug. Sie schlugen vor, diesen prominenter zu platzieren oder noch deutlicher hervorzuheben. Ein anderer Teilnehmer schlug vor, das Zitieren auch über ein einfaches Anklicken der Antwort zu ermöglichen.

Die Historie fanden die meisten Nutzer gut und verständlich, was besonders durch die Interviewfragen deutlich wird. Allerdings merkten einige Nutzer an, dass sie eine horizontale Version unterhalb des Editors noch leichter verständlich gefunden hätten.

Ein Nutzer wünschte sich, dass das momentan aktive Tool in der Werkzeugleiste des Editors besser hervorgehoben sein sollte.

Feedback zum Ablauf. Ein Nutzer gab an, dass es sinnvoll sei, die finale Version einer Grafik als Antwort senden zu können, dass aber zusätzlich das Kommentieren des Verlaufs äußerst wichtig sei. Der Teilnehmer wünschte sich dafür eine Kommentarfunktion an den jeweiligen Historieneinträgen. Die Kommentare könnten dem ursprünglichen Ersteller in einer Übersichtsansicht dargestellt werden.

Probleme von Nutzern unterschiedlicher Plattformen. Es fiel auf, dass Mac-Nutzer zum Löschen die Backspace-Taste nutzen wollten. Auf der Windows und Linux Plattform ist es hierfür üblicher die Entfernen-Taste zu betätigen. Dies muss zukünftig für die Belegung der Shortcut-Tasten berücksichtigt werden.

Fehler. Einige Fehler in der Anwendung fielen während des Testens auf:

- Beim Einfügen einer Linie wird nach dem Loslassen der linken Maustaste noch eine weitere Linie eingefügt.
- Manchmal wurde der Nutzer auch bei einem scheinbar leeren Editor gefragt ob er wirklich zitieren möchte. Dieser Test sollte angepasst werden, da er den Nutzer im aktuellen Zustand verwirrt.
- Das Textfeature muss überarbeitet werden. Nachdem Text eingefügt wurde, ist er nicht direkt zu bearbeiten. Außerdem darf die Entfernen-Taste während der Texteingabe nicht das Textelement selbst löschen, sondern nur den ausgewählten Text.

Item	A	B	C	D	E	F	G	Durchschnitt
1	4	4	4	5	5	4	5	4,4
2	3	5	4	4	4	5	3	4,0
3	4	4	5	5	5	4	5	4,6
4	4	4	5	4	5	3	3	4,0
5	5	5	5	5	5	5	4	4,9
6	5	4	3	5	5	5	4	4,4
7	2	4	3	3	3	2	3	2,9
8	4	4	4	5	4	4	3	4,0

Tabelle 7.2.: Die Auswertung der Interviewfragen

8. Zusammenfassung und Ausblick

In dieser Arbeit wurde zunächst das aktuelle Konzept des Graphical Discussion Systems (Graphicuss) untersucht. Dabei wurden verwandte Konzepte analysiert und Anforderungen für eine Neuimplementierung definiert. Anschließend wurde eine neue Version des Frontends auf dem neusten Stand der Technik implementiert und am Backend Veränderungen vorgenommen, welche die neue Version ermöglichen.

Für die Implementierung wurde Angular.js gewählt um Graphicuss in Zukunft besser mit AMCS integrieren zu können.

Im Frontend wurde die Historie durch Vorschaubilder, die Möglichkeit Zustände zu Löschen oder als Phase zu markieren, verbessert. Nutzer können außerdem Fragen oder Antworten zitieren und damit die zugehörige Grafik in ihren Editor laden um sie zu bearbeiten oder zu kommentieren.

Auf Grundlage der Neuimplementierung wurde eine Nutzerevaluation nach der System Usability Scale (SUS) vorgenommen. Diese wurde mit der Evaluation von Chen verglichen und erreichte leicht verbesserte Ergebnisse (75 vs. 73.5 Punkte auf der SUS-Skala).

Im Anschluss wurde eine weitere Evaluation zur Usability durchgeführt. Der Fragebogen ist in Anhang A zu finden. In der Auswertung fiel auf, dass insbesondere die Fragen zur neuen Historie und zum Zitieren von Fragen oder Antworten sehr gut bewertet wurden.

Außerdem wurde durch einen Think-Aloud-Ansatz Feedback der Probanden gesammelt. Dieses lieferte wertvolle Erkenntnisse zur Bedienbarkeit und für die weitere Entwicklung von Graphicuss.

Obwohl der vorliegende Prototyp alle definierten Anforderungen erfüllt und die gewünschte Funktionalität liefert, besteht Verbesserungspotential in Bezug auf Usability und weitere gewünschte Features.

Einbinden bestehender Grafiken. Ein wichtiges Feedback war der Wunsch, bestehende Grafiken in den grafischen Editor einbinden zu können. Dadurch müssten bestehende Grafiken nicht erneut im Editor erstellt werden, sondern könnten zu Diskussionszwecken direkt an eine Frage oder Antwort angehängt und modifiziert werden. Hierbei müssten eventuelle Bedenken zum Copyright Teil des Konzeptes und in diesem Zuge adressiert werden.

Kommentieren von Historienzuständen. In einer zukünftigen Version könnte es ermöglicht werden, einzelne Zustände der Historie zu kommentieren und damit entweder dem Ersteller einer Grafik oder Nutzern, welche die eigene Grafik zitieren, Hinweise zu einzelnen Schritten bereits bei der Erstellung der Grafik zu geben.

Anpassungen bei Keyboardshortcuts. Das Konzept der Keyboardshortcuts muss hinsichtlich verschiedener Plattformen überdacht werden. Während der Evaluierung fiel auf, dass beispielsweise Mac-Nutzer versuchten, Elemente über die Backspace-Taste zu lö-

schen. Dieses Verhalten wurde bei der windows-/linuxbasierten Entwicklung nicht bedacht und muss in einer zukünftigen Version verbessert werden.

Zitate stärker hervorheben. Zitierte Antworten sind momentan nur durch eine textuelle Bemerkung markiert. Diese Markierung könnte in Zukunft durch die Anordnung in einer Baumstruktur oder eine andere sinnvolle Darstellung ersetzt werden.

Besseres Einbinden des Phasenkonzeptes. Das Phasenkonzept stieß generell auf gutes Feedback. Einige Nutzer merkten jedoch an, dass es eine bessere initiale Erklärung zur Benutzung geben sollte. Außerdem könnte für eine Zeichnung direkt angegeben werden, wie viele aktive Phasen existieren, um das Interesse des Nutzers zu wecken, sich diese speziellen Zustände anzusehen.

In dieser Arbeit konnte eine verbesserte Version des grafischen Editors von Graphicuss geplant, implementiert und evaluiert werden. Zum Ablauf des Gesamtprozesses der Erstellung von Fragen und Antworten mit Hilfe des grafischen Editors in der Praxis besteht jedoch weiterer Forschungsbedarf.

A. Anhang

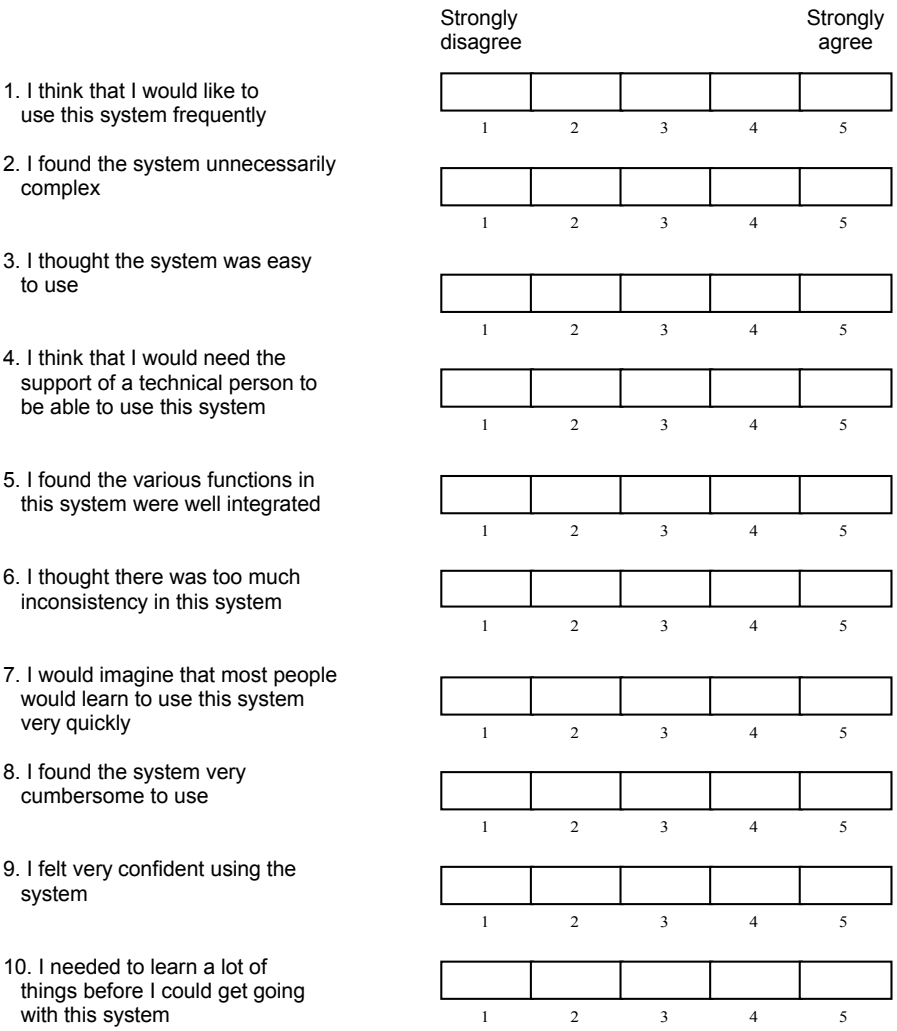
A.1. Usability Fragebogen

Usability Fragebogen

	Starke Ablehnung				Starke Zustimmung
1. Es fiel mir leicht eine Frage zu stellen oder eine Antwort zu geben.	1	2	3	4	5
2. Das Abstimmungssystem ist hilfreich um nützliche Antworten zu finden.	1	2	3	4	5
3. Texteingabe im graphischen Editor ist nützlich / wichtig.	1	2	3	4	5
4. Im graphischen Editor gibt es genug einfügbare Elemente.	1	2	3	4	5
5. Zitieren einer Frage oder Antwort mit Übernahme der Grafik halte ich für sinnvoll.	1	2	3	4	5
6. Ich fand die Historienfunktion (rückwärts/vorwärts) praktisch.	1	2	3	4	5
7. Das automatische sortieren der Antworten war intuitiv verständlich.	1	2	3	4	5
8. Ich kann mir vorstellen, dass die Echtzeitfunktionalität die Anwendung verbessert.	1	2	3	4	5

A.2. System Usability Scale

System Usability Scale



Literatur

[BBH14] Lars Beier, Iris Braun und Tenshi Hara. „auditorium-Frage, Diskutiere und Teile Dein Wissen!“ In: *GeNeMe*. 2014, S. 117–124.

[BKM09] Aaron Bangor, Philip Kortum und James Miller. „Determining What Individual SUS Scores Mean: Adding an Adjective Rating Scale“. In: *J. Usability Studies* 4.3 (Mai 2009), S. 114–123. ISSN: 1931-3357. URL: http://dl.acm.org/citation.cfm?id=2835587.2835589 (besucht am 23. 04. 2018).

[Bra+15] Iris Braun u. a. „Onlinegestützte Audience Response Systeme: Förderung der kognitiven Aktivierung in Vorlesungen und Eröffnung neuer Evaluationsperspektiven“. In: (2015). URL: http://nbn-resolving.de/urn:nbn:de:bsz:14-qucosa-181601.

[Bro96] John Brooke. „SUS-A quick and dirty usability scale“. In: *Usability evaluation in industry* 189.194 (1996), S. 4–7.

[Che16] Kaijun Chen. *Graphical Discussion System*. TU Dresden, Masterarbeit, 2016.

[Eri95] Thomas D. Erickson. „Working with interface metaphors“. In: *Readings in Human–Computer Interaction*. Elsevier, 1995, S. 147–151.

[Har16] Tenshi Hara. „Analyses on tech-enhanced and anonymous Peer Discussion as well as anonymous Control Facilities for tech-enhanced Learning“. de. In: (März 2016). URL: http://www.qucosa.de/recherche/frontdoor/?tx_slubopus4frontend[id]=20551 (besucht am 22. 03. 2018).

[Ilj17] Anastasia Iljassova. „Nutzerschnittstellenentwurf für unterschiedliche Nutzungskontexte zur Visualisierung von Zeitinformationen innerhalb Canvas-basierter Diskussionsbeiträge in Graphicuss“. In: (Juni 2017).

[KHB16] Tommy Kubica, Ing Tenshi Hara und Ing Iris Braun. „Entwicklung eines Prototyps zur Auswahl und zum Einsatz technischer Werkzeuge-/Werkzeugkombinationen in unterschiedlichen Lehrformen“. In: (Aug. 2016).

[Mad94] Kim Halskov Madsen. „A guide to metaphorical design“. In: *Communications of the ACM* 37.12 (1994), S. 57–63.