Technology
Arts Sciences
TH Köln

Ad-Hoc Kollaboration auf Basis von Peer-to-Peer Browsertechnologien

Masterarbeit

ausgearbeitet von

David Bellingroth

zur Erlangung des akademischen Grades

Master of Science (M.Sc.)

vorgelegt an der

Technischen Hochschule Köln
Campus Gummersbach
Fakultät für Informatik und
Ingenieurwissenschaften

im Studiengang

Medieninformatik

Erster Prüfer: Prof. Dr. Kristian Fischer
Technische Hochschule Köln

Zweiter Prüfer: Prof. Dr. Christian Kohls
Technische Hochschule Köln

Gummersbach, im Oktober 2015

Adressen:

David Bellingroth
Am Heiligenstock 5
51645 Gummersbach
david.bellingroth@th-koeln.de

Prof. Dr. Kristian Fischer
Technische Hochschule Köln
Institut für Informatik
Steinmüllerallee 1
51643 Gummersbach
kristian.fischer@th-koeln.de

Prof. Dr. Christian Kohls
Technische Hochschule Köln
Institut für Informatik
Steinmüllerallee 1
51643 Gummersbach
christian.kohls@th-koeln.de

Kurzfassung

Bei der Zusammenarbeit zwischen Menschen kommt immer häufiger auch unterstützende Software zum Einsatz. Diese Kollaborationswerkzeuge werden oftmals in Form von Webapplikationen im Browser realisiert. Webapplikationen bieten entscheidende Vorteile, darunter eine einfache Einrichtung und kaum notwendige Konfiguration. Sie müssen nicht fest auf dem System eines Nutzers installiert werden und haben daher eine niedrigere Einstiegshürde, was vor allem bei spontaner Zusammenarbeit wichtig ist. Gleichzeitig stehen viele Menschen Webapplikationen kritisch gegenüber, vor allem weil bei ihrer Nutzung Daten auf zentralen Servern verarbeitet und gespeichert werden, die in der Regel nicht unter der Kontrolle des Nutzers stehen.

Neue Webtechnologien, wie **WebRTC** und **IndexedDB**, ermöglichen es Webapplikationen, Daten über eine direkte Verbindung untereinander auszutauschen und lokal zu speichern. Daraus ergibt sich das Potential, die Abhängigkeit von zentralen Servern zu reduzieren und somit einige Nachteile von Webapplikationen aufzuheben. Die vorliegende Arbeit untersucht die Machbarkeit von Kollaborationsanwendungen für die lokale Zusammenarbeit auf der Grundlage dieser neuen Webtechnologien. Basierend auf vorher in einer Kontextanalyse erhobenen Kriterien, wird eine Softwarearchitektur für eine Kollaborationsplattform im Webbrowser entworfen, die in weiten Teilen auf zentrale Infrastruktur verzichten kann. Anhand einer prototypischen Implementierung dieser Architektur wird ein Nachweis für die generelle Machbarkeit des Konzepts erbracht und es werden Herausforderungen bei dessen Realisierung identifiziert.

Inhaltsverzeichnis

Abbildungsverzeichnis

Begriffe und Abkürzungen

Awareness Nach der Definition von Dourish und Bellotti ist Awareness „ein Verständnis der Handlungen anderer, welches einen Kontext für die eigenen Aktivitäten liefert“ [DB92].

CAD Computer-Aided Design

DHT Eine Distributed Hash Table ist eine mit einer Hashtabelle vergleichbare Datenstruktur, bei der die Datensätze auf mehrere physikalische Knoten verteilt werden. Jeder Knoten erhält eine eindeutige Kennung innerhalb eines festgelegten Adressraums. Der Hash determiniert, auf welchem Knoten der zugehörige Wert gespeichert wird. Über einen speziellen Routing-Algorithmus kann jeder Knoten jeden anderen Knoten mit einer garantierten maximalen Anzahl von Hops (in der Regel $O(\log n)$) erreichen. Distributed Hash Tables finden vor allem in Peer-to-Peer-Netzwerken mit vielen Teilnehmern, wie dem BitTorrent-Netzwerk, Verwendung.

DOM Document Object Model

ICE Interactive Connectivity Establishment

MVC Model View Controller

MVP Model View Presenter

NAT Network Adress Translation

OT Operational Transformation

RPC Remote Procedure Call

SCTP Stream Control Transmission Protocol

SDP Session Description Protocol

SIP Das Session Initiation Protocol ist ein Protokoll zur Aushandlung und Verwaltung von Kommunikationssitzungen zwischen einer oder mehreren Parteien [vgl. Ros+02] und dient unter anderem als Basis für viele Voice over IP Lösungen.

STUN Session Traversal Utilities for NAT

SVG Scalable Vector Graphics

WYSIWIS What You See Is What I See

1 Einleitung

1.1 Motivation

Die eng verzahnte Kollaboration und Kommunikation zwischen unterschiedlichen Personen ist in der heutigen Welt mehr denn je ein entscheidender Erfolgsfaktor, sei es in professionellen Kontexten, in Forschung und Lehre oder auch im privaten Bereich. Schon früh hat die Forschung das Potential von computergestützten Systemen erkannt, menschliche Zusammenarbeit zu unterstützen und bereichern. Auch ermöglichen vernetzte Systeme völlig neue Formen der Zusammenarbeit. Sie erlauben es Menschen, unabhängig von Zeit und Ort, gemeinsam an den gleichen Artefakten zu arbeiten.

Es existieren verschiedene Formen von Kollaborationslösungen. Ellis, Gibbs und Rein stellten 1991 eine zweidimensionale Matrix vor, welche Groupwaresysteme anhand des Grades ihrer räumlichen Verteilung und zeitlichen Asynchronität kategorisiert [vgl. EGR91]. Diese ist in Abbildung 1.1 dargestellt. Ein Großteil der Forschung im Bereich der Computer Supported Collaborative Work (CSCW) bezieht sich auf Systeme die die Nutzer räumlich oder zeitlich voneinander entkoppeln und somit den rot eingefärbten Quadranten zuzuordnen währen.

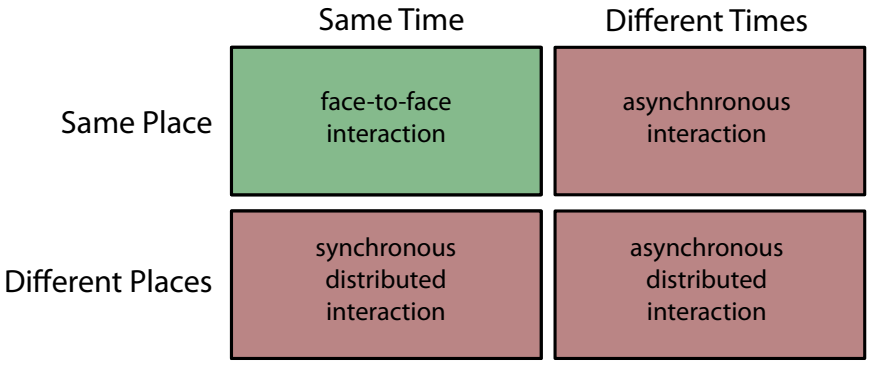

Abbildung 1.1: Kategorisierung von Groupwaresystemen [mod. nach EGR91, S.41]

Auch wenn asynchrone Zusammenarbeit immer relevanter wird, hat die klassische Face-to-Face-Kollaboration wenig an Wichtigkeit eingebüßt. In vielen Fällen bietet ein persönliches Treffen zwischen den Beteiligten Vorteile. Die hierbei mögliche Kombination aus verbaler und nonverbaler Kommunikation ist reichhaltiger und beugt Missverständnissen vor. Die Interaktion der Teilnehmer ist direkter und unmittelbarer. Allerdings können auch bei persönlicher Anwesenheit Softwaresysteme die Zusammenarbeit bereichern.

Groupwaresysteme, die diese Art der Zusammenarbeit adressieren, werden dem in Abbildung 1.1 grün eingefärbten Quadranten zugeordnet.

Face-to-Face-Kollaborationssituationen können sehr kurzfristig auftreten, zum Beispiel indem aus einer zufälligen Begegnung eine spontane Zusammenarbeit entsteht. Die beteiligten Personen können auch aus verschiedenen Kontexten, beispielsweise unterschiedlichen Unternehmen, stammen. Aus diesen Faktoren ergeben sich Herausforderungen. Klassische Softwarelösungen setzen in der Regel eine bestimmte Infrastruktur voraus, die bei spontaner Zusammenarbeit oft nicht vorhanden ist. Dieses Problem kann und wird durch die Verwendung von Peer-to-Peer-Systemen umgangen, die eine direkte Kommunikation der Endgeräte untereinander ermöglichen [vgl. z.B. KGB02; RPR06; WSF05; Ney+09]. Aber auch hier wird das Vorhandensein oder die Installation einer bestimmten Software auf allen Endgeräten vorausgesetzt. Die Notwendigkeit der Installation von Software auf dem eigenen Endgerät kann gerade in spontanen Kollaborationssituationen bereits eine Hürde darstellen, welche den Einsatz einer solchen verhindert.

In solchen Fällen bieten im Browser ausgeführte Webapplikationen Vorteile, denn für ihre Nutzung ist keine Installation notwendig, ein Webbrowser ist in den meisten Fällen vorhanden und die entsprechende Infrastruktur wird von außerhalb gestellt. Letzteres kann allerdings wiederum zu Problemen führen, denn viele potentielle Nutzer stehen „Cloud-Anwendungen“ aus Datenschutzbedenken heraus kritisch gegenüber, da hier ein Drittanbieter Kontrolle über Daten und Kommunikation hat. In vielen Fällen hat dieser Drittanbieter seinen Sitz im Ausland, so dass das lokale Datenschutzrecht nicht oder nur teilweise greift. Bei der Kollaboration im professionellen Kontext werden oftmals kritische Unternehmensinterna ausgetauscht und im Bereich der Lehre fallen personenbezogene Daten an. Dabei handelt es sich um Informationen, die nur ungern aus der Hand gegeben werden. Aber auch viele private Nutzer nutzen „Cloud-Anwendungen“ aus ähnlichen Motiven nur ungern.

Angesichts der oben betrachteten Aspekte, stellt sich die Frage ob es möglich ist, die Vorteile von Webapplikationen mit denen nativer Peer-to-Peer-Kollaborationslösungen zu kombinieren. Seit einiger Zeit halten Peer-to-Peer-Technologien Einzug in verschiedenen Webbrowsern. Der hierzu geschaffene Standard WebRTC ist vor allem für die Audio- und Video-Kommunikation konzipiert, ermöglicht aber in neueren Versionen auch den Austausch beliebiger Daten über eine direkte Verbindung zweier Instanzen einer Webapplikation. Außerdem existieren seit längerem Möglichkeiten, mit denen eine Webapplikation Daten lokal speichern kann. Es liegt nahe, solche Technologien auch für Kollaborationslösungen einzusetzen, die weitestgehend auf zentrale Infrastruktur für Datenaustausch und -speicherung verzichten können. Inwiefern dies umsetzbar und praktikabel ist, soll in der vorliegenden Arbeit untersucht werden. Um die Wiederverwendbarkeit der

Ergebnisse und Erkenntnisse zu gewährleisten, liegt der Fokus nicht auf einer einzelnen Kollaborationsanwendung für einen speziellen Anwendungsfall sondern auf Kollaborationsplattformen, die wiederum eine Grundlage für spezielle Anwendungen bieten können.

1.2 Ziele & Aufbau

Basierend auf den Überlegungen, die in Abschnitt 1.1 beschrieben wurden, lautet die dieser Arbeit zugrundeliegende Forschungsfrage folgendermaßen:

Ist mit heute verfügbaren Technologien die Umsetzung einer browserbasierten Plattform zur Unterstützung von Kooperationssituationen mit persönlicher Anwesenheit möglich, die bei weitgehendem Verzicht auf zentrale Infrastruktur, Installation und komplexe Initiationsschritte einen mit bestehenden Lösungen vergleichbaren Funktionsumfang bietet?

Um die Beantwortung der Forschungsfrage zu ermöglichen, ist eine Reihe Aktivitäten notwendig. Die Ziele der Arbeit werden, in Form von zu erstellenden Artefakten und Ergebnissen, in der nachfolgenden Liste aufgeführt:

- Eine Analyse bestehender Lösungsansätze
- Eine Analyse des soziotechnischen Kontextes in Face-to-Face-Kollaborationssituationen
- Die Ermittlung generischer High-Level-Anforderungen an entsprechende Kollaborationsplattformen
- Eine Analyse verfügbarer Browsertechnologien
- Ein Architekturentwurf für eine browserbasierte Ad-Hoc-Kollaborationsplattform
- Eine prototypische Umsetzung der Architektur in einem Kollaborationstool
- Eine Evaluation des Prototypen
- Eine Bewertung des Ansatzes auf Basis der gewonnenen Erkenntnisse

Aus den Zielen ergibt sich bereits ein grober Rahmen für die Struktur der Arbeit. Um einen Überblick zu liefern, sollen Inhalt und Aufbau der folgenden Kapitel an dieser Stelle kurz angerissen werden.

In Kapitel 2 werden Forschungsergebnisse und bestehende Softwarelösungen vorgestellt, die mit dem gewählten Themengebiet in Zusammenhang stehen. Kapitel 3 widmet sich der Analyse des sozialen und technischen Kontextes, in dem spontane Kollaboration stattfindet. Basierend darauf beschreibt Kapitel 4 den Entwurf einer Softwarearchitektur für eine Ad-Hoc-Kollaborationsplattform. Die Architektur wird in einem Prototypen umgesetzt, dessen Entwicklung in Kapitel 5 beschrieben wird. Die Architektur und der Prototyp werden in Kapitel 6 einer Evaluation unterzogen und kritisch reflektiert. Kapitel 7 bewertet die im Laufe der Arbeit gewonnenen Erkenntnisse, zieht Fazit und gibt einen Ausblick auf Themengebiete, die für zukünftige Forschungsarbeiten relevant sein können.

2 Related Works

Wie bereits in Abschnitt 1.1 angesprochen, beschäftigt sich ein Großteil der Forschung im Bereich CSCW mit zeitlich oder räumlich entkoppelter Zusammenarbeit. Trotzdem spielt auch die Unterstützung von Face-to-Face-Kollaboration durch Software eine nicht zu vernachlässigende Rolle. Im Folgenden werden einige Ansätze und Erkenntnisse aus der Forschung in diesem Bereich sowie bestehende Lösungen, die sich im praktischen Einsatz befinden, vorgestellt.

2.1 Forschung im Bereich Face-to-Face-Kollaboration

In diesem Abschnitt werden beispielhaft einige Studien kurz angerissen, die sich mit dem Einsatz von Software zur Unterstützung von Kollaboration im Face-to-Face-Kontext auseinandersetzen. Ziel ist es einerseits, die Relevanz dieses Themenkomplexes zu unterstreichen und andererseits, Anregungen für die eigene Arbeit zu erhalten. Die Betrachtung beschränkt sich jeweils auf die im Rahmen dieser Arbeit als relevant erachteten Aspekte.

2.1.1 Studie von Horton u. a.

Horton u. a. gingen 1991 der Frage nach, inwiefern der Einsatz von unterstützender Technologie die Face-to-Face-Zusammenarbeit von Gruppen beeinflusst. Dabei legten sie den Fokus auf kollaboratives Schreiben [Hor+91].

Durchführung der Studie

Verglichen wurden Gruppen, die jeweils ein Dokument ohne Softwareunterstützung und eines mit Softwareunterstützung erarbeiten sollten. Während im ersten Fall Papier und Stift sowie ein Flipchart verwendet werden durften, standen den Teilnehmern im zweiten Fall jeweils ein eigener Computer mit einer Textverarbeitungssoftware zur Verfügung. Zudem verfügte der Raum über einen zentralen Computer mit einem großen Monitor, auf dem ebenfalls eine Textverarbeitungssoftware lief. Über eine spezielle Taste konnten die Teilnehmer Zugriff auf diesen Computer anfordern und diesen fernsteuern. Zudem ließen sich per Copy & Paste Textabschnitte zwischen dem zentralen und dem private Computer transferieren. Ein gleichzeitiger Zugriff auf den zentralen Computer durch mehrere Teilnehmer war nicht möglich [Hor+91].

Ergebnisse

Wie sich zeigte, hatte die Verwendung von technischen Hilfsmitteln einen signifikanten Einfluss auf den Arbeitsprozess der Gruppen. Stand keine Softwareunterstützung zur Verfügung, planten die Gruppen ihre Zusammenarbeit im Plenum, bevor sie mit der eigentlichen Schreibarbeit begannen. Unter der Verwendung der Software tendierten sie dazu, weniger Zeit in initiale Planung zu investieren und begannen stattdessen schneller damit, individuell Texte zu produzieren [Hor+91].

Generell stellten Horton u. a. fest, dass unter Verwendung der technischen Hilfsmittel der Anteil der individuellen Arbeit stark anstieg. Texte wurden einzeln erarbeitet und dann zusammengetragen. Dementsprechend stieg auch die Anzahl der, an den Texten vorgenommenen, Überarbeitungen. Ohne Technologie fand der eigentliche Schreibprozess meist in Gruppenarbeit und unter ständigem Austausch statt. Abbildung 2.1 stellt die Unterschiede der Gruppenprozesse dar, wobei T für die Verwendung von technischen Hilfsmitteln und NT für die Nicht-Verwendung von technischen Hilfsmitteln steht.

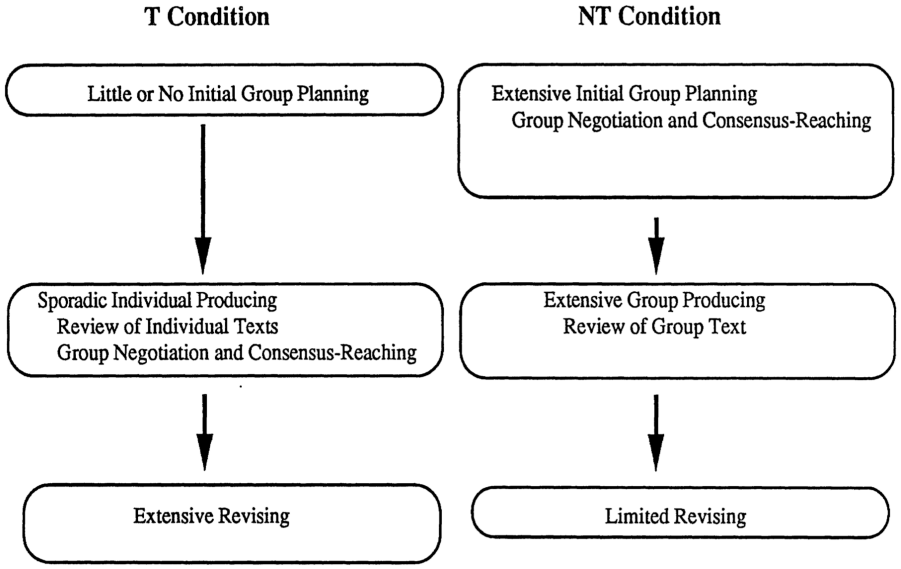

Abbildung 2.1: Unterschiedliche Vorgehensweisen abhängig von Technologieverwendung [Quelle: Hor+91]

Horton u. a. empfehlen für die Gruppenarbeit mit kollaborativer Software eine initiale Planungsphase. Die Verwendung der Technologie brachte im Schnitt keine besseren Ergebnisse hervor. Stattdessen erreichten vor allem nur diejenigen Gruppen, welche ihre Vorgehensweise planten und Aufgaben explizit auf verschiedene Teilnehmer aufteilten unter Verwendung der Technologie hochwertigere Ergebnisse, als mit Stift und Papier [Hor+91].

Erkenntnisse für die eigene Arbeit

Wie sich in dieser Studie zeigt, ist die Verbesserung der Zusammenarbeit durch die Verwendung von kollaborativer Software nicht trivial. Stattdessen gibt es einige Faktoren zu beachten. Ein Faktor, der sich bei Horton u. a. klar zeigt ist die Unterscheidung zwischen tatsächlicher Gruppenarbeit im Plenum und individueller Arbeit. Beide spielen bei der Zusammenarbeit eine wesentliche Rolle und hängen wiederum eng zusammen mit der Unterscheidung zwischen koordinatorischer Arbeit und der Arbeit an den eigentlichen Inhalten. Daher sollten sie auch in einer entsprechenden Softwarelösung berücksichtigt werden.

2.1.2 Studie von Olson u. a.

Olson u. a. führten 1993 eine Studie durch, um die Auswirkungen der Nutzung eines kollaborativen Texteditors auf Meetings in kleinen Gruppen zu untersuchen. Dabei sollte die Software explizit keine bestimmte Struktur oder Arbeitsmethodik vorgeben. Vielmehr vermuteten sie, dass eben diese Vorgabe bestimmter Vorgehensweisen durch eine Software bei anderen Studien zu Problemen geführt haben könnte [Ols+93].

Durchführung der Studie

Untersucht wurden insgesamt 38 Gruppen von jeweils 3 Personen. Sie bekamen ein Designproblem vorgelegt und hatten 90 Minuten Zeit für die Erarbeitung einer Lösung. Eine Hälfte der Gruppen setzte dabei einen kollaborativen Texteditor namens ShrEdit ein, während die andere Hälfte nur auf herkömmliche Hilfsmittel wie Papier, Stift und Whiteboard zurückgreifen durfte.

Untersucht wurde einseits das Ergebnis des Prozesses, das heißt die Qualität der erarbeiteten Lösungen, aber auch die Zufriedenheit der Teilnehmer und der Ablauf des Kollaborationsprozesses an sich [Ols+93].

Ergebnisse

Wie sich zeigte, war die durchschnittliche Qualität der Ergebnisse in denjenigen Gruppen, die den kollaborativen Editor verwendet hatten signifikant höher. Zudem war die Varianz der Qualität dort geringer, woraus sich schließen lässt, dass gerade schwächere Gruppen vom Einsatz der Software profitierten.

Die Zufriedenheit der Teilnehmer mit dem Ablauf der Zusammenarbeit, war bei denjenigen ohne Softwareunterstützung etwas höher. Auch betrachteten diese Gruppen, entgegen der Erwartung der Autoren, mehr Aspekte und Alternativen als diejenigen, die das Kollaborationstool einsetzten. Bei eingehender Untersuchung zeigte sich allerdings, dass es sich bei den zusätzlich betrachteten Aspekten oftmals nicht um Kernaspekte des Problems handelte. Eine mögliche Ursache für das bessere Ergebnis der Gruppen welche die Software verwendeten liegt daher darin, dass diese fokussierter auf das eigentliche Problem waren.

Obwohl die Software explizit kein bestimmtes Vorgehensmodell vorgab, hatte ihre Verwendung großen Einfluss auf den Ablauf der Zusammenarbeit. Die traditionell arbeitenden Gruppen sammelten Ideen oftmals in einem zweistufigen Prozess, bei dem Ideen erst einzeln generiert und dann in der Gruppe an einem Whiteboard zusammengetragen wurden. Dabei gingen Ideen verloren. In Gruppen mit Softwareunterstützung gab es keinen mehrstufigen Prozess. Stattdessen waren die Übergänge zwischen einzelnen Aktivitäten des kreativen Prozesses fließender. Ein bestimmtes Vorgehen etablierte durch die Verwendung der Software nicht. Olson u. a. beobachteten stattdessen viele unterschiedliche Arten der Zusammenarbeit [Ols+93].

Erkenntnisse für die eigene Arbeit

In der Studie von Olson u. a. zeigt sich, dass die Unterstützung durch kollaborative Software in Face-to-Face-Situationen durchaus von Vorteil sein kann. Die Flexibilität der Software, verschiedene Vorgehensweisen bei der Arbeit zu unterstützen, spielt hierbei eine Rolle für deren erfolgreichen Einsatz.

2.1.3 Synergieeffekte in kleinen Gruppen

Larson beschäftigt sich in Kapitel 3 seines Buches „In Search of Synergy in Small Group Performance“ mit Synergieeffekten bei der Ideengenerierung in Gruppen. Betrachtet wird hier die klassische Brainstorming-Technik [Lar10].

Larson führt an, dass bereits frühe Studien belegen, dass sich die gewünschten Synergieeffekte beim Brainstorming nicht einstellen [vgl. bspw. TBB58]. Stattdessen hat die Gruppensituation in vielen Fällen eher negative Auswirkungen. Grund hierfür sind eine Reihe von Effekten, welche die Produktivität der einzelnen Gruppenmitglieder reduzieren. Diese Effekte werden im Folgenden Aufgeführt:

Production Blocking Da in einer Gruppensituation nur jeweils eine Person gleichzeitig sprechen kann, sind die Teilnehmer simultan damit beschäftigt, jemand anderem zuzuhören und über dessen Ideen nachzudenken, eigene Ideen zu generieren und sich diese zu merken sowie währenddessen auf Indikatoren für eine Gelegenheit achten, selbst das Wort zu ergreifen. Dies führt einerseits zu einem Zeitverlust, andererseits aber auch zu einer hohen kognitiven Belastung und einem möglichen Vergessen eigener Gedanken und Ideen [Lar10].

Angst vor Bewertung In einer Gruppensituation unterliegt man einer ständigen Beobachtung durch andere Personen. Daraus kann eine Angst vor negativer Bewertung der eigenen Äußerungen entstehen, auch wenn diese nicht explizit geäußert wird [Lar10].

Trittbrettfahrer-Effekt Beim klassischen Brainstorming wird nicht festgehalten, wer welchen Beitrag zum Endergebnis geleistet hat. Hieraus folgt in manchen Fällen eine verminderte Motivation, da die eigene Leistung nur einen geringen persönlichen Vorteil für das Individuum zur Folge hat [Lar10].

Gefühl der Entbehrlichkeit Je größer eine Gruppe ist, desto mehr kann bei Einzelnen das Gefühl der Entbehrlichkeit entstehen. Hat eine Person den Eindruck, keinen substanziellen Beitrag leisten zu können, hält sie sich in der Gruppenarbeit zurück [Lar10].

Anpassung an die Gruppenleistung Arbeiten Personen über längere Zeiträume zusammen, so lernen sie die Leistungsfähigkeit der anderen besser einzuschätzen. Stärkere Gruppenmitglieder passen sich den schwächeren an. Damit vermeiden sie das Gefühl, von den anderen ausgenutzt zu werden oder die Arbeit alleine machen zu müssen [Lar10].

Aufgrund der genannten Effekte ist ein individuelles Brainstorming der einzelnen Teilnehmer in vielen Fällen produktiver als ein Brainstorming in der Gruppe. Larson führt zudem alternative Kreativitätstechniken auf, die negative Gruppeneffekte in Teilen kompensieren können. In größeren Gruppen bietet sich laut Larson ein softwaregestütztes Brainstorming an, bei dem jeder Teilnehmer an einem eigenen Computer sitzt. Neben einem gemeinsamen Bereich zur Anzeige der Ideen, gibt es einen getrennten Bereich zur Ideeneingabe, der für andere nicht sichtbar ist. Diese Trennung beugt dem Effekt des *Production Blocking* vor [Lar10].

Erkenntnisse für die eigene Arbeit

In Face-to-Face-Kollaborationssituationen kann eine Reihe von negativen Effekten auftreten. Eine Softwarelösung kann helfen, die Effektivität und Effizienz der Gruppenarbeit zu steigern. Allerdings muss sie die genannten Effekte hierfür auch berücksichtigen. Eine Konsequenz, die auch schon in Abschnitt 2.1.1 angesprochen wurde, ist die Sinnhaftigkeit einer Trennung zwischen individueller Arbeit und der gemeinsamen Arbeit an Artefakten, um das Problem des *Production Blocking* zu mindern.

2.2 Bestehende Lösungsansätze

Während der Recherche konnte bisher kein Ansatz ausfindig gemacht werden, der ebenfalls eine Kollaborationslösung auf Basis von Peer-to-Peer-Webtechnologien umsetzt. Dennoch gibt es viele Ansätze, die eine gewisse Schnittmenge mit diesem Thema aufweisen. An dieser Stelle sollen einige bestehende Lösungsansätze erläutert und analysiert werden, die ähnliche Ziele verfolgen, wie die geplante Softwarearchitektur. Das geschieht einerseits, um einen Einblick in das Feld der Peer-to-Peer-Kollaboration zu gewinnen und andererseits in der Hoffnung auf wichtige Impulse für die Gestaltung einer eigenen Architektur.

2.2.1 Li u. a.

Li u. a. stellten 2004 eine Architektur für ein verteiltes Editing-System vor, die auf einer Kombination aus einer zentralen Komponente und der Verwendung von Peer-to-Peer-Kommunikation basiert. Dabei wird besonderer Wert auf eine zuverlässige Kommunikation gelegt [Li+04]. Das System ist jedoch nicht primär auf Face-to-Face-Kollaboration ausgelegt und sieht für die Kommunikation der einzelnen Komponenten eine Internetverbindung vor.

Architektur

Abbildung 2.2 stellt die Architektur grafisch dar.

Während die eigentliche Kommunikation der Clients untereinander mittels eines unstrukturierten Peer-to-Peer-Netzwerks (vgl. Abschnitt 4.4.1) abläuft, argumentieren Li u. a., dass sich für die Umsetzung bestimmter Funktionalitäten eine zentrale Instanz anbietet.

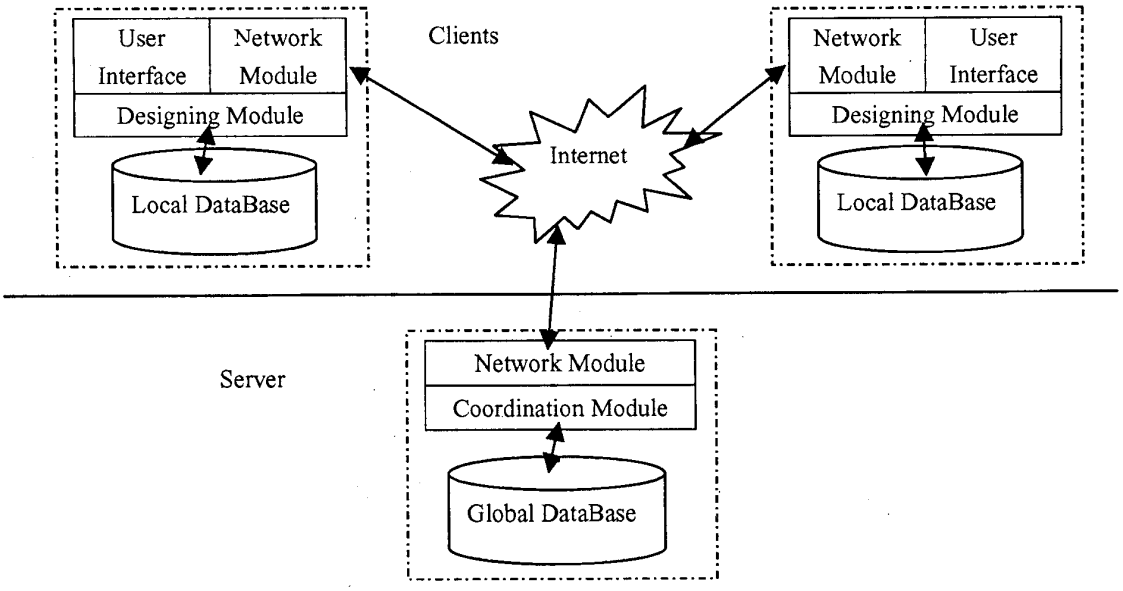

Abbildung 2.2: Systemarchitektur von Li u. a. [Quelle: Li+04, S. 245]

Daher enthält ihre Architektur einen Server, welcher sich unter anderem um das Management von Konferenzen kümmert und die Konsistenz der bearbeiteten Inhalte gewährleisten soll.

Jeder Client besitzt eine eigene lokale Datenbank, in welcher er den aktuellen Arbeitsstand persistent festhalten kann. Userinterface und Netzwerkkommunikation sind in eigene Module ausgelagert, welche von einer dritten Komponente, dem „Designing Module", kontrolliert werden, das den Großteil der notwendigen Logik umsetzt [Li+04, S. 245f].

Verfügbarkeit

Li u. a. gehen davon aus, dass jederzeit Knoten dem Peer-to-Peer-Netzwerk beitreten oder es verlassen können. Um die Verfügbarkeit der Verbindungen zu gewährleisten, sendet jeder Knoten des Netzwerks jedem anderen Knoten in regelmäßigen Abständen eine spezielle Nachricht, deren Empfang durch eine entsprechende Antwortnachricht quittiert wird. Trifft nach überschreiten einer definierten Zeitspanne keine Antwort ein, wird der verwendete Routingpfad als ungültig markiert und ein alternativer Pfad verwendet [Li+04, S. 246].

Multicast

Soll eine Nachricht an mehrere andere Knoten weitergeleitet werden (Multicast), so sendet der Client die Nachricht nicht an jeden einzelnen Knoten. Stattdessen setzen Li u. a. auf einen effizienteren Multicast-Algorithmus. Hierbei wird die Nachricht nur an Knoten versandt, zu denen der Ausgangsknoten eine direkte Verbindung besitzt. Dieser leitet die Nachricht nach dem selben Muster weiter. Eine Liste der Knoten, welche die Nachricht erhalten sollten, wird in die Nachricht integriert [Li+04, S. 246f].

Nebenläufigkeit und Konflikte

Da mehrere Personen gleichzeitig an den gleichen Daten arbeiten und diese Änderungen über Netzwerkverbindungen ausgetauscht werden, die unzuverlässig sein können, besteht die Gefahr, dass Inkonsistenzen entstehen und somit verschiedene Clients verschiedene Versionen der Daten besitzen. Um diesem Problem entgegenzuwirken, verwenden Li u. a. einen Algorithmus von Ellis und Gibbs [EG89]. Dieser basiert auf einer Datenstruktur, genannt „State-Vector“, die jeder Client lokal vorhält. In dieser wird für jeden Knoten im Peer-to-Peer-Netzwerks ein Zähler gespeichert, der inkrementiert wird, sobald eine Änderung der Daten von diesem Knoten ausgeht. Beim Versand einer Änderung sendet der Ausgangsknoten immer den lokalen Stand seines Zählers mit. Stellt ein anderer Client eine Diskrepanz zwischen seinem eigenen und dem in der Nachricht enthaltenen Zähler fest, deutet das auf den Verlust einer Nachricht hin. Der Client kann dann die verpassten Änderungen vom Ausgangsknoten explizit neu anfordern [Li+04, S. 246].

Erkenntnisse für die eigene Arbeit

Die von Li u. a. vorgestellte Architektur könnte für den Anwendungsfall dieser Arbeit vor allem aus Sicht der Kommunikation interessant sein. Sie schlagen Lösungen für einige diesbezügliche Probleme vor, darunter die Erkennung von und der Umgang mit Verbindungsabbrüchen, die Umsetzung von Multicast-Mechanismen und die Gewährleistung einer konsistenten Datenbasis.

2.2.2 GroupKit

Roseman und Greenberg entwickelten 1992 ein Toolkit namens „Groupkit“. GroupKit soll Entwickler dabei unterstützen, Anwendungen für verteilte, aber auch Face-to-Face Meetings zu entwickeln. Der adressierte Anwendungsfall ist die Implementierung von

geteilten Arbeitsflächen nach dem „What You See Is What I See (WYSIWIS)"-Prinzip. Hierbei arbeiten alle Teilnehmer einer Sitzung gemeinsam an Artefakten. Änderungen die ein Teilnehmer an den Artefakten vornimmt sind instantan auch für andere Teilnehmer sichtbar [RG92].

Anforderungen

Roseman und Greenberg identifizieren verschiedene Anforderungen an ihr Toolkit. Einige davon, die für dieses Projekt ebenfalls relevant sein können, sollen im Folgenden kurz erläutert werden [vgl. RG92]:

Unterstützung verschiedener Interaktionsparadigmen Roseman und Greenberg nennen explizit die Verwendung von Gesten im User Interface sowie die Möglichkeit der grafischen Annotation von Artefakten [RG92]. Bei beidem ist Interaktion der Teilnehmer in Echtzeit von Vorteil.

Integration mit bestehenden Workflows Die Groupware sollte sich in bestehende Vorgehensweisen integrieren und diese unterstützen. Roseman und Greenberg nennen hier beispielhaft die Bereitstellung eines Audiokanals oder die Integration von Single-User-Anwendungen über einen geteilten Bildschirminhalt [RG92].

Eine robuste Kommunikationsinfrastruktur Ein Groupwaretoolkit sollte laut Roseman und Greenberg mindestens die Möglichkeit bieten, Nachrichten an andere Prozesse zu verschicken und nach Möglichkeit auch einen Multicast oder Broadcast ermöglichen [RG92].

Persistente Sessions Da Zusammenarbeit sich oft über mehrere Sitzungen erstreckt, sollten Informationen persistent gespeichert werden können [RG92].

Gleichzeitige Bearbeitung Wenn mehrere Teilnehmer mit denselben Artefakten arbeiten, ist es in vielen Fällen notwendig, die parallele Bearbeitung zu koordinieren, sei es durch einfaches Locking oder komplexere Mechanismen [RG92].

Trennung von Darstellung und Daten Viele Anwendungen mit grafischer Benutzungsschnittstelle trennen die Darstellung von der zugrunde liegenden Datenrepresentation. Roseman und Greenberg argumentieren, dass nach Patterson diese Trennung in Groupwaresystemen noch wesentlicher ist [RG92]. Das ermöglicht es auch, dem Nutzer verschiedene Perspektiven auf die gleichen Daten anzubieten [Pat91].

Die Architektur und Kommunikationsinfrastruktur von GroupKit ist in Abbildung 2.3 abgebildet. Sie umfasst eine zentrale Komponente, den sogenannten Registrar, und mehrere Clients. GroupKit vereint Merkmale zentraler und dezentraler Architekturen.

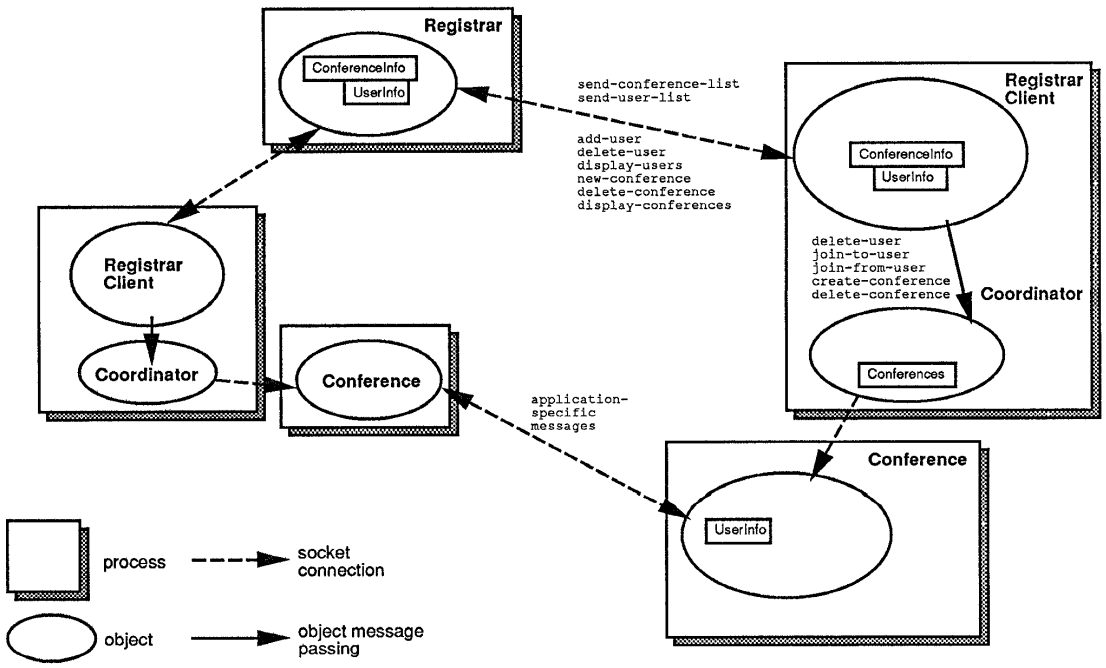

Abbildung 2.3: Architektur von Groupkit [Quelle: RG92, S. 46]

Der Registrar dient der zentralen Verwaltung von Informationen zu Nutzern und Konferenzen. Er stellt anderen Komponenten Funktionalitäten bereit um diese Informationen abzurufen und zu manipulieren. Somit dient er als Vermittlungsstelle für die eigentlichen Groupwareanwendungen [RG92].

Die Anwendung des Nutzers besteht aus verschiedenen Komponenten. Der Registrar-Client dient der Kommunikation mit dem Registrar. Er ermöglicht es dem Nutzer, sich beim Registrar anzumelden, neue Konferenzen anzulegen oder bestehenden beizutreten. Tritt ein Nutzer einer Konferenz bei, so teilt der Registrar-Client dies dem Coordinator mit. Der Coordinator startet einen neuen Prozess, welcher ein Conference-Objekt instanziiert. Das Conference-Objekt stellt die eigentliche Groupwareanwendung dar, die über eine direkte Verbindung mit den Conference-Objekten anderer Nutzer applikationsspezifische Nachrichten austauscht [RG92].

Ein Großteil der in Abbildung 2.3 dargestellten Komponenten sind wiederverwendbar und hängen nicht oder nur bedingt von der Art der Groupware ab. Nur das Conference-

Objekt muss je nach Anwendungsfall neu implementiert werden, da es die eigentliche Logik der Groupwareanwendung umsetzt.

Erkenntnisse für die eigene Arbeit

Roseman und Greenberg arbeiten einige Anforderungen heraus, die sich auch auf eine browserbasierte Groupware für Ad-Hoc-Kollaboration anwenden lassen. Relevant ist in diesem Kontext unter anderem die strikte Trennung von Darstellung und Datenrepräsentation und die Persistenz von Sitzungen. Die Architektur selbst könnte insofern eine Rolle spielen, als das auch in einer Webapplikation höchstwahrscheinlich nicht ganz auf eine zentrale Komponente verzichtet werden kann. Somit kommt hier ebenfalls eine Mischform aus einer zentral und dezentral organisierten Architektur in Frage.

2.2.3 SOMU

Neyem u. a. stellen in ihrem Paper „„An Architectural Pattern for Mobile Groupware Platforms“ ein Architekturmuster vor, das auf mobile Groupwarelösungen ausgelegt ist. Zusätzlich präsentieren sie eine Implementierung dieses Architekturmusters namens „Service-Oriented Mobile Unit“ (SOMU). Dabei legen sie folgende Anforderungen zugrunde, die ein entsprechendes System berücksichtigen sollte [Ney+09]:

- Flexibilität im Bezug auf Änderungen von Gruppengröße und Struktur
- Konsistenz und Verfügbarkeit geteilter Informationen
- Awareness bezüglich der Arbeit der Gruppe
- Informationssicherheit und Datenschutz
- Integration und Interoperabilität im Bezug auf unterschiedliche Plattformen
- Zuverlässige synchrone und asynchrone Kommunikation
- Unabhängigkeit von bestehender Netzwerkinfrastruktur

Architektur

Das von Neyem u. a. vorgeschlagene Architekturmuster ist in Abbildung 2.4 grob abgebildet. Es basiert auf einem dreigeteilten Schichtenmodell. Die Autoren argumentie-

ren, dass eine Schichtenarchitektur die Skalierbarkeit, Wartbarkeit und Adaptierbarkeit des Systems erhöht. Die Schichten sind hierarchisch organisiert, so dass nur benachbarte Schichten miteinander kommunizieren können [Ney+09].

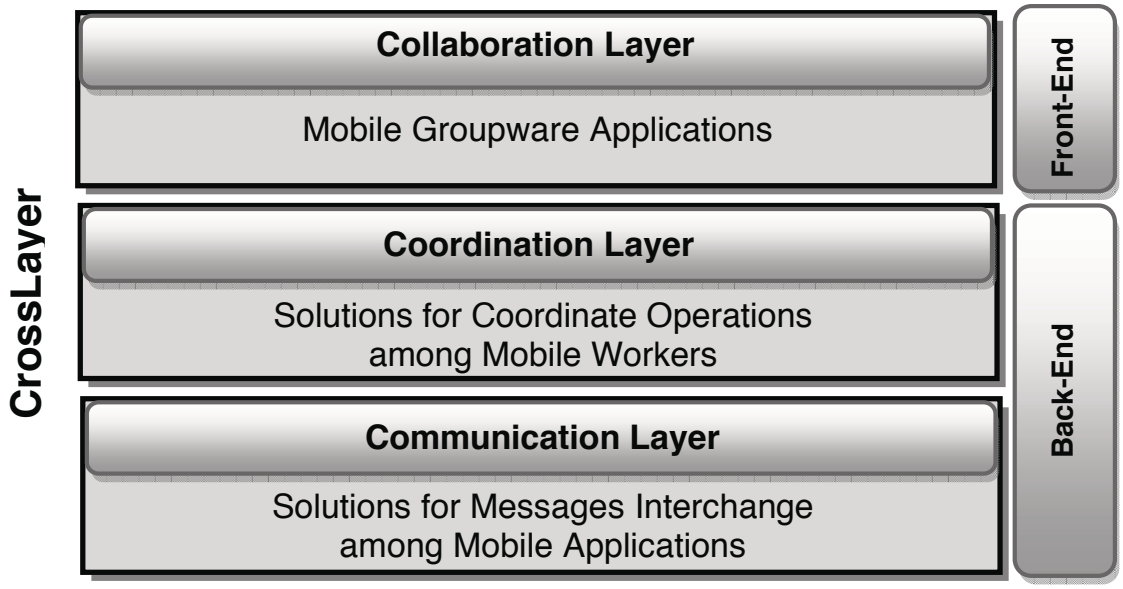

Abbildung 2.4: Schichtenarchitektur nach Neyem u. a. [Quelle: Ney+09, S. 403]

Der **Communication Layer** ist die unterste Schicht und zuständig für die eigentliche Kommunikation. Er ermöglicht den Austausch von Nachrichten über Netzwerkverbindungen. Zu seinen Aufgaben zählt unter anderem das Routing von Nachrichten in einer Ad-Hoc-Netzwerkumgebung. Außerdem stellt er anderen Komponenten die Möglichkeit zur Verfügung, Services zu implementieren, die auf einem Remote Procedure Call (RPC) ähnlichen Prinzip beruhen [Ney+09].

Der **Coordination Layer** greift auf den Communication Layer zurück und stellt Dienste zur Verfügung, die von der eigentlichen Kollaborationsanwendung genutzt werden. Neyem u. a. teilen diese Schicht wiederum in drei Komponenten auf. Eine Komponente übernimmt das Management von Sitzungen, Nutzern und Rollen. Die zweite Komponente verwaltet geteilte und private Ressourcen. Eine dritte Komponente, genannt **Context Management**, ist zuständig für Funktionen, die Kontextfaktoren betreffen, welche das Verhalten der mobilen Groupware beeinflussen können. Dazu gehören unter anderem Hardwareressourcen der verwendeten Endgeräte oder die Qualität von Netzwerkverbindungen [Ney+09].

Der **Collaboration Layer** stellt die eigentliche Groupware-Anwendung dar. Er ist dementsprechend unter anderem für die Bereitstellung eines User Interface zuständig. Er greift auf die vom Coordination Layer zur Verfügung gestellte Funktionalität zurück [Ney+09].

Erkenntnisse für die eigene Arbeit

Neyem u. a. gehen in ihrem Paper auf generelle Anforderungen an mobile Groupwaresysteme ein, die auch für für die hier vorliegende Arbeit von Bedeutung sind. Die vorgestellte Architektur orientiert sich stark an klassischen objektorientierten Konzepten und eine Umsetzung im Browser wäre suboptimal. Trotzdem liefert das Modell Anregungen dafür, wie die Komponenten einer dezentralen Groupwareplattform strukturiert werden können, um eine hohe Skalierbarkeit und Wiederverwendbarkeit zu ermöglichen.

2.2.4 Saucedo-Tejada und Mendoza

Saucedo-Tejada und Mendoza schlagen eine Architektur für ein Toolkit für mobile Groupwaresysteme vor. Dabei versuchen sie einerseits auf spezifische Probleme mobiler Plattformen, vor allem Smartphones einzugehen. Andererseits adressieren sie eine Reihe von allgemeineren Fragen, die auch für browserbasierte Plattformen von Interesse sind.

Saucedo-Tejada und Mendoza argumentieren, dass mobile Groupwaresysteme in verschiedenen Situationen von Vorteil sein können. Dazu zählt spontane ungeplante Kollaboration, bei der die Verwendung einer Softwarelösung die Koordination der Zusammenarbeit, die Awareness bezüglich der Handlungen anderer und die Verteilung der Ergebnisse unter den Teilnehmern vereinfachen bzw. verbessern kann [SM11]. Daher legen sie den Fokus ihrer Architektur auf Smartphoneanwendungen.

Architektur

Die vorgeschlagene Architektur des Toolkits ist vollständig dezentral. Für die Kommunikation kommen hauptsächlich Bluetooth-Direktverbindungen zum Einsatz. Es ist keine zentrale Infrastruktur vorgesehen und alle Knoten innerhalb des Netzwerks sind gleichberechtigt. Die einzelnen Systemkomponenten des Toolkits sind in Abbildung 2.5 abgebildet.

Die eigentliche Groupwareanwendung kann über eine Reihe von APIs auf die Funktionalität des Toolkits zugreifen. Hierzu gehört die Interaktion mit sogenannten „Shared Objects“, welche die Daten repräsentieren, mit denen die Nutzer gemeinsam arbeiten. Eine

eigene Softwarekomponente kümmert sich darum, diese Daten auf allen teilnehmenden Systemen konsistent zu halten. Das Toolkit besitzt eine Komponente zur Verwaltung von Nutzerdaten. Anwendungen können zudem über einen generischen Event-Mechanismus miteinander kommunizieren. All diese Subsysteme greifen intern auf eine Komponente zu, die sich um das Management von Ad-Hoc-Netzwerken und das Routing von Nachrichten kümmert [SM11].

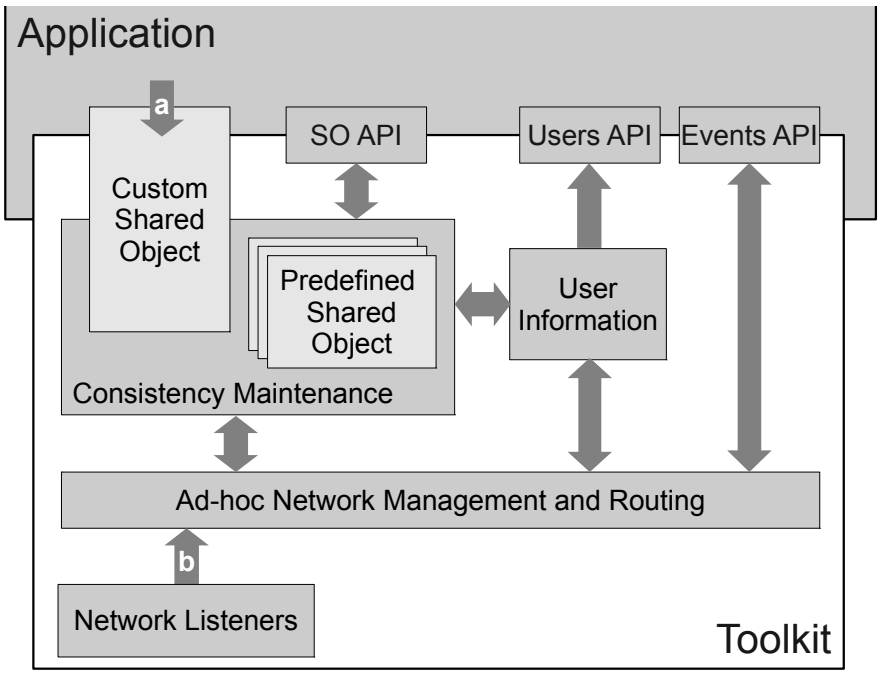

Abbildung 2.5: Architektur des Toolkits von Saucedo-Tejada und Mendoza [SM11]

Konsistenzmanagement

Ein wesentlicher Aspekt der Architektur von Saucedo-Tejada und Mendoza ist das Konsistenzmanagement beziehungsweise der Umgang mit parallelen Bearbeitungen derselben Daten. Wie in Abbildung 2.5 dargestellt, ist hierfür eine eigene Komponente vorgesehen.

Saucedo-Tejada und Mendoza plädieren dafür, dass jeder Knoten eine vollständige Kopie aller Daten besitzt. Dies ermöglicht es allen Nutzern, Daten lokal abzurufen, unabhängig von der Verfügbarkeit anderer Knoten. Zudem hat es einen positiven Einfluss auf die Reaktionsgeschwindigkeit der Anwendung [SM11].

Um den Nutzern eine zeitgleiche Bearbeitung von Inhalten zu ermöglichen, verwenden Saucedo-Tejada und Mendoza Operational Transformation (OT). Diese Technik kommt in vielen kollaborativen Softwarelösungen zum Einsatz und ist robust gegenüber Netzwerklatenzen oder sogar Konnektivitätsausfällen [SM11]. Ein Kernaspekt von OT be-

steht darin, dass alle Datenmanipulationen ausschließlich über sogenannte Operationen ausgeführt werden. Diese Operationen können über Parameter spezifiziert werden. Das Ergebnis einer Operation hängt dabei vom Ausführungskontext ab. Stößt ein Nutzer eine Änderung der Daten an, so wird die zugehörige Operation im lokalen Kontext augenblicklich ausgeführt und an andere Netzwerkknoten gesendet. Dabei kann es beispielsweise aufgrund von Netzwerklatenzen passieren, dass in der Zwischenzeit weitere Operationen auf den Daten ausgeführt wurden und sich somit der Ausgangszustand (Kontext) der Operation verändert hat. In diesem Fall muss die Operation transformiert werden, so dass sie auch im neuen Kontext zum gleichen Ergebnis führt [SM11]. Abbildung 2.6 veranschaulicht das Prinzip.

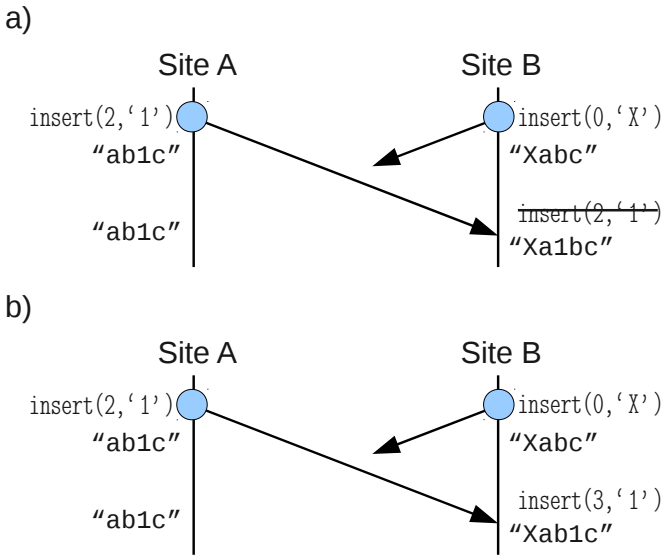

Abbildung 2.6: Operational Transformation [SM11]

Awareness

Saucedo-Tejada und Mendoza gehen davon aus, dass die Anforderungen an Awareness je nach Anwendungsfall stark variieren. Daher modellieren sie innerhalb des Toolkits keine Awarenessinformationen explizit. Stattdessen stellen sie einen generischen eventbasierten Kommunikationsmechanismus zur Verfügung, den die Groupwareanwendungen zum Austausch von Awarenessinformationen verwenden können [SM11].

Erkenntnisse für die eigene Arbeit

Die von Saucedo-Tejada und Mendoza vorgeschlagene Architektur gibt Anregungen dafür, wie mit der verteilten Datenbasis in Peer-to-Peer-Kollaborationsanwendungen umgegangen werden kann. Insbesondere wird deutlich, dass sich Techniken wie OT, die

ursprünglich für Szenarien mit zentraler Organisationseinheit ausgelegt sind, auch für dezentrale Architekturen adaptieren lassen. Ihr Vorschlag, Awarenessinformationen über einen generischen Eventmechanismus auszutauschen aufgrund seiner Abstraktion für verschiedene Kollaborationsanwendungen geeignet und könnte somit auch für eine browserbasierte Kollaborationsplattform von Nutzen sein.

3 Kontextanalyse

Um die Entwicklung einer für Face-to-Face-Kollaboration adäquaten Softwarearchitektur zu ermöglichen, wird in diesem Kapitel eine Analyse des sozialen und technischen Kontextes durchgeführt, in dem eine solche Lösung Anwendung finden könnte. Ziel der Analyse ist, neben einem grundlegenden Verständnis der Domäne, die Ermittlung essentieller Eigenschaften, die eine solche Architektur aufweisen muss, um Menschen bei ihren Aufgaben zu unterstützen und eine ausreichende Nutzungsmotivation zu schaffen.

3.1 Vorüberlegungen

Wie bei jeder Analyse, lagen auch dem im Folgenden beschriebenen Vorgehen bestimmte Vermutungen und grundlegende Überlegungen zu Grunde. Sie haben Einfluss auf die Erhebung und können die Ergebnisse der Analyse beeinflussen. Deshalb sollen sie an dieser Stelle der Vollständigkeit und Transparenz halber kurz beschrieben werden.

Rollen

Da es sich bei Face-to-Face-Kollaboration um eine komplexe Interaktion zwischen unterschiedlichen Menschen handelt, wurde vermutet, dass sich in vielen Fällen eine bestimmte Rollenstruktur existiert, sei es explizit oder implizit. Eine solche Rollenstruktur könnte sowohl positive als auch negative Konsequenzen auf die Zusammenarbeit haben. Beispielsweise könnte eine Spezialisierung einzelner Teilnehmer auf ihre jeweiligen Stärken die Effektivität und Effizienz der Gruppenarbeit verbessern. Andererseits könnte sich beispielsweise die Dominanz einzelner Teilnehmer negativ auf die Leistung der restlichen Teilnehmer auswirken. Solche Effekte sind unter anderem aus Untersuchungen von Brainstorming-Sitzungen bekannt [Lar10, S. 91 ff]. Die positiven und negativen Aspekte eine Rollenstruktur könnten durch eine Softwarelösung unterstützt oder teilweise kompensiert werden.

Initialisierung

Wie bereits in Abschnitt 1.1 angesprochen, können Face-to-Face-Kollaborationssituationen kurzfristig und spontan auftreten. Daher wurde vermutet, dass die einfache und schnelle Initialisierung einer Kollaborationssoftware eine entscheidende Rolle für deren Annahme in der Praxis spielen könnte. Gerade in Situationen, bei denen das Gespräch

schon im Gange ist und spontan eine Software mit eingebracht werden soll, die einige Teilnehmer bisher nicht verwenden, würde sich eine aufwändige Einrichtung negativ auf den Prozess auswirken und möglicherweise die Verwendung der Software von vornherein verhindern.

Bei Peer-to-Peer-Systemen, die nicht auf eine zentrale Komponente zurückgreifen können, werden für einen initialen Verbindungsaufbau bestimmte Informationen benötigt. Die Herausforderung besteht darin, es einerseits dem System zu ermöglichen möglichst viele Informationen automatisiert zu erhalten und andererseits, falls eine manuelle Eingabe von Informationen unabdingbar ist, diese für den Nutzer möglichst effizient, einfach und nachvollziehbar zu gestalten.

Sicherheit und Vertraulichkeit

Da in persönlichen Gesprächen wie Meetings oftmals kritische Informationen ausgetauscht oder wirtschaftlich relevante Ideen entwickelt werden, wurde vermutet, dass Aspekte wie Sicherheit und Vertraulichkeit, vor allem im professionellen Kontext, eine wesentliche Rolle spielen.

Geschwindigkeit und Latenz

Findet Kollaboration in einem Face-to-Face-Kontext statt, ist ein Großteil der Interaktion und Kommunikation sehr direkt. Wichtige Informationen können verbal schnell ausgetauscht werden. Die Reaktionen des Gegenübers erreichen den Gesprächsteilnehmer ohne eine wahrnehmbare oder als störend empfundene Latenz. Die Vermutung liegt nahe, dass eine Softwarelösung, die die Face-to-Face-Kommunikation ergänzen soll, den Aspekten Geschwindigkeit und Latenz ebenfalls Beachtung schenken sollte, um sich dieser quasi verzögerungsfreien Kommunikation anzupassen. Verzögerungen während der Nutzung und bei der Übertragung und Darstellung von Informationen könnten den natürlichen Redefluss unterbrechen und ablenkend wirken.

3.2 Wahl der Methodik

Bei Face-to-Face-Kollaborationen handelt es sich um komplexe Zusammenhänge, deren einzelne Faktoren sich nur schwer isolieren lassen. Außerdem liegt die Vermutung nahe, dass das subjektive Erleben des Einzelnen einen großen Einfluss auf Entscheidungen

hat, welche in Kollaborationssituationen getroffen werden. Daher kann eine quantitative Untersuchung unter Laborbedingungen, die auch den Rahmen dieser Arbeit sprengen würde, dem Nutzungskontext nur eingeschränkt gerecht werden. Aus diesem Grund fiel die Wahl auf qualitative Verfahren, mit deren Hilfe untersucht werden soll, wie Face-to-Face-Kollaboration ablaufen kann, welche Faktoren dabei eine Rolle spielen und wie die Teilnehmer die entsprechenden Situationen subjektiv erleben.

Um einen umfassenderen Einblick in verschiedene Arten der Kollaboration zu erhalten und gleichzeitig das subjektive Erleben des Einzelnen berücksichtigen zu können, wurden Interviews mit potentiellen Nutzern durchgeführt. In Ergänzung dazu wurde eine reale Kollaborationssituation in einer Fallstudie beobachtet und analysiert.

3.3 Interviews

Bei der Durchführung von Interviews spielt die Wahl einer angemessenen Interviewtechnik eine wichtige Rolle. Da High-Level-Anforderungen an eine Softwarearchitektur ermittelt werden sollen, die in unterschiedlichen Kontexten verwendet werden kann, ist es wichtig, ein gewisses Spektrum an potentiellen Nutzern zu befragen. Auch spielt die Vergleichbarkeit der Ergebnisse eine Rolle. Werden wiederkehrende Muster in den Gesprächen entdeckt, so können daraus möglicherweise allgemeinere Prinzipien abgeleitet werden, aus denen sich Anforderungen an die zu entwickelnde Architektur ergeben. Um in einem begrenzten zeitlichen Rahmen mehrere Arten potentieller Nutzer befragen zu können und die Ergebnisse vergleichbar zu halten, eignen sich semistrukturierte Experteninterviews, da hier ein im Vorhinein erarbeiteter Interviewleitfaden den Fokus des Gesprächs auf ein bestimmtes Thema lenkt. Gleichzeitig wird den Teilnehmern die Freiheit gelassen, ihre eigenen Gedanken zu artikulieren und Themen anzusprechen, deren Relevanz dem Interviewer zuvor nicht bewusst war.

3.3.1 Sampling

Neben pragmatischen Überlegungen, wie der Zugänglichkeit bestimmter Personenkreise, wurden inhaltliche Kriterien bei der Auswahl der geeigneten Interviewpartner angewendet. Zwei wichtige Kontexte, in denen Face-to-Face-Kollaboration eine wichtige Rolle spielt, sind der professionelle Kontext und der Kontext der Lehre. Um beide abzudecken, wurden einerseits mehrere Mitarbeiter eines mittelständischen Unternehmens befragt, in dem kreative Produktentwicklung ein hohes Maß an Kommunikation erforderlich macht und andererseits mehrere Studenten, in deren Studiengang Wert auf projektorientierte

Teamarbeit gelegt wird. Alle Teilnehmer verfügen über mehrere Jahre Erfahrung im jeweiligen Gebiet.

3.3.2 Interviewleitfaden

Vor der Durchführung der eigentlichen Interviews wurde ein Interviewleitfaden entwickelt, der den groben Ablauf der Interviews vorgibt und den Fokus auf bestimmte Themen lenken soll. Die finale Version des Leitfadens ist in Abbildung 3.1 dargestellt. Eine erste Version sah einen deutlich größeren Fragenkatalog vor. Wie sich zeigte, hätte dieser jedoch zu einer schlecht händelbaren Menge an Transkritionsmaterial geführt. Um dennoch in angemessener Zeit vergleichbare Ergebnisse zu erhalten, wurde der Fragenkatalog daher auf wesentliche Aspekte reduziert.

Zu Beginn jedes Interviews wird dem Interviewpartner die thematische Ausrichtung erläutert. Dies soll ein zu starkes Abschweifen in späteren Interviewphasen und eine daraus resultierende Notwendigkeit der Intervention durch den Interviewer präventiv verhindern. Die im Hauptteil gestellten Fragen sind auf zwei Gliederungsebenen verteilt. Während die Fragen auf der ersten Ebene obligatorisch sind und den Rahmen des Interviews festlegen, handelt es sich bei den Fragen auf zweiter Ebene um optionale Vertiefungsfragen. Generell ist dem Interviewer vorbehalten, Fragen an den jeweiligen Interviewkontext anzupassen, um flexibel auf den jeweiligen Interviewpartner reagieren zu können.

Thematisch gliedern sich die Fragen des Hauptteils in zwei Bereiche. Die ersten drei Fragen zielen darauf ab, Informationen über den Kontext der Kollaboration, Abläufe und Tätigkeiten, sowie damit zusammenhängenden Erwartungen und Problemen zu gewinnen. Die folgenden Fragen adressieren direkt Softwaresysteme, die die Face-to-Face-Kollaboration unterstützen. Dabei wird einerseits auf bereits genutzte Softwarelösungen eingegangen und andererseits konkret nach Anforderungen an potentiell zu verwendende Software in diesem Bereich gefragt.

Am Ende jedes Interviews wird zusätzlich Zeit für weitere Äußerungen der Interviewpartner eingeräumt, um etwaige Gedanken einzufangen, die im Verlauf des Interviews nicht zur Sprache gekommen sind.

3.3.3 Durchführung

Im Vorhinein wurden die entsprechenden Personen kontaktiert und nach ihrer Bereitschaft gefragt, als Interviewpartner zur Verfügung zu stehen. Insgesamt wurden acht In-

Interviewleitfaden
==========

Einleitung
Es geht um Softwaresysteme, die Menschen bei der Face-to-Face-Kollaboration unterstützten, d.h. wenn sie mit anderen Menschen im gleichen Raum zusammenarbeiten. Ich versuche mit Hilfe dieser Interviews zu ermitteln, welche grundlegenden Eigenschaften eine solche Software aufweisen müsste.

Interview

- Was sind deine Aufgaben im Arbeits-/Studienalltag und in welchen Situationen arbeitest du dabei mit anderen Personen im gleichen Raum zusammen?
  - Meetings, Ideen entwickeln, gemeinsam an Dokumenten arbeiten?
- Wie läuft diese Zusammenarbeit ab?
  - Wie bereitet ihr euch vor?
  - Wie läuft die Kommunikation ab?
  - Habt ihr bestimmte Rollen? Reden alle gleich viel?
  - Welche Ergebnisse stehen am Ende? Gibt es Artefakte?
- Was ist dir bei dieser Art der Zusammenarbeit besonders wichtig? Womit bist du manchmal unzufrieden?
- Nutzt du/ihr irgendeine Form von Softwareunterstützung bei dieser Zusamenarbeit?
  - Warum? bzw. warum nicht?
  - Was war der Grund, sich für diese Software zu entscheiden?
  - Welche Erfahrungen habt ihr damit gemacht?
- Welche Bedingungen muss eine Software erfüllen, damit du sie in solchen Situation verwenden würdest?
  - Wie schätzt du die Bereitschaft der Personen, mit denen du zusammenarbeitest ein?
  - Gibt es bei euch bestimmte Unternehmensrichtlinien?

Ende
- Gibt es im Bezug auf das Thema noch irgendeinen Punkt, der dir wichtig ist?
- Danksagung

Abbildung 3.1: Leitfaden für die semistrukturierten Experteninterviews

terviews durchgeführt. Wenn möglich, fanden die Interviews im selben räumlichen Kontext statt wie auch die besprochene Kollaboration, das heißt im Unternehmensgebäude auf dem Campus der Hochschule. In einem kurzen Vorgespräch wurde das Thema der Masterarbeit sowie der Zweck des Interviews erläutert sowie auf die Notwendigkeit ei-

ner Audioaufnahme hingewiesen. Da dedizierte Tonaufnahmegeräte möglicherweise als ungewohnt empfunden werden und somit die Nervosität des Interviewpartners steigern könnten, wurden die Interviews mit Hilfe eines auf dem Tisch liegenden Smartphones aufgezeichnet. Falls ein Interviewpartner nach Beendigung der Aufnahme noch relevante Aussagen machte, so wurden diese in textuellen Feldnotizen festgehalten.

Nach Abschluss der Interviews wurden diese transkribiert (siehe Anhang) und einer induktiven thematischen Analyse [vgl. z.B. BC06] unterzogen. In den Transkripten vorkommende Konzepte wurden im Text kodiert. Diese Codings wurden wiederum analysiert, um daraus Themen zu erarbeiten. Dabei wurde vor allem Wert auf folgende Punkte gelegt:

- Typische Kollaborationsabläufe
- Probleme in der gängigen Praxis
- Relevante Eigenschaften von Softwarelösungen

Die identifizierten Themen und ihre Struktur sind in Abbildung 3.2 dargestellt. Diese Themen wurden in einem weiteren Analyseschritt wiederum anhand der Transkripte auf ihre tatsächliche Relevanz hin gegengeprüft.

3.3.4 Ergebnisse

Im Folgenden sollen die Erkenntnisse aus der Durchführung und Analyse der Interviews zusammengefasst werden. Dies geschieht anhand von für die zugrundeliegende Fragestellung relevanten Aspekten.

Beteiligungsförderung

Ein wesentlicher Punkt, der in einigen Interviews zur Sprache kam, war die Förderung der Beteiligung in Face-to-Face-Kollaborationssituationen. Oftmals gibt es Teilnehmer, die sich aus verschiedenen Gründen weniger beteiligen, obwohl der Wunsch vorhanden ist, die Meinung jedes einzelnen zu berücksichtigen und sie vielleicht substanziell zur Zusammenarbeit beitragen könnten(vgl. z.B. Interview 1 und 2).

Ein Kollaborationssystem für Face-to-Face-Situationen sollte die Beteiligung einzelner fördern und beteiligungshemmenden Faktoren entgegenwirken.

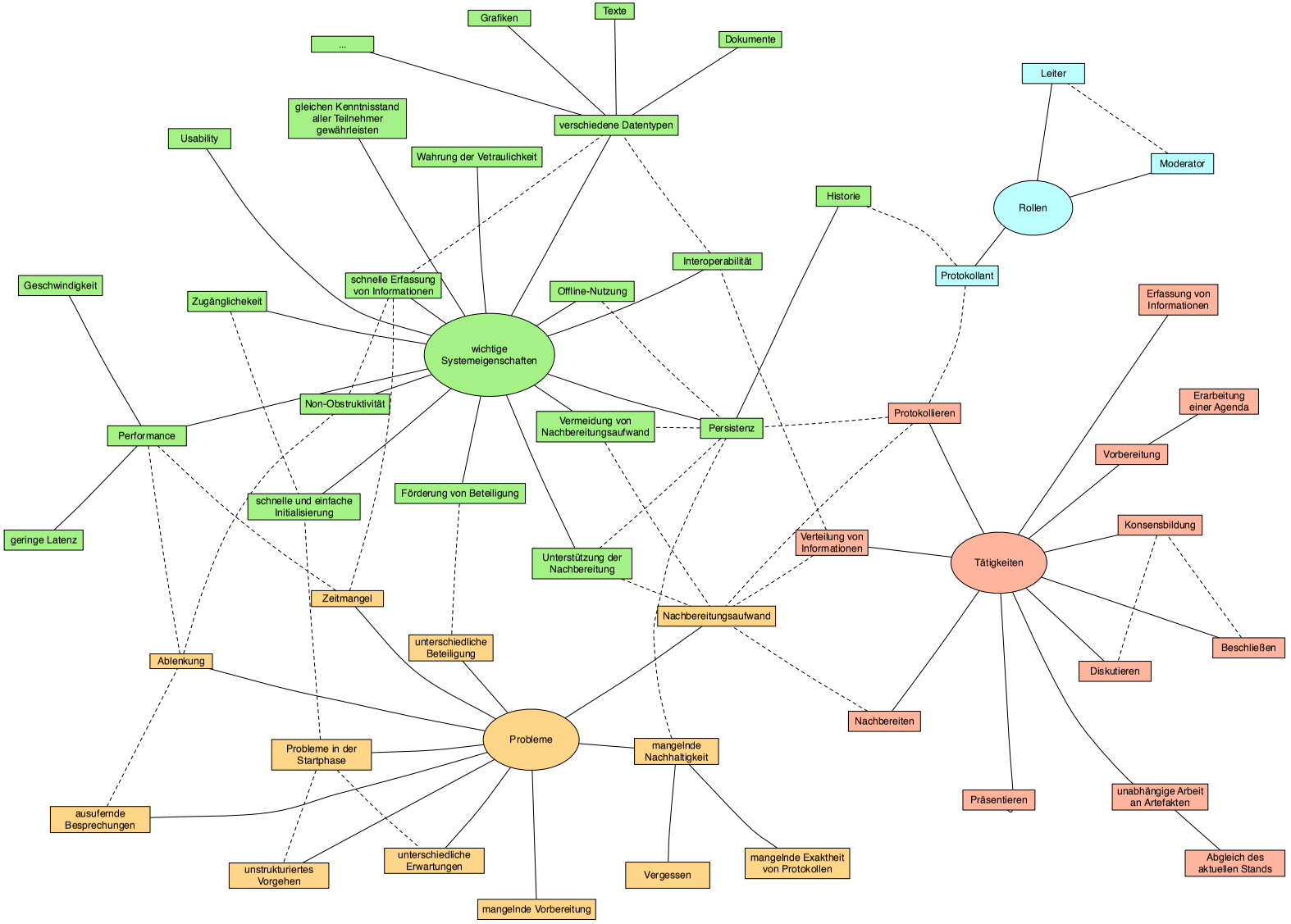

Abbildung 3.2: Aus den Interviews abgeleitete Themenstruktur

Initialisierung

In Abschnitt 3.1 wurde bereits vermutet, dass gerade die einfache Initialisierung einer Software ein wichtiger Faktor für die Bereitschaft sein könnte, diese Software auch tatsächlich einzusetzen. Diese Vermutung bestätigte sich bei der Analyse der Interviews zumindest teilweise.

Die Initialisierungsphase wurde nur von wenigen Teilnehmern direkt angesprochen. Ein Interviewpartner beispielsweise äußerte den Wunsch, eine Software ohne großen Konfigurationsaufwand direkt verwenden zu können (vgl. Interview 4). Dennoch kamen andere Faktoren zur Sprache, die implizit mit der Initialisierung zusammenhängen. So wurde mehrfach geäußert, dass Face-to-Face-Kollaboration spontan stattfinden kann. Auch wurde von Problemen in der Startphase der Kollaboration berichtet. Eine komplexe Initialisierung würde in dieser kritischen Phase der Zusammenarbeit weitere Hürden in den Weg stellen.

Als Grund für die Nutzung bestimmter Hilfsmittel, seien sie technisch oder auch nicht, wurden mehrfach deren schnelle Verfügbarkeit und Zugänglichkeit für alle Teilnehmer angeführt. Beispielsweise sind Papier und Stift sofort einsatzbereit (vgl. Interview 3).

Ein Kollaborationssystem für Face-to-Face-Situationen sollte der Zusammenarbeit, gerade in der Startphase, möglichst keine Hürden in den Weg legen und durch eine einfache Initialisierung möglichst schnell für alle Teilnehmer verfügbar sein.

Nachhaltigkeit

Ein Aspekt, dessen Relevanz im Vorhinein nicht antizipiert wurde, der sich aber schon während der Durchführung und auch bei der Analyse der Interviews als wichtig herausstellte, ist die Nachhaltigkeit von Kollaboration und deren Ergebnissen. Es handelt sich um ein Thema, das viele Interviewpartner ansprachen (vgl. Codings zum Thema Nachhaltigkeit im Anhang) und das einige Probleme hervorruft. Oftmals vergessen Teilnehmer die letztendlichen Beschlüsse und verlieren möglicherweise festgelegte Verantwortlichkeiten und Termine aus den Augen. Artefakte der Zusammenarbeit gehen verloren, oder werden anderen Teilnehmern nicht zur Verfügung gestellt.

Eine gängige Vorgehensweise zur Gewährleistung von Nachhaltigkeit ist die Anfertigung eines Protokolls. Oft wird hierfür explizit ein Protokollant bestimmt. In zwei Interviews wurde angesprochen, dass dabei die Exaktheit des Protokolls eine wesentliche Rolle spielt (vgl. Interview 1 und 2). Die Erstellung eines Protokolls alleine reicht allerdings nicht aus.

Die bei einem persönlichen Treffen anfallenden Informationen und Artefakte müssen an die Teilnehmer verteilt werden und dauerhaft zugänglich bleiben.

Ein Kollaborationssystem für Face-to-Face-Situationen sollte diesen Aspekt der Nachhaltigkeit berücksichtigen. Dabei können unter anderem Faktoren wie langfristige Persistenz und die Verteilung von Informationen eine Rolle spielen.

Vor- und Nachbereitung

Face-to-Face-Kollaboration ist oft kurzfristig, bedarf aber trotzdem einer gewissen Vor- und Nachbereitung. Wie sich in den Interviews zeigte, bereiten sich viele Teilnehmer vor, indem sie sich mit den für die Face-to-Face-Kollaboration vorgesehenen Themen auseinandersetzen. In einigen Fällen wird vorher von einer oder mehreren Personen eine Agenda erarbeitet. Auch werden zuvor vorbereitete Artefakte in persönlichen Treffen anderen Personen präsentiert.

Der Aspekt der Nachbereitung hängt eng mit der oben angesprochenen Nachhaltigkeit zusammen. Dazu gehören die Abarbeitung von beim persönlichen Treffen festgelegten Aufgaben, aber auch die Aufbereitung und Verteilung von Informationen an alle Teilnehmer. Letzteres stellt einen Aufwand dar, der reduziert werden kann.

Ein Kollaborationssystem für Face-to-Face-Situationen sollte dementsprechend die Vor- und Nachbereitung unterstützen aber, wo sinnvoll und möglich, auch automatisieren um Menschen zu entlasten und ihnen die Möglichkeit zu geben, sich auf andere Aufgaben zu konzentrieren.

Kodalitäten und Datenformate

Kollaboration hat einen starken Bezug zu Artefakten, die in die Zusammenarbeit einfließen oder während der Zusammenarbeit entstehen. Wie sich zeigte, spielen dabei unterschiedliche Kodalitäten von Informationen und dementsprechend auch eine Vielzahl von Datenformaten eine Rolle.

Ein Großteil der in persönlichen Treffen verwendeten oder anfallenden Daten sind textuell. Sie unterscheiden sich in ihrer Strukturiertheit (lose Notizen, Dokumente, Aufgaben und Verantwortlichkeiten , Agendas, Literaturlisten). Aber auch andere Kodalitäten spielen eine Rolle. Ein Interviewpartner sprach explizit an, dass er bei persönlichen Treffen visuelle Formen der Kommunikation, also beispielsweise kurzfristig angefertigte Skizzen, Grafiken und Präsentationsfolien, bevorzugt (vgl. Interview 8). Ein anderer Interview-

partner sprach nach Beendigung des Interviews an, dass Informationen möglicherweise in auditiver Form schneller festgehalten werden könnten als in Textform.

Für verschiedene Kodalitäten von Informationen sind auch verschiedene Datenformate notwendig. Wie zu erwarten kommen in der momentanen Praxis viele unterschiedliche Formate zum Einsatz. Explizit angesprochen wurden von den Interviewpartnern Dokumentenformate wie PDF, Word-Dokumente, Latex, Präsentationsformate wie Powerpoint-Folien, aber auch andere Formate, wie beispielsweise 3D-Modelle, die mit verschiedenen CAD-Lösungen erstellt wurden.

Ein Kollaborationssystem für Face-to-Face-Situationen sollte diese Vielfalt an verwendeten Kodalitäten und Datenformaten berücksichtigen.

Interoperabilität

Das Thema Interoperabilität wurde von den interviewten Personen nicht direkt angesprochen. Dennoch zeigte sich in der Analyse anderer Aspekte eine gewisse Notwendigkeit der Interoperabilität mit anderen Systemen.

Da sich Teilnehmer in vielen Fällen auf die Face-to-Face-Kollaboration vorbereiten, liegen oftmals schon zu Beginn der eigentlichen Zusammenarbeit Informationen in einem bestimmten Format vor. Diese müssen in irgendeiner Form in die Zusammenarbeit integriert werden. Die Verwendung verschiedener Datenformate wurde im vorherigen Abschnitt bereits angesprochen.

Ein wesentlicher Punkt, der in einigen Interviews zur Sprache kam, ist die Verteilung von Informationen an Teilnehmer und dritte Personen nach Abschluss der Face-to-Face-Kollaboration und die Nachverfolgung von Beschlüssen (siehe Abschnitt Nachhaltigkeit). Auch hierfür wäre eine Integration mit bereits verwendeten Softwarelösungen sinnvoll. Somit könnten bestehende Workflows und Kommunikationskanäle weitergenutzt werden.

Ein Kollaborationssystem für Face-to-Face-Situationen sollte bereits verwendete Softwarelösungen berücksichtigen und eine Integration mit diesen ermöglichen.

Non-Obstruktivität

Wie die Analyse der Interviewtranskripte zeigte, spielt der Faktor der Non-Obstruktivität eine wichtige Rolle bei der Verwendung von Software in Face-to-Face-Kollaborations-

situationen, d.h. in wie fern eine Softwarelösung den Arbeits- und Gedankenfluss der beteiligten Personen behindert.

Einige Interviewpartner hatten die Erfahrung gemacht, dass in Face-to-Face-Situationen ein besonders hohes Potential für Abschweifen oder Ablenkungen vom eigentlichen Thema besteht (vgl. Interview 3 und 5). Auch äußerten einige explizit den Wunsch, dass eine Kollaborationslösung nicht von den eigentlichen Inhalten der Zusammenarbeit ablenken oder den Gesprächsfluss behindern sollte (vgl. Interview 1 und 3).

Auch im Sinne der oben angesprochenen Beteiligungsförderung spielt dieser Aspekt eine Rolle. Würden andere Teilnehmer durch die Verwendung der Software während des Gesprächs abgelenkt, so würde dies die Berücksichtigung der jeweiligen Gesprächsbeiträge verschlechtern.

In Abschnitt 3.1 wurde bereits vermutet, dass auch die Geschwindigkeit und Latenz eines kollaborativen Software eine wesentliche Rolle spielen könnten. Diese beiden Faktoren lassen sich ebenfalls unter dem Oberbegriff Non-Obstruktivität einordnen. Ein Interviewpartner äußerte beispielsweise, dass die Sichtbarkeit von Bearbeitungen in Echtzeit wichtig wäre, um ein direktes Anknüpfen an die Beiträge anderer Teilnehmer zu ermöglichen (vgl. Interview 4).

Ein Kollaborationssystem für Face-to-Face-Situationen sollte dementsprechend non-obstruktiv sein, Teilnehmer nicht ablenken und den Arbeits- und Gesprächsfluss nicht negativ beeinflussen.

Vertraulichkeit

Entgegen den Erwartungen spielte für viele Interviewpartner der Aspekt der Vertraulichkeit von Informationen eine weniger wichtige Rolle. In Unternehmen gibt es durchaus Projekte, die einer gewissen Geheimhaltung unterliegen und deren Informationen nur bestimmten Personen zugänglich gemacht werden dürfen (vgl z.B. Interview 1). Auch sind Mitarbeiter teilweise vertraglich verpflichtet, für das Unternehmen kritische Informationen nicht an Dritte weiterzugeben (vgl z.B. Interview 2). Allerdings kamen zumindest in den Interviews keine konkreten, aktiven Maßnahmen zur Sprache, die in der bisherigen Praxis getroffen werden, um diese Vertraulichkeit von Informationen zu wahren.

Es lässt sich vermuten, dass Vertraulichkeit bei Face-to-Face-Situationen in so fern eine untergeordnete Rolle spielt, dass ein gewisses Vertrauen der Teilnehmer untereinander als Basis für die Zusammenarbeit vorausgesetzt wird. Bei verbaler Kommunikation in einem Raum, bekommen potentiell auch alle Anwesenden alles was gesagt wird mit. Vertrau-

lichkeit würde sich dann vor allem auf die Sicherheit der Informationen vor dem Zugriff Dritter beziehen, die nicht an der Zusammenarbeit beteiligt sind.

Ein Kollaborationssystem für Face-to-Face-Situationen sollte dementsprechend während der Zusammenarbeit anfallende Informationen angemessen vor dem Zugriff durch Dritte schützen.

3.3.5 Kritische Reflexion

An dieser Stelle sollen Durchführung und Ergebnisse der Interviews einer kritischen Reflexion unterzogen werden.

Interviewtechniken eignen sich gut zur Erhebung bestimmter Aspekte eines Problems. Andere Aspekte sind hiermit nur schwer zugänglich. Dazu gehören unterbewusste Faktoren, die in der Lebensrealität der Befragten zwar eine Rolle spielen, die sie aber nicht explizit äußern können. Im Blick auf die geplante Architektur können das beispielsweise Faktoren wie die Geschwindigkeit und Latenz sein, mit der das System auf Nutzerinteraktionen reagiert. Es liegt nahe, dass diese Aspekte eine Rolle bei der subjektiven Wahrnehmung einer Anwendung durch den Nutzer spielen, auch wenn er sie ab einer bestimmten Grenze nicht bewusst kritischen Faktor betrachtet. Um solche Faktoren dennoch zu berücksichtigen, bedürfen die Ergebnisse der Befragung einer entsprechenden Interpretation, die andererseits immer das Risiko einer Fehleinschätzung birgt.

Neben den natürlichen Einschränkungen von Interviewtechniken, wurden, aufgrund des begrenzten Zeitrahmens und der begrenzten Erfahrung, auch Kompromisse im Bezug auf Planung und Durchführung der Interviews gemacht. Ein solcher Punkt ist die Samplingstruktur. Im wesentlichen wurden zwei Personengruppen befragt. Wie sich in der Analyse zeigte, ähneln sich die Antworten der Mitglieder einer Gruppe relativ stark. Es steht zu vermuten, dass eine größere Varianz bei der Wahl der Interviewpartner zu reichhaltigeren Ergebnissen geführt hätte. Ein weiterer Punkt ist die Gestaltung des Interviewleitfadens. Einige Annahmen, die bei der Anfertigung der Fragen im Raum standen, spiegeln sich relativ stark in den finalen Codings wieder. Ein Beispiel hierfür ist die Vermutung, dass bei der Kollaboration bestimmte Rollenstrukturen existieren. Eine offenere Formulierung der Fragen hätte möglicherweise einen breiteren, weniger verzerrten Blick auf den Problemraum ermöglicht. Gleichzeitig hätte sie allerdings auch eine größeres Spektrum an Themen hervorgebracht. Da im Rahmen dieser Arbeit ein relativ enger Fokus gesetzt werden muss, wurden die Nachteile einer konkreteren Fragestellung in Kauf genommen.

Trotz einiger kritisch zu betrachtenden Aspekte der Interviews, stellen sie insgesamt betrachtet eine wertvolle Quelle für die Gestaltung der geplanten Architektur dar. Es konnten Erkenntnisse gewonnen werden, die deutlich über die im Vorhinein angestellten Vermutungen hinausgehen und aus denen sich einige abstrakte Anforderungen ableiten lassen.

3.4 Fallstudie

Zusätzlich zu den Interviews, wurde eine Fallstudie in Form einer teilnehmenden Beobachtung durchgeführt, deren Zweck es war, einen Einblick in die Abläufe realer Kollaborationssituationen zu gewinnen. Hierfür wurde eine Vorlesungssituation ausgewählt, in der die Studenten vom Dozenten spontan dazu aufgefordert wurden, ein Kollaborationstool zu nutzen, um Begriffe zu bestimmten Themen zusammenzutragen, welche im weiteren Verlauf der Vorlesung dann aufgegriffen wurden.

Aus Gründen der Zugänglichkeit, wurde auf die Vorlesung eines Informatikstudiengangs zurückgegriffen. Es lässt sich dementsprechend vermuten, dass viele Teilnehmer eine gewisse technische Affinität besitzen und teilweise auch Vorerfahrungen mit Kollaborationstools mitbringen.

3.4.1 Verwendete Software

Zum Einsatz kam die Softwarelösung „Smart Notebook“ [SMA15]. Dabei handelt es sich um eine Software des Unternehmens SMART Technologies, die unter anderem für den Einsatz auf den vom selben Unternehmen vertriebenen SMART Boards, einem interaktiven Whiteboard für Lehrsituationen, konzipiert ist. Dabei ist das Gerät, auf dem die Software läuft, an ein zentrales Anzeigemedium, im hier betrachteten Fall ein Projektor, angeschlossen. Im Speziellen kam die „XC Collaboration“-Komponente der Software zum Einsatz, die ein kollaboratives Zusammentragen von Informationen ermöglicht. Hierfür können die Teilnehmer ihre eigenen Endgeräte, wie Smartphones oder Notebooks, verwenden.

Um eine Kollaborationssitzung zu starten, wird auf dem zentralen Bildschirm ein QR-Code angezeigt, welcher die Endgeräte der Teilnehmer auf eine entsprechende Webapplikation weiterleitet, mittels der sie mit der Smart Notebook Software auf dem Gerät des Lehrenden kommunizieren können. Sollte kein QR-Code Reader verfügbar sein, wird alternativ eine URL inklusive eines einzugebenden Identifikationscodes angezeigt.

Haben sie die Webapplikation auf ihrem Endgerät aufgerufen, können die Teilnehmer Texte und Bilder an die Smart Notebook Anwendung senden. Diese erscheinen dann auf dem zentralen Bildschirm. Der Lehrbeauftragte hat die Möglichkeit, sie auf einer zweidimensionalen Fläche anzuordnen und so gegebenenfalls zu strukturieren. Das Senden der Texte und Bilder kann wahlweise anonym erfolgen. Um die Beteiligung zu fördern, wurde auf diese Funktion zurückgegriffen und die Studenten darauf hingewiesen, dass sich nicht nachvollziehen lässt, welche Informationen von wem gesendet wurden.

3.4.2 Durchführung

Die Durchführung der Beobachtung wurde im Vorhinein mit dem Dozenten abgesprochen. Zu Beginn der Fallstudie wurden die anwesenden Studenten um ihr Einverständnis gebeten, aufgezeichnet zu werden. Ein Teil der Veranstaltung wurde mit Hilfe von zwei Kameras aufgezeichnet. Um möglichst wenig invasiv zu wirken und die Studenten nicht durch die Anwesenheit größerer Kameras zu verunsichern, wurden hierfür ein Smartphone und eine Kompaktkamera verwendet. Die Kameras waren in verschiedenen Winkeln positioniert, um die Situation sowohl von der Vorderseite als auch von der Rückseite des Raumes aus beobachten zu können.

Nach Abschluss des relevanten Vorlesungsteils, wurden die Kameras abgeschaltet. Die dabei enstandenen Aufzeichnungen wurden in einer nonlinearen Videoschnittsoftware anhand ihrer Audiospur synchronisiert und in einem Bild-in-Bild-Format exportiert. Dies erleichterte die Analyse, da die Möglichkeit bestand, die Szene gleichzeitig aus verschiedenen Winkeln zu betrachten. Die entstandene kombinierte Aufzeichnung wurde vollständig transkribiert (siehe Anhang). Dabei wurde nicht nur der Wortlaut der sprechenden Personen festgehalten, sondern, sofern möglich, auch im Verlauf der Vorlesung durchgeführte Handlungen und Ereignisse. Das erstelle Transkript wurde dann in Kombination mit dem Videomaterial analysiert.

3.4.3 Beobachtungen

Wie sich bei der Analyse der Fallstudie zeigte, war die Initialisierung mittels QR-Code oder URL für die meisten Studenten kein Problem. Noch während der Erläuterung des Verbindungsprozesses durch den Dozenten, gelang es den ersten Teilnehmern sich mit der Software zu verbinden. Nur in einem Fall ergaben sich beim Scannen des QR-Codes mit einem Smartphone Schwierigkeiten, so dass auf eine manuelle Eingabe der URL zurückgegriffen wurde.

Auch die Beteiligung der Studenten erwies sich in dieser Situation nicht als problematisch. In der Regel erschienen bereits kurz nach der Aufforderung des Dozenten Antworten auf die entsprechende Frage auf dem Bildschirm und die Zahl der gesendeten Begriffe war relativ hoch. Allerdings zeigte sich, dass der Dozent keine Möglichkeit hat um einzuschätzen, ob noch jemand damit beschäftigt ist, eine Antwort zu schreiben und somit weitere Begriffe folgen werden. Diese fehlende Awareness-Information wirkte sich störend auf den Verlauf der Vorlesung aus, da der Dozent gezwungen war, entweder sicherheitshalber eine gewisse Zeit abzuwarten oder explizite Nachfragen zu stellen, was sich aber wiederum als problematisch herausstellte, da hierdurch die angekündigte Anonymität gefährdet worden wäre.

3.4.4 Erkenntnisse aus der Fallstudie

Initialisierung

Die Initialisierung einer Sitzung mittels QR-Code oder URL und Authentifizierungsnummer, wie von Smart Notebook umgesetzt, bereitet zumindest einer technisch versierten Zielgruppe nur wenige Probleme. Allerdings bleibt fraglich, in wie fern sich diese Erkenntnis auf andere Zielgruppen ausweiten lässt. In jedem Fall jedoch bietet die Darstellung eines Authentifikationsmerkmals auf einem zentralen Anzeigemedium im Raum den Vorteil, dass es sich um einen relativ sicheren Kommunikationskanal handelt, der im Idealfall gewährleistet, dass nur persönlich anwesende Personen Zugriff auf die entsprechende Kollaborationssitzung erhalten.

Awareness

Der Bereich der Awareness bietet weitere Verbesserungschancen für Face-to-Face Kollaborationssysteme. Zunächst lag die Vermutung nahe, dass durch die persönliche Anwesenheit der beteiligten Personen bereits alle notwendigen Kontextinformationen für eine effektive Zusammenarbeit verfügbar seien. Dabei wurde allerdings die Verwendung der Software selbst außer Acht gelassen. Bereits während der Beobachtung fiel auf, dass sich, trotz persönlicher Anwesenheit des Analysten im Raum, die Aktivitäten einzelner Teilnehmer innerhalb der Software nur schlecht nachverfolgen lassen. Und wie in Abschnitt 3.4.3 angesprochen, wirkte sich diese fehlende Information auf Seiten des Dozenten negativ auf den Verlauf der Vorlesung aus.

Da die Verwendung der Software ein Bestandteil der Interaktion der Teilnehmer ist, sollte eine Kollaborationslösung diesem Aspekt Beachtung schenken und, wo sinnvoll, den

Teilnehmern Kontextinformationen zur Verfügung stellen, die sich nicht bereits aus der physischen Anwesenheit ergeben.

3.4.5 Kritische Reflexion

Ähnlich wie bei den Interviews, sind auch die Ergebnisse der Fallstudie nicht ohne eine kritische Reflexion zu betrachten. Beobachtet wurde eine einzige, relativ spezielle Situation. Daher kann hieraus kein umfassendes Bild spontaner Kollaborationssituationen gewonnen werden. Die gemachten Beobachtungen dienen vielmehr als Impuls für eigene Überlegungen. In diesem Sinne betrachtet, war die Fallstudie erfolgreich, da sie einen Fokus auf bestimmte Aspekte des Problems gelegt hat, die ansonsten möglicherweise untergegangen wären, allen voran das Thema Awareness.

4 Entwurf einer Architektur

Nach Abschluss der Recherche und Kontextanalyse soll nun in diesem Kapitel die Gestaltung einer Softwarearchitektur beschrieben werden, welche die gesteckten Ziele umsetzt. Hierfür werden, basierend auf den Erkenntnissen der vorherigen Kapitel, zunächst noch einmal grundlegende Eigenschaften, in Form von High-Level-Anforderungen, festgelegt, die die Architektur aufweisen sollte. Es folgt eine kurze Analyse der zur Verfügung stehende Browsertechnologien. Anschließend sollen wesentliche Aspekte der Architektur sowie damit zusammenhängende Überlegungen und Abwägungen erläutert werden, bevor diese in einer Gesamtarchitektur zusammengefasst werden.

4.1 High-Level-Anforderungen

In den vorherigen Kapiteln ergaben sich bereits einige Erkenntnisse bezüglich der Anforderungen an Softwaresysteme für Face-to-Face-Kollaboration. Im Folgenden sollen nun die wesentlichen Anforderungen festgelegt werden, an denen sich die zu erarbeitende Architektur orientieren wird.

Da die Architektur für eine Plattform ausgelegt ist, werden keine konkreten funktionalen Anforderungen für eine spezielle Kollaborationslösung defininiert. Stattdessen handelt es sich um High-Level-Anforderungen, das heißt generelle Eigenschaften, die die Plattform bzw. ihre Architektur aufweisen sollte.

Unabhängigkeit von zentraler Infrastruktur

Die Unabhängigkeit von zentraler Infrastruktur ist ein Grundgedanke dieser Arbeit (vgl. Abschnitt 1.2). Es handelt sich um eine Lösung für Face-to-Face-Kollaboration, die auch spontan stattfinden kann. In solchen Situationen ist oftmals keine Infrastruktur vorhanden und der Einrichtungsaufwand sollte so gering wie möglich gehalten werden. Zudem kann nicht in allen Fällen von einer bestehenden Internetverbindung ausgegangen werden. Daher ist eine Kernanforderung an die zu entwickelnde Architektur die Unabhängigkeit von vorhandener Infrastruktur. Das umfasst vor allem folgende Aspekte:

- Unabhängigkeit von einer Internetverbindung
- Unabhängigkeit von zentralen Organisationseinheiten
- Unabhängigkeit von, auf den Endgeräten installierter, Software

Performance und Geschwindigkeit

Die Performance der Anwendung ist in vielerlei Hinsicht ein kritischer Faktor. In Abschnitt 3.3.4 wurde angesprochen, dass eine Kollaborationsanwendung non-obstruktiv sein sollte. Face-to-Face-Kollaboration basiert zu großen Teilen auf verbaler Kommunikation und kann dementsprechend sehr schnell ablaufen. Durch eine schlechte Performance würde eine Kollaborationsanwendung die Zusammenarbeit ausbremsen und störend wirken. Während der Kontextanalyse (vgl. Kapitel 3) äußerte beispielsweise ein Interviewpartner explizit, dass Änderungen, die von anderen Teilnehmern vorgenommen wurden, instantan sichtbar sein sollten, damit man daran anknüpfen kann (vgl. Interview 4). Auch in Abschnitt 2.2.2 wurden die Vorteile einer Interaktion in Echtzeit angesprochen.

Die Architektur sollte also die Entwicklung performanter Anwendungen ermöglichen. Dabei spielen folgende Aspekte eine Rolle:

- Ermöglichung einer reichhaltigen Interaktion mit möglichst geringer Latenz
- Unterstützung größerer Teilnehmerzahlen
- Schneller Umgang mit größeren Datenmengen
- Vermeidung unnötiger Systemlast

Flexibilität und Wiederverwendbarkeit

Da die zu entwickelnde Architektur als Plattform dienen soll, auf der verschiedene Kollaborationslösungen umgesetzt werden können, spielen Flexibilität und Wiederverwendbarkeit von Komponenten eine wesentliche Rolle bei der Entwicklung. Auch bei den in Abschnitt 2.2 betrachteten Kollaborationslösungen, wurde auf diese Aspekte Wert gelegt.

Ein Faktor, der besonders wichtig erscheint, ist die Trennung der Datenrepräsentation von ihrer Darstellung. Diese Trennung bietet eine Reihe von Vorteilen. Sie erleichtert die Gewährleistung einer konsistenten Datenbasis bei gleichzeitiger Bearbeitung durch mehrere Teilnehmer. Die Datenrepräsentation kann autonom Entscheidungen darüber treffen, welche Informationen zwischen den Teilnehmern ausgetauscht werden um die Daten konsistent zu halten, ohne Implikationen für das User Interface berücksichtigen zu müssen [vgl. Pat91]. Gleichzeitig kann das User Interface implementiert werden, ohne die internen Mechanismen der Datenrepräsentation zu berücksichtigen. Im Idealfall spielt es für das User Interface keine Rolle, ob eine Änderung der Daten lokal oder durch einen anderen Teilnehmer initiiert wurde. Außerdem begünstigt die Trennung von Datenre-

präsentation und -darstellung die Umsetzung verschiedener Perspektiven auf die gleichen Daten [vgl. Pat91].

Die zu entwickelnde Architektur sollte daher folgenden Aspekten Rechnung tragen:

- Hohe Wiederverwendbarkeit der einzelnen Komponenten
- Flexible Paradigmen, die sich für verschiedene Anwendungsfälle nutzen lassen
- Trennung von Datenrepräsentation und -darstellung

Persistenz

Beim Antritt dieser Arbeit, wurde dem Aspekt der Persistenz wenig Bedeutung zugemessen. Im Laufe der Recherche (vgl. Kapitel 2) und der Kontextanalyse (vgl. Kapitel 3) wurde allerdings klar, dass die längerfristige Speicherung von Inhalten eine wichtige Rolle spielt. Kollaboration erstreckt sich oftmals über mehrere Sitzungen (vgl. GroupKit). Teilnehmer wollen auch im Nachgang auf Ergebnisse der Kollaboration zurückgreifen können. Hierfür ist eine persistente Speicherung von Inhalten unumgänglich.

Konsistenz

Wenn mehrere Teilnehmer parallel an denselben Artefakten arbeiten und spontan neue Teilnehmer hinzukommen können, muss die Konsistenz der Daten auf verschiedenen Systemen gewährleistet werden. Die Architektur sollte hierfür Mechanismen vorsehen. Auch die Geschwindigkeit der Synchronisation von Daten spielt eine Rolle (vgl. Abschnitt 4.1).

Unterstützung verschiedener Datentypen

In vielen Fällen umfasst Kollaboration Informationen in einer Vielzahl von Kodalitäten, seien es Texte, Bilder, Audio, Video, 3D-Modelle oder sonstige Artefakte. In vielen Fällen ist eine Repräsentation dieser Daten in Binärform üblich. Es können zudem größere Datenmengen anfallen. Auch wenn nicht jede Anwendung all diese Formate unterstützen muss, sollte die Architektur für eine Kollaborationsplattform grundsätzlich die Verwendung beliebiger Datenformate ermöglichen. Die zu entwickelnde Architektur sollte folgende Aspekte berücksichtigen:

- Grundsätzliche Unterstützung von textuellen Daten

- Grundsätzliche Unterstützung von binären Daten
- Unterstützung größerer Datenmengen

Interoperabilität und Integration

Eine Groupwareanwendung steht selten für sich allein. Wie sich im Laufe der Kontextanalyse herausstellte, müssen oft Inhalte geteilt werden, die in anderen Anwendung erstellt wurden. Die Ergebnisse der Kollaboration müssen weiterverarbeitet oder über andere Kommunikationskanäle weiterverteilt werden. Dementsprechend sollte die Architektur es Anwendungen ermöglichen, sich in bestehende Workflows zu integrieren und die Möglichkeit der Interoperabilität mit anderen Softwarelösungen gewährleisten. Hierzu gehören die Möglichkeit des Imports bestehender Daten aus anderen Anwendungen und die Möglichkeit des Exports von Daten zum Zweck der Verwendung in anderen Anwendungen.

Datenschutz und Sicherheit

Ein Grund für die Verwendung einer dezentralen Kollaborationslösung liegt in Datenschutzbedenken gegenüber klassischen „Cloud-Anwendungen" begründet (vgl. Abschnitt 1.1). Auch wenn eine dezentrale Struktur einem Zugriff durch eine zentrale Instanz vorbeugt, existieren weiterhin Möglichkeiten, um Zugriff auf personenbezogene oder Unternehmensdaten zu erhalten, beispielsweise durch ein abhören des Kommunikationskanals. Die zu entwickelnde Architektur sollte diesem Problem Rechnung tragen und die Vertraulichkeit und Authentizität der Daten gewährleisten.

Einfache Initialisierung

Eine weitere Motivation für den Verzicht auf zentrale Infrastruktur ist die Vermeidung von Konfigurationsaufwand. Bei der spontanen Entscheidung, ob eine Kollaborationslösung eingesetzt wird oder nicht, kann der initiale Aufwand für den Einsatz der Software eine entscheidende Rolle spielen (vgl. auch Abschnitt 3.3.4). Dementsprechend sollte der Architekturentwurf einer Kollaborationsanwendung die Initialisierung einer Kollaborationssitzung mit möglichst geringem Aufwand ermöglichen. Folgende Aspekte spielen dabei eine Rolle:

- Kein Voraussetzen von Wissen über die interne Struktur des dezentralen Systems

- Einfaches und schnelles Aufsetzen einer neuen Kollaborationssitzung
- Einfache und schnelle Möglichkeiten in eine Kollaborationssitzung einzusteigen

Awareness

Auf den ersten Blick scheint Awareness in einer Face-to-Face-Situation keine große Rolle zu spielen, da viele Awarenessinformationen bereits über die räumliche Nähe zu den anderen Teilnehmern verfügbar sind. Wie sich allerdings im Laufe der Recherche und Kontextanalyse zeigte, ist diese Einschätzung unzutreffend. Tatsächlich existiert eine Reihe von Kontextinformationen, die für die gemeinsame Arbeit an Artefakten von Bedeutung sind, und die nicht durch räumliche Anwesenheit abgedeckt werden (vgl. Abschnitt 3.4.4).

Diese Informationen betreffen vor allem die Nutzung des Systems an sich. Wenn jeder Teilnehmer an seinem eigenen Endgerät arbeitet, sind die Handlungen anderer nur begrenzt nachvollziehbar. In vielen Fällen ist es beispielsweise von Nutzen, zu wissen ob und wie eine andere Person gerade mit dem System interagiert, um deren Konzentrationsfähigkeit für verbale Kommunikation einzuschätzen oder nachvollziehen zu können, welche Person man bezüglich einer gerade gemachten Änderung ansprechen muss.

Aus oben genannten Gründen, sollte die Architektur den darauf basierenden Kollaborationsanwendungen Mechanismen zur Verfügung stellen, um Awarenessinformationen zeitnah austauschen zu können.

Unterstützung individueller Arbeit

Wie sich im Laufe der Recherche gezeigt hat, besteht Kollaboration nicht nur aus gemeinsamer Arbeit an den gleichen Artefakten sondern hat oftmals auch einen individuellen Anteil (vgl. z.B. Abschnitt 2.1.1). Zudem kann es von Vorteil sein, Teilnehmern die Möglichkeit einzuräumen, Artefakte zuerst unabhängig von anderen zu erstellen. Wie sich zeigt, können durch solche Maßnahme negative soziale Effekte der Gruppenarbeit abgeschwächt werden (vgl. Abschnitt 2.1.3).

Der Architekturentwurf sollte dementsprechend nicht nur die gemeinsame Arbeit an Artefakten berücksichtigen sondern auch die individuelle Erstellung und Bearbeitung von Inhalten vorsehen. Zudem sollten individuell erstellte Inhalte nachträglich mit in die gemeinsame Arbeit eingebracht werden können.

4.2 Verwendete Webtechnologien

Vor allem aus Sicherheitsbedenken unterliegen Webapplikationen bestimmten Beschränkungen. Beispielsweise können sie nicht auf beliebige Systemressourcen zugreifen und nur unter bestimmten Vorraussetzungen mit anderen Systemen kommunizieren [Moz15b]. In den letzten Jahren, gerade im Zuge der HTML5-Standardisierung, wurden Wege gesucht, Webapplikationen mit mehr Möglichkeiten auszustatten und eine Reihe neuer Technologien wurde eingeführt.

Im Folgenden sollen einige Webtechnologien beschrieben werden, die für die Umsetzung einer Peer-to-Peer-Kollaborationslösung in Frage kommen und die in aktuellen Browsern bereits verfügbar sind. Sie dienen als Basis für die zu entwickelnde Architektur. Da sich einige der angeprochenen Technologien noch in der Standardisierungsphase befinden, können sich die hier angegebenen Informationen in Zukunft ändern.

JavaScript

JavaScript (bzw. ECMAScript) ist zum gegenwärtigen Zeitpunkt die einzige standardisierte Möglichkeit, um clientseitige Logik in Webapplikationen zu realisieren. Version 5 des EMCAScript Standards wird von den gängigen Browsern größtenteils implementiert. Mittlerweile ist Version 6 standardisiert und Version 7 in Arbeit. Deren Features werden aber bisher von keinem Browser vollständig unterstützt [ZA15].

WebRTC

Bei WebRTC handelt es sich um einen, in Entwicklung befindlichen, W3C-Standard (Working Draft), der es verschiedenen Instanzen von Webapplikationen ermöglicht, über eine direkte Verbindung miteinander zu kommunizieren und Medieninhalte zu senden [vgl. Ber+15]. WebRTC definiert hierfür eine JavaScript-API, die Methoden zum herstellen und Verwalten von Direktverbindungen zur Verfügung stellt.

Bisherige Browsertechnologien wie XMLHttpRequest (Ajax) oder WebSockets ermöglichen nur die Kommunikation mit einem Webserver. Daten die direkt zwischen zwei Webapplikationen ausgetauscht werden sollen, müssen daher auch immer über einen solchen Server laufen. Gerade im Falle von Telekommunikationsanwendungen (sowohl Audio als auch Video) macht jedoch eine direkte Verbindung Sinn, da diese Latenzen verringert und zentrale Server entlastet. Für diese Zwecke wurde WebRTC geschaffen, dass sich auch an bestehenden Lösungen aus dem Telekommunikationsbereich orientiert, aber auch den

Austausch beliebiger Daten ermöglicht. Da WebRTC zum momentanen Zeitpunkt die einzige standardisierte Möglichkeit für Direktverbindungen zwischen Webapplikationen darstellt, wird es im Architekturentwurf für den Großteil der Kommunikation eingesetzt.

WebRTC stellt eine der Grundlagen dar, auf denen diese Arbeit basiert. Daher sollen in den folgenden Unterabschnitten einige Aspekte der Technologie genauer betrachtet werden, unter anderem, wie sie mit Herausforderungen und Beschränkungen bei der Realisierung von Peer-to-Peer-Kommunikation im Webbrowser umgeht.

Sicherheit

Webapplikationen direkten Zugriff auf den TCP/IP-Stack des Betriebssystems zu geben wäre aufgrund von Sicherheitsimplikationen problematisch. Eine Webapplikation könnte beispielsweise, vom Nutzer unbemerkt, auf Dienste des lokalen Netzwerks zugreifen. Daher wird die benötigte Funktionalität gekapselt und auf bestimmte Anwendungsfälle beschränkt. Das direkte Herstellen einer TCP-Socketverbindung oder das versenden von UDP-Datagrammen ist mit WebRTC nicht möglich. Zudem ist in den meisten Implementierungen von WebRTC die Verwendung einer Transportverschlüsselung obligatorisch.

NAT und Firewalls

WebRTC ist primär für den Einsatz im Internet konzipiert worden. Während Webserver in der Regel über eine öffentliche IP-Adresse erreichbar sind, verfügen die Endgeräte der Nutzer oftmals nicht über eine solche Adresse, sondern teilen sich dieselbe Adresse mit anderen Endgeräten mittels einer Network Adress Translation (NAT). In IPv6-Netzwerken besitzen die Endgeräte zwar meist eine öffentliche Adresse, jedoch blockiert oftmals eine Firewall den Zugriff von außen. Um trotzdem eine direkte Verbindung zwischen Endgeräten zu ermöglichen, orientiert sich WebRTC an etablierten Peer-to-Peer-Lösungen und setzt auf Technologien wie ICE und STUN. Dabei werden, teilweise unter Zuhilfenahme von externen Servern, öffentliche IP-Adressen ermittelt und Ports einer lokalen Firewall auf das entsprechende Endgerät umgeleitet.

WebRTC ermöglicht auch einen Einsatz in lokalen Netzwerken, was für das in dieser Arbeit angedachte Einsatzszenario interessant ist. Im lokalen Netzwerk entfällt das Umgehen einer Firewall. Da WebRTC den lokalen Fall aber nicht explizit berücksichtig, müssen zur Vermittlung einer Direktverbindung dennoch die im folgenden Abschnitt beschriebenen Schritte durchgeführt werden.

Vermittlung

Bevor eine direkte Verbindung hergestellt werden kann, müssen die Endgeräte erst einige Informationen, darunter IP-Adressen, Protokolle oder zu nutzende Ports, austauschen. Hierfür verwendet WebRTC das Session Description Protocol (SDP), das beispielsweise auch bei SIP zum Einsatz kommt. Die SDP-Daten müssen über einen anderen Kommunikationskanal (Signaling-Channel) versandt werden. Dieser Kanal ist im WebRTC-Standard nicht genauer spezifiziert, einzige Bedingung ist die Möglichkeit, bidirektional Daten auszutauschen. In der Regel kommen für das Signaling Technologien wie Ajax oder WebSockets zum Einsatz, die eine einfache Kommunikation mit dem Webserver ermöglichen.

Über den Signaling-Channel handeln zwei Parteien eine Direktverbindung aus. Eine der beiden generiert ein SDP-Offer, welches unter anderem Informationen bezüglich der zu verwendenden IP-Adressen, Ports und Sicherheitsmaßnahmen enthält. In Abbildung 4.1 ist beispielhaft ein SDP-Offer abgebildet, das in Mozilla Firefox generiert wurde. Dieses Offer wird per Signaling-Channel an die zweite Partei übertragen, die eine passende SDP-Answer generiert. Diese wird wiederum per Signaling-Channel an die erste Partei gesendet. Verfügen beide Parteien über alle nötigen Informationen, wird eine Verbindung hergestellt. Informationen über IP-Adressen und Ports, die für die Kommunikation zur Verfügung stehen (sogenannte Interactive Connectivity Establishment (ICE)-Candidates), können auch nachträglich und asynchron über den Signaling-Channel übertragen werden.

```
v=0
o=mozilla...THIS_IS_SDPARTA240.0a2 6127538818722111706 0 IN IP4 0.0.0.0
s=#
t=0 0
a=sendrecv
a=fingerprint:sha.
256 A9:E3:18:B5:AA:90:A1:81:78:2D:D6:66:50:8F:1F:89:2F:FC:04:C6:5D:45:C8:B1:B9:5E:4F:F6:89:90:1B:5D
a=group:BUNDLE sdparta_0
a=ice&options:trickle
a=msid'semantic:WMS *
m=application 51552 DTLS/SCTP 5000
c=IN IP4 141.105.102.62
a=candidate:0 1 UDP 2130444543 192.168.0.110 51552 typ host
a=candidate:1 1 UDP 1694302207 141.105.102.62 51552 typ srflx raddr 192.168.0.110 rport 51552
a=sendrecv
a=end&of&candidates
a=ice&pwd:6559471a16dcb031a269ce7227a93fc9
a=ice&ufrag:a1c22fbf
a=mid:sdparta_0
a=sctpmap:5000 webrtc/datachannel 256
a=setup:actpass
```

Abbildung 4.1: SDP-Beispiel

Basierend auf der ausgehandelten Verbindung können weitere Kommunikationskanäle für die konkreten Anwendungsfälle geöffnet werden. Hierzu gehören Audio- und Videostreams und ein Kanal für den generischen nachrichtenbasierten Datenaustausch, genannt Data Channel. Über Data Channels können sowohl Strings als auch Binärdaten versandt werden.

Performance und Zuverlässigkeit

Je nach Anwendungsfall spielen bei Peer-to-Peer-Anwendungen die Performance und die Zuverlässigkeit der Datenverbindung unterschiedliche Rollen. Bei Voice over IP steht oft die Geschwindigkeit der Übertragung im Vordergrund, weshalb das verbindungslose UDP zum Einsatz kommt. Kleinere Fehler bei der Übertragung fallen hierbei in der Praxis weniger auf. In anderen Fällen, wie zum Beispiel bei der Übertragung von Dokumenten, ist eine vollständige und korrekte Übertragung notwendig. Daher verwendet man hier normalerweise zuverlässige Protokolle wie TCP oder implementiert selbst eine entsprechende Fehlererkennung und -korrektur.

WebRTC geht einen Mittelweg und setzt für den Data Channel auf das relativ unbekannte Stream Control Transmission Protocol (SCTP), ein weiteres Transportprotokoll neben TCP und UDP. Dieses kann je nach Verwendungszweck konfiguriert werden, die Integrität und Reihenfolge der gesendeten Datenpackete zu gewährleisten oder auch nicht. Für Audio- und Videostreams kommt wird weiterhin UDP verwendet.

Verbreitung

WebRTC wird bisher nur von einem Teil der gängigen Webbrowser implementiert [Dev15]. Darunter sind sowohl Desktopbrowser als auch solche für mobile Geräte. Microsoft hat die Unterstützung eines kompatiblen Standards angekündigt [Mic14]. Die bisherige Adaption von WebRTC lässt darauf schließen, dass es in Zukunft noch weitere Verbreitung finden wird.

WebSockets

Für Kommunikation zwischen den Systemkomponenten soll WebRTC zum Einsatz kommen. Wie bereits erläutert, benötigt WebRTC allerdings bestimmte Informationen, die im SDP-Format ausgetauscht werden, um eine Verbindung überhaupt herstellen zu können. SDP-Nachrichten sind nur bedingt menschenlesbar und oftmals relativ umfangreich.

Außerdem müssen für jede einzelne Verbindung zwei neue SDP-Nachrichten generiert werden, was bedeutet, dass die Informationen kurzlebig sind und nicht wiederverwendet werden können. Ein manueller Austausch von SDP-Nachrichten durch Menschen macht dementsprechend wenig Sinn. Daher ist für die initiale Aushandlung einer Verbindung eine zentrale Instanz notwendig. Weil diese zentrale Instanz dem Ziel der Unabhängigkeit von zentraler Infrastruktur widerspricht, sollte die Kommunikation darüber so gering wie möglich gehalten werden.

Für den Austausch der SDP-Nachrichten über eine zentrale Instanz kommen prinzipiell alle klassischen Möglichkeiten infrage, mittels derer eine Webapplikation mit einem Webserver kommunizieren kann. WebRTC-Datachannels arbeiten nachrichtenbasiert und es liegt nahe, für die Kommunikation mit der zentralen Komponente eine Technologie zu wählen, die ähnliche Paradigmen verwendet, weil dadurch der Implementierungsaufwand verringert wird. Eine solche Möglichkeit bietet das **WebSocket**-Protokoll.

WebSocket ermöglicht einen bidirektionalen asynchronen Datenaustausch auf Basis einer bestehenden HTTP-Verbindung. Hierfür wird vom Client ein spezieller HTTP-Header gesendet. Akzeptiert der Server die Anfrage, antwortet er dementsprechend ebenfalls mit einem speziellen Header. Daraufhin wird die TCP-Verbindung, welche der HTTP-Sitzung zugrund lag, aufrechterhalten und für den asynchronen Datenaustausch genutzt [FM11].

Alle gängigen Browser unterstützen das WebSocket-Protokoll und stellen eine entsprechende JavaScript-Schnittstelle bereit [Dev15], welche zudem als Vorlage für die entsprechende WebRTC-Schnittstelle dient [Ber+15]. Daher scheint WebRTC für die geplante Architektur geeignet und wird dort zum Einsatz kommen.

Persistente Datenhaltung

Da bei einem browserbasierten Peer-to-Peer-System die Anwendungslogik im Browser selbst läuft, liegt es nahe, dass dort auch eine lokale und persistente Datenhaltung von Nöten ist. Lange Zeit waren Cookies die einzige Möglichkeit einer Speicherung von Daten über Browsersessions hinweg, die allerdings starken Einschränkungen unterlag. Moderne Browser bieten verschiedene Möglichkeiten einer persistenten Speicherung von größeren Datenmengen, die den folgenden Abschnitten betrachtet werden.

Web Storage

Die Web Storage API ist Teil von HTML5 und stellt eine mächtigere Alternative zu Cookies dar. Sie erlaubt es Webapplikationen, clientseitig Daten in Form von Schlüssel-Wert-Paaren abzuspeichern. Die Lebensdauer der Daten hängt davon ab, ob sich der Entwickler für einen sitzungsgebundenen Speicher entscheidet (sessionStorage) oder einen sitzungsübergreifenden (localStorage) [Hic15].

Die Web Storage API ist sehr rudimentär und dementsprechend einfach zu verwenden. Es ist keine Initialisierung oder Konfiguration notwendig. Allerdings unterliegt sie auch gewissen Beschränkungen. Sie ist weniger geeignet für die persistente Speicherung größerer Datenmengen, da erstens Daten nur in Form von Strings abgespeichert werden können und zweitens gängige Browser den zur Verfügung stehenden Speicherplatz pro Webapplikation auf wenige Megabyte begrenzen. Auch aus Performancesicht betrachtet sind große Datenmengen ein Problem, da die Web Storage API nur ein synchrones Interface zur Verfügung stellt, welches den restlichen Programmablauf während des Datenabrufs blockiert. Daher scheint sie für die geplante Architektur weniger geeignet.

IndexedDB

Die Indexed Database API ermöglicht Webapplikationen den Zugriff auf eine lokale Datenbank, welche Daten unter einem bestimmten Schlüssel speichert und zusätzlich Indizes verwaltet, die den schnellen Zugriff auf Werte anhand bestimmter Eigenschaften ermöglicht [Meh+15].

Im Vergleich zu der in Abschnitt 4.2 vorgestellten Web Storage API, ist IndexedDB komplexer. Datenbanken und Indizes müssen durch den Entwickler initialisiert und verwaltet werden. Ein Vorteil ist die höhere Performance, da eine echte Datenbank auf Basis von B-Bäumen verwendet wird [Meh+15], die eine Suche von Elementen in logarithmischer Zeit ermöglicht. Zudem bietet IndexedDB ein asynchrones Interface auf Basis von Callbacks, sodass Abfragen parallel zur Ausführung des Anwendungscodes laufen können.

Die bisherigen Implementierungen von IndexedDB sehen keine oder sehr großzügige Beschränkungen der maximalen Größe einer Datenbank vor [Moz13]. Beliebige Datentypen können gespeichert werden. Daher eignet sich IndexedDB auch für größere Datenmengen. JavaScript-Objekte werden mittels des „Structured Clone Algorithmus“ [Ber+14, Kapitel 2.7.5][Meh+15] serialisiert, der einige Vorteile gegenüber einer JSON-Serialisierung besitzt, darunter die Unterstützung von mehr Datentypen und zyklischen Referenzen [Moz15c].

IndexedDB wird nicht von allen gängigen Webbrowsern vollständig implementiert. Allerdings ist die Schnittmenge mit den Browsern die den WebRTC-Standard implementieren groß [Dev15]. Wird WebRTC vorausgesetzt, stellt die Unterstützung von IndexedDB also kein Problem dar.

Aufgrund der oben genannten Vorteile, erscheint IndexedDB im Gegensatz zur Web Storage API geeigneter, die gesteckten Ziele zu erreichen. Dementsprechend kommt es in der geplanten Softwarearchitektur zum Einsatz.

File API

Da die Kollaborationsplattform sich in bestehende Workflows integrieren und mit anderen Anwendungen zusammenarbeiten können soll, sollte eine darauf basierende Anwendung auf Dateien im lokalen Dateisystem des Nutzers zugreifen können. HTML5 spezifiziert hierfür eine File API, die Webapplikationen genau das ermöglichen soll. Allerdings ist der Zugriff aus Sicherheitsgründen stark reglementiert. Lokale Dateien können nur dann gelesen werden, wenn der Nutzer diese explizit durch eine entsprechende Interaktion auswählt [RS15]. Trotz dieser Einschränkungen erlaubt die File API somit den Import von Dateien aus anderen Softwarelösungen in eine Webapplikation.

Daten können mittels der File API sowohl in Textform als auch in Binärform gelesen und verarbeitet werden. Theoretisch ist somit die Verarbeitung beliebiger Dateiformate möglich (vgl. Abschnitt 4.1).

Ein Schreiben von Dateien und somit der Export von in der Webapplikation generierten Inhalten ist über die File API nicht möglich. Allerdings bietet HTML5 über ein neues Attribut des Anker-Elements die Möglichkeit, lokal generierte Inhalte unter einem bestimmten Dateinamen herunterzuladen [RS15].

Typed Arrays

Bis vor einiger Zeit bot JavaScript keine effizienten Möglichkeiten, um rohe Binärdaten zu verarbeiten. Viele moderne Anwendungen bedürfen jedoch einer solchen Möglichkeit. Aus diesem Grund wurden Typed Arrays eingeführt. Dabei werden Binärdaten in einem sogenannten ArrayBuffer gespeichert. Dieser enthält nur die Rohdaten, schreibt aber kein bestimmtes Format vor und bietet selbst keine Möglichkeit auf die Daten zuzugreifen oder sie zu manipulieren. Um auf Daten zuzugreifen, muss auf sogenannte Views zurückgegriffen werden. Diese implementieren ein Array-ähnliches Interface zum Zugriff auf die

Daten und sind auf einen bestimmten Datentypen festgelegt [Moz15a]. Abbildung 4.2 verdeutlicht diesen Zusammenhang.

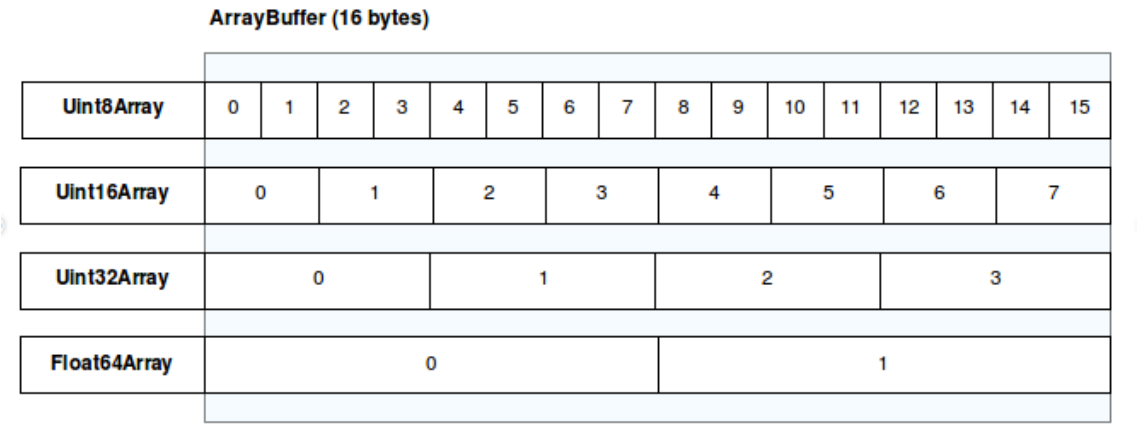

Abbildung 4.2: JavaScript Typed Arrays [Moz15a]

Typed Arrays ermöglichen den effizienten Zugriff auf und die Manipulation von binären Daten. Da die zu entwickelnde Architektur in der Lage sein soll mit solchen Daten umzugehen, werden Typed Arrays in diesen Fällen zum Einsatz kommen.

4.3 Architektur- und Entwurfsmuster

Entwurfsmuster sind ein gängiges Hilfsmittel der Softwareentwicklung. Sie ermöglichen es, wiederkehrende Probleme mit generalisierten Ansätzen zu lösen. Entwurfsmuster sind in der Regel praxiserprobt und verringern so die Gefahr von Designschwächen. Daher ist es sinnvoll, bei der Gestaltung der zu entwickelnden Softwarearchitektur in Frage kommende Entwurfsmuster zu berücksichtigen. Im Folgenden sollen einige Architektur- und Entwurfsmuster betrachtet werden, die sich für den Einsatz in der geplanten Architektur eigenen.

4.3.1 Model View Presenter

In Abschnitt 4.1 wurde der Anspruch einer strikten Trennung von Datenrepräsentation und -darstellung bereits erläutert. Eine solche Trennung ist in vielen interaktiven Softwaresystemen von Bedeutung und ein gängiges Entwurfsmuster, welches dieser Anforderung Rechnung trägt ist Model View Controller (MVC). Hierbei sind die Komponenten, welche Daten und Geschäftslogik kapseln (Model), getrennt von den Komponenten, wel-

che sich um die Darstellung der Daten für den Nutzern kümmern (View). Eine weitere Komponente (Controller) kümmert sich um die Steuerung von und Vermittlung zwischen View und Model. Je nach Ausprägung muss bei MVC allerdings eine View explizites Wissen über die Struktur des Models haben, um auf dessen Daten zugreifen zu können. Zudem enthält eine View in der Regel eigene Logik. In einer Webapplikation wird die Darstellung mittels HTML realisiert. HTML sieht an sich keine eigene Logik vor (auch wenn diese mittels JavaScript realisiert werden kann).

Um eine losere Kopplung der Komponenten zu erreichen und die notwendige Logik in der View zu minimieren, kommt in der geplanten Architektur eine MVC-Variante namens Model View Presenter (MVP) zum Einsatz. Hierbei interagieren View und Model nicht direkt miteinander. Stattdessen existiert eine Presenter-Komponente, die zwischen diesen beiden vermittelt. Die Beziehung zwischen Model, View und Presenter ist in Abbildung 4.3 dargestellt.

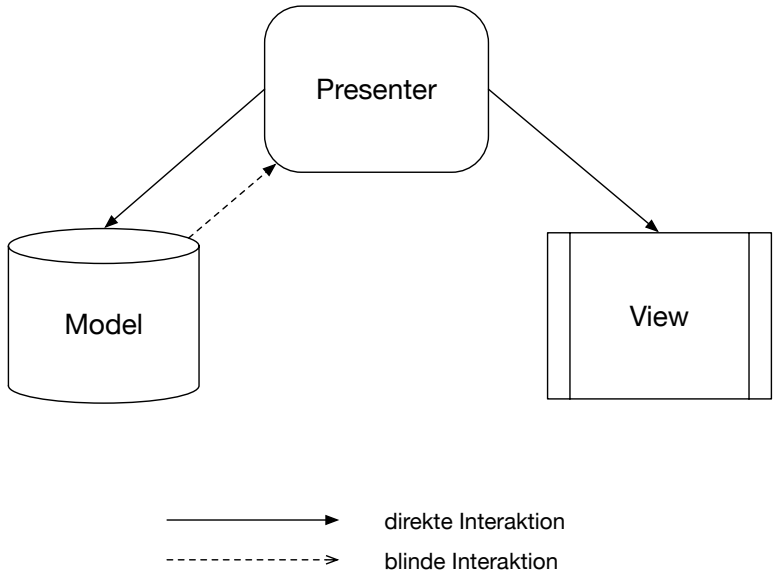

Abbildung 4.3: Das Model View Presenter Entwurfsmuster

Bei MVP ist die View passiv, das heißt sie enthält keine eigene Logik und hat keine Kenntnis von Model oder Presenter. Stattdessen ist der Presenter dafür verantwortlich die View falls notwendig anzupassen und auf Nutzerinteraktionen zu reagieren. Der Presenter übernimmt dementsprechend jegliche Interaktion mit dem Model. Das Model hat allerdings selbst keine Kenntnis vom Presenter. Stattdessen kommuniziert es mit dem Presenter nur über definierte Schnittstellen (blinde Kommunikation, siehe auch Observer-Pattern im nächsten Abschnitt)

4.3.2 Observer-Pattern

Um eine lose Kopplung zwischen Komponenten, wie beispielsweise Model und Presenter, zu erreichen bietet sich das Observer-Entwurfsmuster an. Das Observer-Pattern ermöglicht es einer Komponente (Observer), über Änderungen in einer anderen Komponente informiert zu bleiben, ohne diese aktiv in regelmäßigen Abständen überprüfen zu müssen (Polling). Die beobachtete Komponente informiert den Observer selbstständig, sobald eine Änderung auftritt. Da diese Kommunikation über eine definierte Schnittstelle abläuft, muss die beobachtete Komponente keine Kenntnis vom Observer haben. Es handelt sich um blinde Kommunikation.

Eine Möglichkeit, blinde Kommunikation zu realisieren sind Eventsysteme. Hierbei löst eine Komponente Events aus (Event Emitter) und andere Komponenten können bei Bedarf auf diese Events reagieren (Event Listener). JavaScript hat starke Anleihen von funktionalen Programmiersprachen. Beispielsweise können Funktionen als Parameter an andere Funktionen übergeben werden (Funktionen höherer Ordnung). Daher eignet sich die Sprache sehr gut zur Umsetzung eines Eventsystems mittels Callback-Funktionen. Zahlreiche Browser-APIs basieren bereits auf diesem Prinzip. Es erscheint daher sinnvoll, auch in der geplanten Architektur das Observer-Entwurfsmuster mittels Events umzusetzen.

4.3.3 Schichtenarchitektur

Eine Anforderung an die zu entwickelnde Architektur ist eine hohe Flexibilität und Wiederverwendbarkeit der Komponenten. Eine verbreitete und bewährte Möglichkeit, Komponenten eines Softwaresystems so zu strukturieren, dass gegenseitige Abhängigkeiten reduziert werden, sind Schichtenarchitekturen.

Eine klassische 3-Tier-Architektur wird den Anforderungen moderner verteilter und interaktiver Anwendungen allerdings nicht gerecht. Entwurfsmuster wie MVP eignen sich hier besser (vgl. Abschnitt 4.3.1). Dennoch kann die Organisation bestimmter Systemkomponenten in Schichten den Entwicklungsaufwand eines komplexen Softwaresystems verringern und die Wartbarkeit erhöhen. Aus diesem Grund werden diejenigen Komponenten der Webbapplikation, die nicht direkt mit der Nutzerinteraktion zusammenhängen in Schichten organisiert.

4.4 Peer-to-Peer-Netzwerk

Im Folgenden werden die Grundlagen der Kommunikation zwischen den verteilten Systemkomponenten innerhalb der geplanten Architektur erläutert. Da die Anwendung so weit wie möglich unabhängig von zentraler Infrastruktur sein soll, läuft der Großteil der Kommunikation über Peer-to-Peer-Verbindungen ab. Die Instanzen der Anwendung bilden ein Peer-to-Peer-Netzwerk, wobei jede Instanz einen Knoten innerhalb dieses Netzwerks darstellt.

4.4.1 Topologie des Netzwerks

Bei der Implementierung eines Peer-to-Peer-Netzwerks müssen einige grundlegende Entscheidungen getroffen werden. Ein wesentlicher Faktor ist hierbei die Netzwerktopologie. Da Peer-to-Peer-Netzwerke auf bestehenden Netzwerken, wie dem Internet, aufbauen, werden sie auch als Overlay-Netzwerke bezeichnet. Bei Peer-to-Peer unterscheidet man grundsätzlich zwischen unstrukturierten und strukturierten Overlay-Netzwerken [vgl. z.B. QB06].

In unstrukturierten Overlay-Netzwerken baut ein Knoten zufällig Verbindungen zu anderen Peers auf. Es ist keine Kenntnis über die Struktur des Netzwerks erforderlich. Ein Beispiel hierfür ist das Gnutella-Filesharing-Netzwerk. Unstrukturierte Overlay-Netzwerke sind oft einfacher zu implementieren. Da aber nicht vorausgesetzt werden kann, dass die Struktur des Netzwerks bestimmte Einschränkungen erfüllt, sind sie bezüglich einiger Aspekte weniger performant. Hierzu zählt die Suche von Inhalten im Netzwerk.

Strukturierte Overlay-Netzwerke geben bestimmte Regeln vor, welche die Verbindungen von Peers und die Verteilung von Daten einschränken. In der Regel sind Peers eindeutig identifizierbar. Das Netzwerk wird anhand der eindeutigen Kennungen der Knoten strukturiert. Ein verbreitetes Konzept für strukturierte Peer-to-Peer-Overlays sind Distributed Hash Tables (DHTs). Da die Struktur des Netzwerks bestimmte Bedingungen erfüllt, können optimierte Algorithmen, beispielsweise für die Suche, implementiert werden, die relativ geringe Laufzeiten garantieren und auch bei großen Nutzerzahlen gut skalieren (in der Regel $\mathcal{O}(\log n)$).

Für die zu entwickelnde Architektur spielen sehr große Nutzerzahlen eine untergeordnete Rolle. Allerdings ist die Stabilität und Performance des Netzwerks ein wichtiger Faktor. Dabei ist es von Vorteil, wenn sich ein System nach vorhersagbaren Regeln verhält. Deshalb fällt die Wahl auf ein strukturiertes Peer-to-Peer-Overlay. Dieses orientiert sich in Teilen an dem in [RD01] vorgestellten Pastry-Netzwerk, welches bereits in einer früheren

Ausarbeitung betrachtet wurde [Bel14]. Allerdings benötigt eine ausgewachsene DHT eine umfangreiche Routingtabelle. Das stellt im Normalfall kein Problem dar. Es reicht das Speichern der eindeutigen Kennung eines Netzwerkknotens zusammen mit seiner IP-Adresse. Da allerdings bei WebRTC vor jedem Verbindungsaufbau ein bidirektionaler Datenaustausch notwendig ist, reicht das Speichern einer statischen Information wie der IP-Adresse hier nicht aus. Stattdessen müsste für jeden Eintrag in der Routingtabelle eine offene Verbindung gehalten werden. Zudem werden viele der Eigenschaften einer DHT, wie die Skalierbarkeit bei sehr großen Nutzerzahlen, nicht benötigt. Daher verzichtet das im Folgenden beschriebene Overlay-Netzwerk auf die Implementierung einer vollständigen DHT und setzt stattdessen eine reduzierte Version dieser Methode um.

Die Topologie des Peer-to-Peer-Netzwerks ist in Abbildung 4.4 abgebildet.

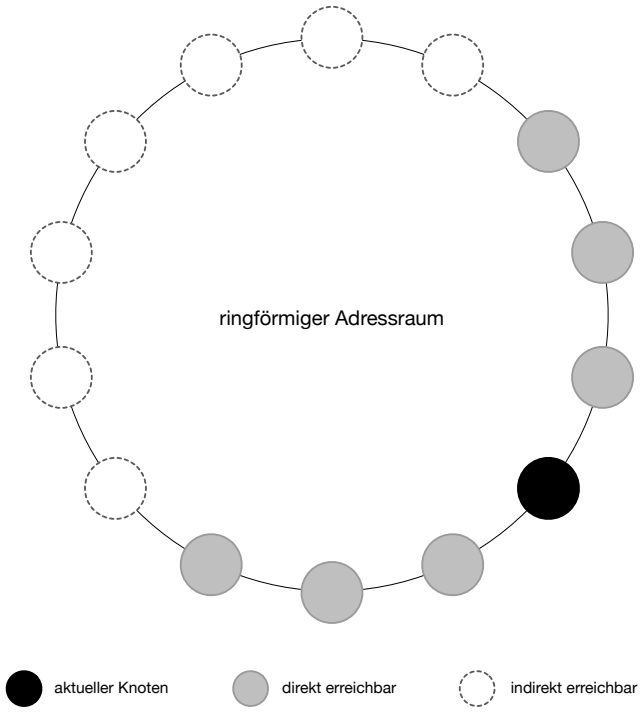

Abbildung 4.4: Topologie des Peer-to-Peer-Netzwerks

Jeder Knoten des Netzwerks erhält eine eindeutige Kennung (ID). Da es keine zentrale Instanz gibt, die diese zuteilen könnte, erzeugt jeder Knoten seine ID auf Basis einer Zufallsfunktion selbst. Um Kollisionen zu vermeiden, sollte die Länge der ID ausreichend groß gewählt werden (beispielsweise 128 Bit). Der Adressraum ist ringförmig, das heißt, es wird angenommen, dass nach der höchstmöglichen ID (alle Bits gesetzt) wieder die 0 kommt. Die Zufallsfunktion sollte gewährleisten, dass die IDs der Knoten gleichmäßig über den Adressraum verteilt werden.

Jeder Knoten hält außerdem eine Liste von nächstgelegenen numerisch größeren und kleineren Knoten vor. Diese Liste wird nach [RD01] im Folgenden „Leaf Set“ genannt. Zu den Knoten im Leaf Set besteht eine offene Verbindung, sodass diese direkt kontaktiert werden können. Alle anderen Knoten des Netzwerks sind nur indirekt erreichbar. Besteht eine Sitzung nur aus wenigen Teilnehmern, führt dieser Mechanismus dazu, dass jeder Knoten jeden anderen Knoten direkt erreichen kann. Da die maximale Anzahl der gleichzeitig offen zu haltenden Verbindungen durch die Größe des Leaf Sets begrenzt ist, skaliert das System auch für größere Nutzerzahlen, wobei dann der Aufwand für die Zustellung einer Nachricht steigt. Die Größe des Leaf Set ist somit ein wesentlicher Parameter zur Beeinflussung des Netzwerkverhaltens.

4.4.2 Initialer Verbindungsaufbau

Möchte ein Teilnehmer einer Kollaborationssitzung beitreten, so muss sein System erst eine Verbindung zum oben beschriebenen Peer-to-Peer-Netzwerk herstellen. Das Peer-to-Peer-Netzwerk muss zu Beginn der Sitzung erst aufgebaut werden. Die Interaktion der verschiedenen Systemkomponenten während des initialen Verbindungsaufbaus ist in Abbildung 4.5 dargestellt und soll im Folgenden erläutert werden.

Zu Beginn des Verbindungsaufbaus muss der neue Netzwerkknoten, also die Webapplikation des neuen Teilnehmers, eine Kennung (ID) generieren, die ihn eindeutig identifiziert (vgl. Abschnitt 4.4.1). Zudem generiert er ein SDP-Offer für eine Direktverbindung zu einem anderen Knoten. Da bisher keine Verbindung zu irgendeinem Knoten des Netzwerks besteht, muss der erste Datenaustausch über den zentralen Webserver abgewickelt werden.

In Abschnitt 4.2 wurden bereits WebSockets als Kommunikationskanal zwischen Webapplikation und Webserver ausgewählt. Auf dieser Basis kommt eine Publish-Subscribe-Lösung zum Einsatz. Publish-Subscribe hat den Vorteil, dass die Kontrolle über den Nachrichtenfluss im wesentlichen bei den Instanzen der Webapplikation liegt. Der Webserver muss keine Anwendungsspezifische Logik enthalten und kann leichter ausgetauscht werden, da eine Vielzahl von Publish-Subscribe-Lösungen existiert, die hierfür in Frage kommen.

Der neue Knoten stellt somit eine Verbindung zum Webserver her und abonniert ein Topic, unter dem er eine Antwort auf seine Verbindungsanfrage erwartet (answertopic). Danach publiziert er seine ID und sein SDP-Offer unter einem Topic, das alle möglicherweise bereits vorhandenen Knoten abonniert haben (offertopic). Erhält er bis zum Ablauf eines festgelegten Timeouts keine Antwort, so kann davon ausgegangen werden, dass bis-

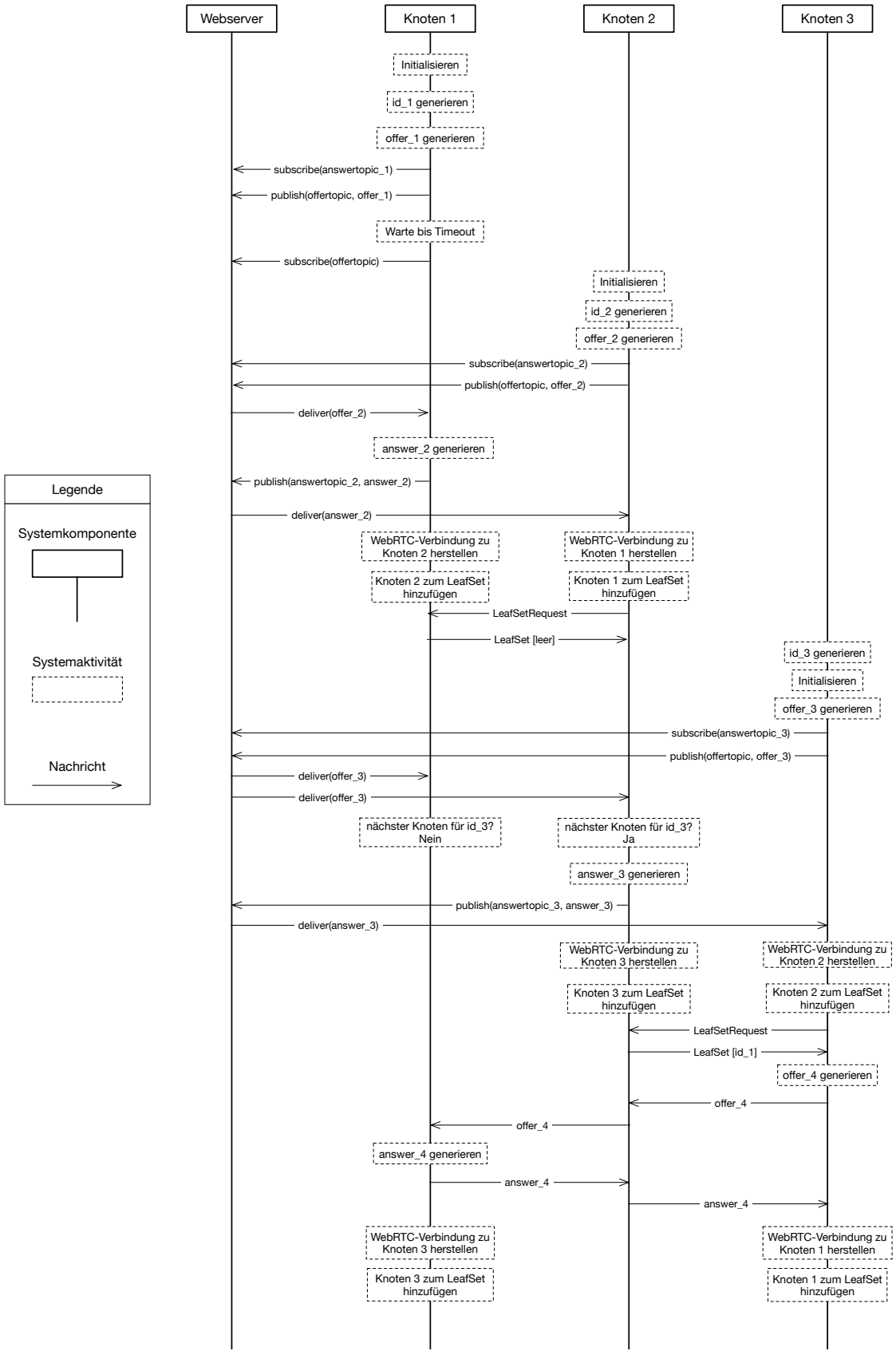

Abbildung 4.5: initialer Verbindungsaufbau zum Peer-to-Peer-Netzwerk

her keine anderen Knoten im Netzwerk vorhanden sind. Der Knoten abonniert das Topic für neue Anfragen (offertopic) und erwartet die Verbindungsanfrage eines weiteren Knoten.

Sind bereits andere Knoten im Netzwerk vorhanden, so erhalten diese von der Publish-Subscribe-Middleware die Verbindungsanfrage des neuen Knotens. Derjenige Knoten dessen ID den geringsten numerischen Abstand zu der des neuen Knotens besitzt ist dafür verantwortlich zu antworten. Ein Knoten kann nur auf Basis seiner lokalen Daten entscheiden, ob er der numerisch nächste Knoten ist, indem er überprüft, ob in seinem Leaf Set ein näherer Knoten existiert. Ist dies nicht der Fall, generiert er eine passende SDP-Answer zum SDP-Offer des neuen Knotens und publiziert es auf dessen spezifischem Antworttopic (answertopic). Der neue Knoten erhält die Antwort und kann darauf eine Direktverbindung zum anderen Knoten herstellen. Um die Kommunikation mit der zentralen Komponente so weit wie möglich einzuschränken, werden alle weiteren notwendigen Schritte über diese Direktverbindung ausgehandelt.

Hat ein Knoten die erste Verbindung zu einem anderen Knoten hergestellt ergänzen beide ihr Leaf Set um den jeweils anderen Knoten. Zusätzlich benötigt der neue Knoten noch Informationen zu weiteren Knoten innerhalb des Netzwerks. Daher sendet er eine Nachricht, welche den anderen Knoten dazu veranlasst ihm eine Kopie der IDs in seinem Leaf Set zukommen zu lassen. Weil es sich bei dem anderen Knoten um den numerisch nächsten handelt, enthält dessen Leaf Set bereits alle Knoten, die auch der neue Knoten benötigt, um sein Leaf Set zu füllen. Für jeden Knoten der Liste generiert der neue Knoten ein entsprechendes SDP-Offer und versendet es über die bestehende Direktverbindung, was den anderen Knoten dazu veranlasst, es an den entsprechenden Empfänger weiterzuleiten. Der Empfänger generiert eine Antwort und sendet sie auf dem umgekehrten Weg zurück. Sobald eine Direktverbindung hergestellt ist, erweitern beide Knoten wieder ihr Leaf Set.

4.4.3 Routing von Nachrichten

Das Routing einzelner Nachrichten ist in Abbildung 4.6 abgebildet und läuft nach dem in diesem Abschnitt beschriebenen Schema ab.

Jede Nachricht enthält die IDs des Absenders und des gewünschten Zielknotens. Möchte ein Knoten eine Nachricht an einen anderen Knoten versenden, so ermittelt er zuerst, ob der Zielknoten Teil seines Leaf Sets und somit für ihn direkt erreichbar ist. Ist dies der Fall, wird die Nachricht über die bestehende Direktverbindung zu diesem Knoten versandt. Falls der Zielknoten nicht direkt erreicht werden kann, ermittelt der Absender den

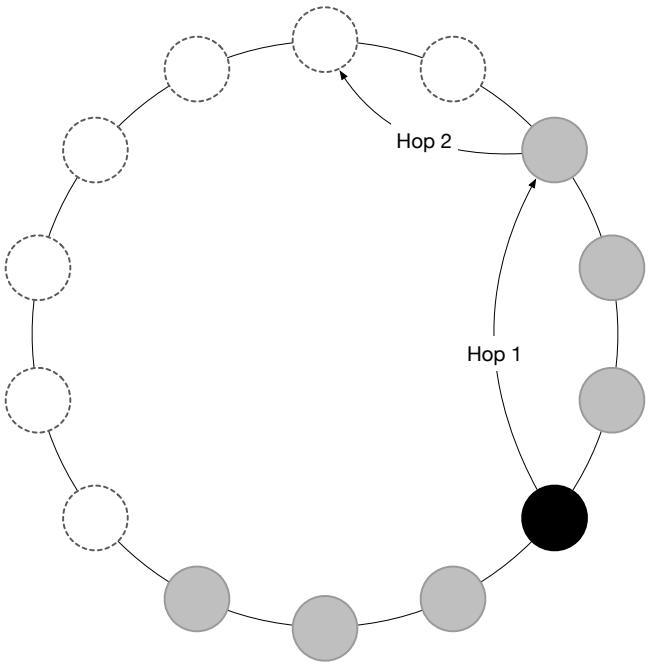

Abbildung 4.6: Routing von Nachrichten

Knoten in seinem Leaf Set, dessen ID der des Zielknotens numerisch am nächsten ist. Die Nachricht wird nun über die bestehende Direktverbindung an diesen Knoten weitergeleitet. Erhält ein Knoten eine Nachricht, die nicht für ihn selbst bestimmt ist, verfährt er nach dem gleichen Muster weiter.

Die Übergänge einer Nachricht von einem Knoten zum nächsten werden als Hops bezeichnet. Bei einer Gesamtmenge von Knoten N und einer Größe des Leaf Sets L beträgt die maximale Anzahl an Hops für die Zustellung einer Nachricht $\lceil \frac{(N-1)}{(L/2)} \rceil$, wenn man eine suboptimale Verteilung der IDs im Adressraum vorrausgesetzt. Dieser Term wächst zwar linear (bzw. durch die Rundung stufenförmig) mit der Anzahl der Knoten N, allerdings mit einer so geringen Steigung, dass dies für die zu erwartenden Nutzerzahlen einer Kollaborationsanwendung kaum eine Rolle spielt.

4.4.4 Broadcast

Neben der Zustellung von Einzelnachrichten, spielt bei einer Kollaborationsanwendung auch das Broadcasting, das heißt der Versand einer Nachricht an alle Knoten, eine wesentliche Rolle. In vielen Fällen müssen Änderungen der gesamten Teilnehmerschaft mitgeteilt werden können. Die Verwendung des normalen Routingmechanismus für diesen Zweck ist zwar möglich, würde aber zu einem unnötigen Overhead führen, da sie selbe

Nachricht mehrfach über die selben Knoten weitergeleitet werden müsste. Daher soll hier ein alternativer Ansatz gefunden werden.

Es existiert eine Vielzahl von Algorithmen zur Umsetzung von Broadcasts in Peer-to-Peer-Netzwerken. Ein bekannter und relativ verbreiteter Ansatz ist das sogenannte „Blind Flooding“. Hierbei sendet der Ausgangsknoten die Nachricht an alle Knoten, die ihm bekannt sind. Erhält ein Knoten die Nachricht, leitet er sie wiederum an alle ihm bekannten Knoten weiter. Um eine endlose Weiterverbreitung der Nachricht zu verhindern, kann die Nachricht beispielsweise mit einem Time-to-live-Zähler ausgestattet werden, welcher bei jedem Hop dekrementiert wird. Zudem können die Knoten feststellen, ob sie eine Nachricht zum wiederholten Male erhalten und diese verwerfen.

Wie sich unschwer erkennen lässt, erzeugen Blind Flooding Algorithmen einen großen Overhead an Nachrichten. Daher wurde eine alternativer Algorithmus entworfen, der sich die Struktur des Netzwerks zu Nutze macht. Die Zustellung einer Broadcastnachricht mittels dieser Methode ist in Abbildung 4.7 dargestellt und soll im Folgenden erläutert werden.

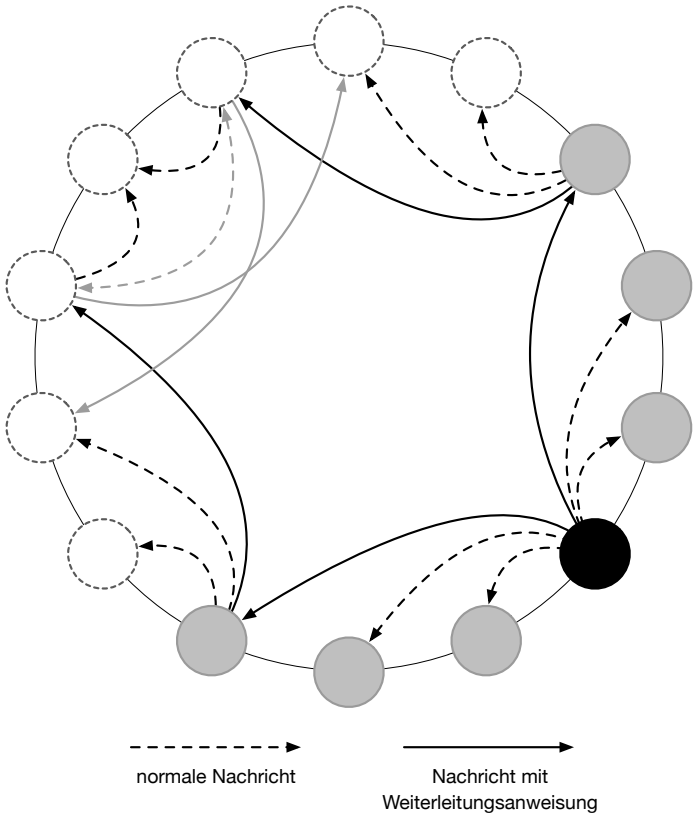

Abbildung 4.7: Broadcast einer Nachricht

Möchte ein Knoten eine Nachricht an alle Knoten des Netzwerks versenden, so sendet er sie über eine Direktverbindung an alle Knoten seines Leaf Sets. Im Normalfall wird die

Nachricht durch diese Knoten sofort verarbeitet und nicht weitergeleitet. Die äußersten beiden Knoten erhalten jedoch eine erweiterte Version der Nachricht, die neben dem eigentlichen Inhalt eine Anweisung zur Weiterleitung enthält. In dieser Anweisung wird die Richtung spezifiziert, in der die Nachricht weitergeleitet werden soll. Erhält ein Knoten solch eine erweiterte Nachricht, leitet er sie nach demselben Schema wie der Ursprungsknoten weiter, allerdings berücksichtigt er hierbei nur die Seite seines Leaf Sets, die in der Weiterleitungsanweisung spezifiziert war.

Zum Ende des Prozesses erhalten einige Knoten die Nachricht zweimal, was in Abbildung 4.7 durch ausgegraute Pfeile dargestellt ist. Daher werden die Nachrichten ebenfalls mit eindeutigen Kennungen (IDs) versehen, welche der Knoten für eine gewisse Zeit speichert. Stellt ein Knoten fest, dass er eine Nachricht zum wiederholten Male erhält, verwirft er diese.

4.4.5 Nachrichtenformat

WebRTC Datachannels unterstützen, genau wie WebSockets, die Übertragung verschiedener Datenformate, darunter Strings und binäre Datentypen wie beispielsweise ArrayBuffers (vgl. Abschnitt 4.2). Da die Architektur beliebige Datentypen, darunter auch Binärformate, unterstützen soll, fällt die Wahl auf ein binäres Nachrichtenformat. Auch für den Fall, dass Daten in Textform übertragen werden, bietet diese Vorgehensweise einen Vorteil. In JavaScript werden Strings mit einem 16-Bit-Zeichensatz kodiert. Bei der Umwandlung in ein Binärformat können sie stattdessen mit einem Zeichensatz wie UTF-8 kodiert werden, der für viele Zeichen nur 8 Bit benötigt. Somit wird Bandbreite eingespart.

Abbildung 4.8 stellt das binäre Nachrichtenformat schematisch dar. Jede Nachricht enthält einen Header mit der Nachrichten-ID, der ID des Absenders und des Empfängers. Der Header enthält zusätzlich eine Angabe über den Typ der Nachricht in Form einer 8-Bit-Zahl. Ein Teil der Typnummern ist für die internen Managementnachrichten des Peer-to-Peer-Netzwerks reserviert. Die restlichen Nummern können von der Anwendung frei gewählt werden. Sollten mehr als 256 unterschiedliche Nachrichtentypen notwendig sein, so kann dieses Feld des Headers vergrößert werden.

Die eigentlichen Nutzdaten (Payload) folgen nach dem Header. Eine Nachricht kann beliebig viele Payloads mit sich führen. Eine Payload beginnt immer mit einer 8-Bit-Typangabe und einer Angabe über ihre Größe. Dann folgen die eigentliche Daten. Vorerst sind folgende Payload-Typen vorgesehen:

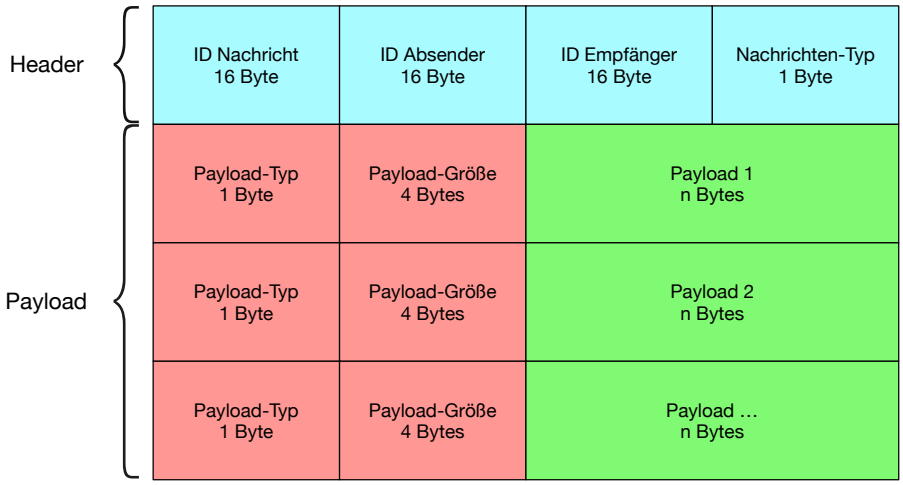

Abbildung 4.8: Das binäre Nachrichtenformat

String Eine reine Zeichenkette (UTF-8 kodiert)

Object Ein als JSON-String serialisiertes JavaScript-Objekt (UTF-8 kodiert)

ArrayBuffer Eine binäre Payload in Form eines ArrayBuffers

4.5 Sitzungsmanagement

Da die Architektur primär auf den Einsatz in lokalen Netzwerken ausgelegt ist, existiert für jede Kollaborationssitzung ein eigenes Peer-to-Peer-Netzwerk. Dennoch ist es möglich, dass mehrere Sitzungen über den selben Webserver initiiert werden. Die Sitzungen müssen somit eindeutig identifizierbar sein. Das kann erreicht werden, indem jede Sitzung, so wie bereits die Knoten innerhalb des Netzwerks, eine eindeutige Kennung erhält. Diese Kennung ist die wesentliche Information, durch die eine Kollaborationssitzung identifiziert wird. Alle anderen Informationen befinden sich auf den Knoten des Peer-to-Peer-Netzwerks. Die Publish-Subscribe-Topics auf dem zentralen Server erhalten ebenfalls ein auf der Kennung basierendes Präfix, um eine Interferenz verschiedener Sitzungen zu verhindern.

Möchte ein Teilnehmer eine neue Kollaborationssitzung öffnen, generiert die Anwendung eine zufällige Sitzungskennung. Um einer Sitzung beizutreten, müssen andere Teilnehmer die Kennung der Sitzung mitgeteilt bekommen, beispielsweise in Form einer URL, die die Kennung als Query-Parameter enthält. Nur dann können sie die entsprechenden

Topics abonnieren, um die für den initialen Verbindungsaufbau benötigten Informationen auszutauschen.

4.6 Datenbankparadigma

Die Möglichkeit einer persistenten Speicherung von Daten stellt eine wesentliche Anforderung an die Architektur dar. In Abschnitt 4.2 wurde bereits IndexedDB als geeignete Technologie hierfür ausgewählt. IndexedDB verhält sich grundsätzlich ähnlich wie ein Key-Value-Store oder eine dokumentenbasierte Datenbank. Daten werden als Werte abgespeichert, auf die mittels eines Schlüssels zugegriffen werden kann. Allerdings kann es sich bei Werten um Objekte handeln, die beliebige Datentypen enthalten. IndexedDB ermöglicht auch das Anlegen mehrerer Object Stores in einer Datenbank und die Erstellung eigener Indizes. Über diese Mechanismen könnten auch komplexere Paradigmen für den Datenzugriff implementiert werden, beispielsweise ein System, dass sich ähnlich wie eine relationale Datenbank verhält.

Für die Zwecke der geplanten Architektur erscheint ein Key-Value-Store geeignet. Er erfordert kein festes Schema, dem die Daten folgen müssten und lässt sich somit flexibel für verschiedene Arten von Kollaborationsanwendungen einsetzen. Zudem lassen sich die möglichen Operationen eines Key-Value-Stores relativ leicht auf Nachrichten abbilden, die über eine Peer-to-Peer-Verbindung ausgetauscht werden können. Daher fällt die Wahl auf dieses Paradigma. Um anderen Komponenten einen einfachen Zugriff auf persistente Daten zu ermöglichen, sollte die Kollaborationsplattform eine Schnittstelle implementieren, die typische Key-Value-Operationen auf Basis von IndexedDB umsetzt.

4.7 Datenrepräsentation

Die geplante Architektur soll das MVP-Entwurfsmuster umsetzen. Daher obliegt die Repräsentation der Daten dem Model. Dieses ist dafür zuständig, die Daten der Kollaborationsanwendung zu verwalten, deren persistente Speicherung zu veranlassen, anderen Komponenten Zugriffsmöglichkeiten auf die Daten zu bieten und die Konsistenz der Daten bei allen Teilnehmern zu gewährleisten.

Auf Basis einer Kollaborationsplattform können verschiedene Arten von Kollaborationsanwendungen umgesetzt werden. Die Art der zu repräsentierenden Daten hängt dabei stark von der jeweiligen Anwendung ab. Es kann sich beispielsweise um ein einzelnes Textdokument handeln, um eine Grafik aber auch um eine Menge von Knoten und Ver-

bindungen, die eine Mindmap bilden. Unterschiedliche Arten von Daten stellen unterschiedliche Anforderungen an die Schnittstelle, mittels derer auf die Datenrepräsentation zugegriffen wird. Und je nach Art der Daten bieten sich unterschiedliche Methoden an, die parallele Bearbeitung zu ermöglichen, Änderungen zu synchronisieren und die Konsistenz der Daten zu gewährleisten. Aus diesem Grund macht die geplante Architektur keine Vorgaben bezüglich der Schnittstellen, die das Model zur Verfügung stellen sollte, und der internen Mechanismen, derer sich das Model bedient, um die Daten zu synchronisieren.

Um den Entwicklern eines konkreten Kollaborationstools dennoch Implementierungsaufwand zu ersparen, wäre es allerdings denkbar, dass die Plattform eine Reihe generischer Modelklassen für verschiedene Anwendungsfälle zur Verfügung stellt. Diese könnten für bestimmte Arten von Daten optimiert sein und für den jeweiligen Anwendungsfall angemessene Algorithmen für die Synchronisation implementieren. Einige solcher Algorithmen wurden im Verlauf dieser Arbeit bereits angesprochen, darunter „Operational Transformation“ (Abschnitt 2.2.4) und „State Vector“ (Abschnitt 2.2.1).

4.8 Sicherheitsaspekte

Bei der Nutzung von WebRTC ist die Verwendung einer Transportverschlüsselung obligatorisch. Dementsprechend wird jedweder Datenverkehr innerhalb des Peer-to-Peer-Netzwerks verschlüsselt. Dennoch ergeben sich einige Probleme bezüglich der Datensicherheit.

Der Zugang zum Peer-to-Peer-Netzwerk anhand der oben beschrieben Prozedur (vgl. Abschnitt 4.4.2) ist sehr einfach und erfordert lediglich eine Möglichkeit mit der zentralen Komponente zu kommunizieren. Ist ein Knoten erst einmal Teil des Peer-to-Peer-Netzwerks, erhält er automatisch alle Broadcast-Nachrichten, die von anderen Knoten versandt werden. Und da die beschrieben Routingverfahren es vorsehen, dass auch Direktnachrichten über mehrere Knoten laufen können, können Teilnehmende Knoten Zugriff auf die private Kommunikation zweier Teilnehmer im Klartext erhalten und die Daten auch manipulieren. Dieses Problem ist unabhängig von einer Transportverschlüsselung, weil diese ja nur den direkten Übertragungsweg zwischen zwei Knoten betrifft.

Aus genannten Gründen sollten, zusätzlich zur vorhandenen Transportverschlüsselung von WebRTC, weitere Vorkehrungen getroffen werden, um die Vertraulichkeit und Integrität der innerhalb des Peer-to-Peer-Netzwerks ausgetauschten Daten zu gewährleisten. Im Folgenden sollen einige Vorschläge für solche Maßnahmen gemacht werden, die

verdeutlichen, dass es sich um ein lösbares Problem handelt. Die genaue Ausgestaltung und Umsetzung der vorgeschlagenen Maßnahmen liegt allerdings nicht im Fokus dieser Arbeit. Da die angesprochenen Sicherheitsaspekte im wesentlichen nur die Kommunikationskomponenten des Systems betreffen, hätten eventuelle Änderungen der Sicherheitsmaßnahmen nur geringen Einfluss auf die Gesamtarchitektur.

4.8.1 Zugriff auf das Peer-to-Peer-Netzwerk

Um den Zugriff unbefugter Dritter auf das Peer-to-Peer-Netzwerk zu unterbinden, wird ein Authentifikationsmerkmal benötigt. Um anderen Personen überhaupt die Teilnahme an einer Kollaborationssitzung zu ermöglichen, müssen ohnehin Informationen ausgetauscht werden. Dabei kann es sich um eine URL handeln, die auf den Webserver der zentralen Komponente zeigt. Diese URL könnte um einen Query-Parameter erweitert werden, der ein entsprechendes Authentifikationsmerkmal enthält. Damit potentielle Angreifer diese Information nicht mitlesen können, sollte der Webserver HTTPS verwenden.

4.8.2 Verschlüsselung von Nachrichten

Um das mitlesen von Nachrichten, auch innerhalb des Netzwerks, zu verhindern, kann der eigentliche Inhalt der Nachrichten zusätzlich verschlüsselt werden. Da der Austausch eines symmetrischen Schlüssels ebenfalls über das Peet-to-Peer-Netzwerk stattfinden müsste, bietet sich die Verwendung einer asymmetrischen Verschlüsselung (Public Key Kryptographie) an. Es wäre denkbar, dass die Knoten des Netzwerks direkt ihren öffentlichen Schlüssel als eindeutige ID verwenden [vgl. z.B. PE10, S. 470]. Auf diese Weise könnten Nachrichten über das beschriebene Routingverfahren ausgetauscht werden, ohne dass andere Knoten deren Inhalt ermitteln können.

4.9 Zentrale Infrastruktur

4.9.1 Signaling Server

Wie bereits angesprochen, kann die Architektur aufgrund der Art, wie WebRTC Verbindungen vermittelt, nicht vollständig auf eine zentrale Komponente verzichten. Diese zentrale Komponente sollte so einfach und austauschbar wie möglich gestaltet werden, um die Abhängigkeit von bestimmter Infrastruktur zu verringern. Ihre wesentliche Aufgabe

besteht darin, als Signaling Server einen Kommunikationskanal für den in Abschnitt 4.4.2 beschriebenen initialen Verbindungsaufbau zur Verfügung zu stellen. Als Protokoll für den Datenaustausch wurden in Abschnitt 4.2 WebSockets ausgewählt.

4.9.2 Webserver

Während ein Signaling Server aus technischen Gründen obligatorisch ist, existiert noch eine weitere Funktion, die optional ebenfalls von der zentralen Komponente übernommen werden könnte. Hierbei handelt es sich um die Auslieferung der Webapplikation per HTTP an die Webbrowser der Teilnehmer. Hier kollidiert die Anforderungen der minimalen Abhängikeit von zentraler Infrastruktur mit der Anforderung einer möglichst einfachen Initialisierung. Es ist auch möglich, eine Webapplikation direkt vom lokalen Dateisystem aus zu starten, was es jedoch erforderlich macht, ihnen vorher eine Kopie der notwendigen Dateien zukommen zu lassen. Für Updates der Webapplikation wäre jedes mal eine Verteilung der entsprechenden Dateien an alle Teilnehmer notwendig. Da die zentrale Komponente für die Bereitsstellung eines WebSocket-Servers ohnehin einen HTTP-Server implementieren muss, erscheint die zusätzliche Auslieferung einiger statischer Dateien per HTTP eine logische Erweiterung. Dementsprechend übernimmt die zentrale Komponente in der geplanten Architektur beide Aufgaben.

4.9.3 Physische Platzierung

Es existieren verschiedene Alternativen dafür, auf welchem physischen System die zentrale Komponente platziert wird. Einerseits kann sie auf einem dedizierten, möglicherweise auch über das Internet verfügbaren, Server laufen. Andererseits kann sie ebenso auf dem System eines Teilnehmers ausgeführt werden. Letzteres erscheint aus Sicht der Unabhängigkeit und Dezentralität wünschenswert, stellt allerdings einen Mehraufwand bei der Einrichtung der Software und Initialisierung einer Kollaborationssitzung dar. Die Serversoftware sollte für den Einsatz auf dem Endgerät eines Nutzers nur wenige Einrichtungsschritte benötigen und über ein grafisches User Interface verfügen, um den Betrieb möglichst einfach zu gestalten.

4.10 Finale Architektur

Nachdem in den vorherigen Abschnitten bereits einige Aspekte der geplanten Softwarearchitektur tiefergehend erläutert wurden, soll hier nun der finale Architekturentwurf vor-

gestellt werden. Dieser gliedert sich in die Gesamtarchitektur des verteilten Systems sowie die interne Architektur der eigentlichen Webapplikation, welche den Großteil des Systems ausmacht.

4.10.1 Architektur des verteilten Systems

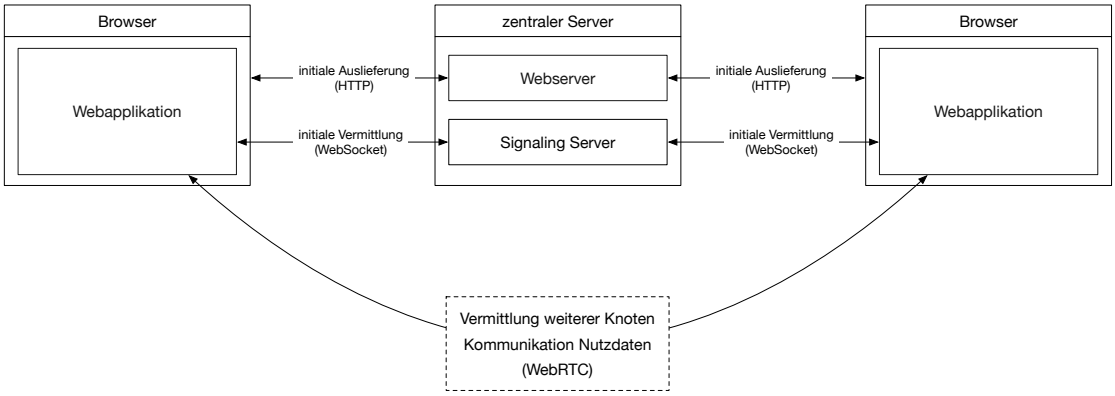

Abbildung 4.9: Architektur des Gesamtsystems

Wie bereits erläutert, besteht das verteilte System aus zwei grundlegenden Komponenten: Einem zentralen Server sowie einer Webapplikation. Während in einer Kollaborationssitzung nur eine Instanz des zentralen Servers existiert, betreibt jeder Teilnehmer einen Webbrowser mit einer eigenen Instanz der Webapplikation. Abbildung 4.9 stellt die Interaktion dieser Komponenten grafisch dar. Ihre Rolle soll an dieser Stelle kurz erläutert werden.

Zentraler Server

Der zentrale Server gliedert sich in zwei Teile, die seinen beiden Hauptfunktionalitäten entsprechen (vgl. Abschnitt 4.9). Einerseits stellt er einen Webserver zur Verfügung, welcher auf Anfrage die Webapplikation per HTTP ausliefert und somit dem Webserver erst ermöglicht, diese auszuführen. Andererseits implementiert er einen Signaling Server, der es der Webapplikation ermöglicht über eine WebSocket-Verbindung mit anderen Instanzen zu kommunizieren, um die initiale Verbindung zum Peer-to-Peer-Netzwerk herzustellen.

Der zentrale Server enthält keine wesentliche Anwendungslogik. Stattdessen stellt er nur die notwendige Kommunikationsinfrastruktur zur Verfügung. Die Interaktion der Web-

applikation mit dem zentralen Server wird, entsprechend der Anforderung der Unabhängigkeit von zentraler Infrastruktur (Abschnitt 4.1) , auf ein Minimum begrenzt.

Webapplikation

Die Webapplikation läuft im Webbrowser der Teilnehmer und realisiert die eigentliche Kollaborationsanwendung. Hier findet sich die Anwendungslogik des Systems. Die Webapplikation kommuniziert über WebRTC-Datachannels mit anderen Instanzen und bildet mit ihnen ein P2P-Netzwerk, über das die eigentlichen Nutzdaten der Anwendung übertragen werden.

Die interne Architektur der Webapplikation wird in Abschnitt 4.10.2 näher erläutert.

4.10.2 Architektur der Webapplikation

Die Architektur der Webapplikation ist in Abbildung 4.10 dargestellt. Wie bereits angesprochen, ist ein Großteil der internen Komponenten in Schichten organisiert. Eine Schicht kann jeweils nur mit niedrigeren Schichten direkt interagieren. Die Kommunikation von niedrigeren zu höheren Schichten ist blind und findet in der Regel asynchron über Event-Mechanismen statt (vgl. Abschnitt 4.3.2).

Die einzelnen Komponenten der Webapplikation werden im Folgenden näher erläutert.

Kommunikationskomponenten

Die Kommunikationskomponenten stellen die Grundlage des kollaborativen Systems dar. Sie ermöglichen verschiedenen Instanzen der Webapplikation den Austausch von Informationen und bauen zu diesem Zweck das Peer-to-Peer-Netzwerk auf. Wie in Abbildung 4.10 ersichtlich, existieren insgesamt drei Kommunikationskomponenten.

Die erste Komponente realisiert das eigentliche Peer-to-Peer-Netzwerk. Sie ist zuständig für das Routing von Nachrichten und hält die notwendigen Datenstrukturen für die Organisation des Netzwerks vor, darunter das Leaf Set. Außerdem stellt sie höheren Schichten eine Schnittstelle zur Verfügung, über die diese Nachrichten an andere Knoten versenden und Nachrichten von anderen Knoten empfangen können, ohne Kenntnisse über die Struktur des Peer-to-Peer-Netzwerks besitzen zu müssen.

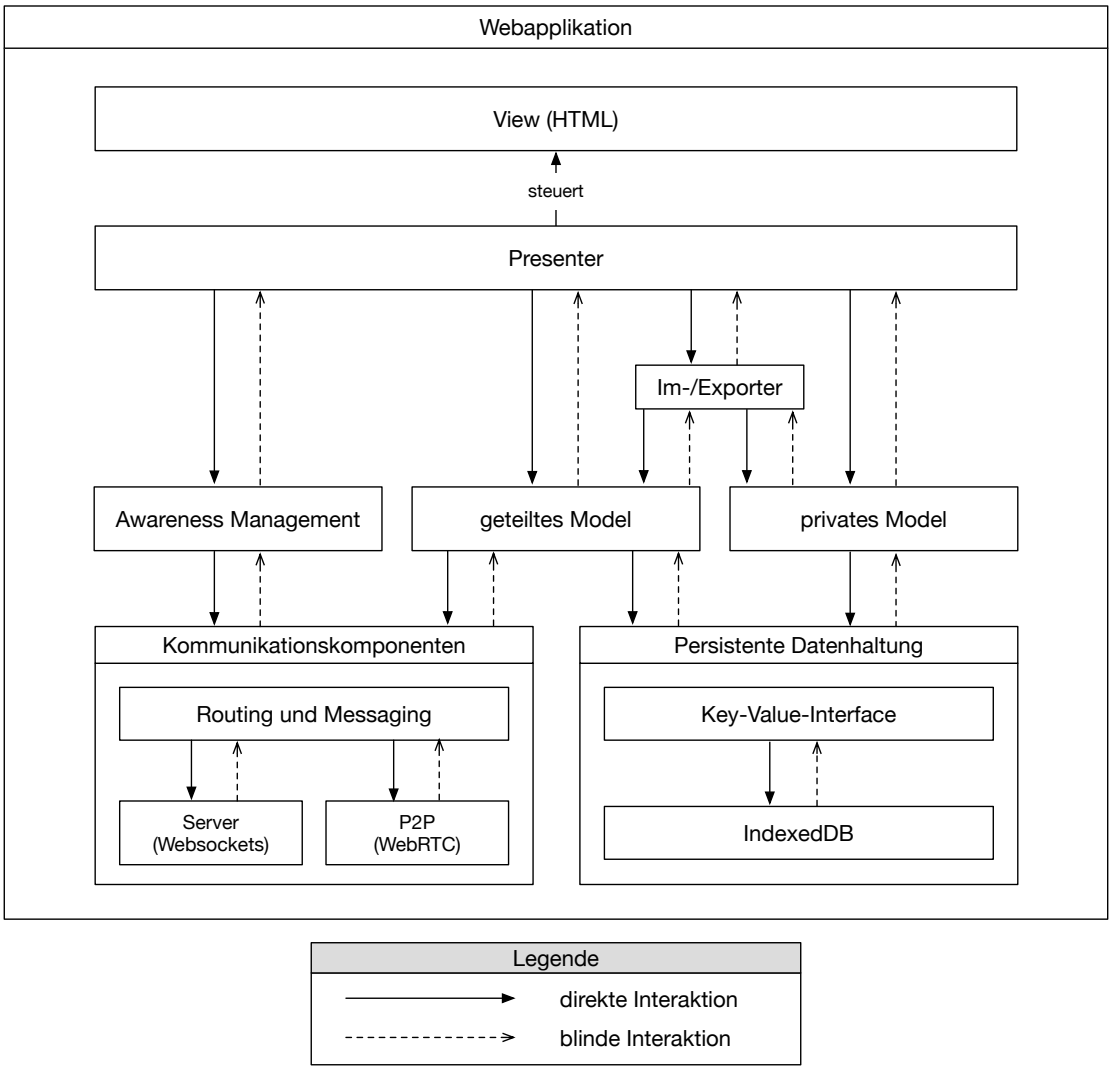

Abbildung 4.10: Architektur der Webapplikation

Für den eigentlichen Datenaustausch greift die Routing-Komponente auf Subkomponenten zu, welche den Versand von Nachrichten über konkrete Protokolle umsetzen. Es existiert eine Komponente für die Kommunikation mit dem zentralen Server per WebSocket und eine weitere, welche die Kommunikation mit anderen Knoten über WebRTC Data Channels realisiert.

Die Kommunikationskomponenten sind ein generischer Teil der Plattform und können für verschiedene Kollaborationsanwendungen verwendet werden, ohne dass Änderungen an ihnen vorgenommen werden müssten.

Persistente Datenhaltung

Die persistente Datenhaltung basiert, wie in Abschnitt 4.6 erläutert, auf der Indexed Database API. Eine weitere Abstraktionsschicht verwaltet die zugrundeliegende IndexedDB und stellt anderen Komponenten ein Key-Value-Interface zur Verfügung, um auf die Daten zuzugreifen (vgl. Abbildung 4.10).

Die persistente Datenhaltung ist ebenfalls ein generischer Teil der Plattform, der keine spezifische Logik für konkrete Kollaborationsanwendungen enthält.

Model

Das Model stellt die eigentliche Datenrepräsentation der Kollaborationsanwendung dar und übernimmt dementsprechend verschiedene Aufgaben. Es interagiert mit der persistenten Datenhaltung, um die Daten, die es verwaltet, persistent speichern und wieder abrufen zu können. Je nach komplexität der Anwendungsdomäne, kann das Model auch Anwendungslogik kapseln, die unabhängig von der Darstellung ist. Zudem ist es zuständig für das Konsistenzmanagement, das heißt die Synchronisation der Daten mit anderen Netzwerkknoten. Dafür greift es auf die Schnittstelle der Kommunikationskomponenten zurück und tauscht auf diesem Weg die nötigen Informationen mit anderen Instanzen der Webanwendung aus. Die konkrete Methodik der Datensynchronisation hängt, wie in Abschnitt 4.7 erläutert, von der jeweiligen Kollaborationsanwendung und deren Anforderungen ab.

Eine Kollaborationsanwendung kann viele verschiedene Models besitzen. Models können optional auch privat sein und nur dem lokalen Nutzer zur Verfügung stehen, um individuelle Arbeit an Artefakten zu ermöglichen (vgl. Abschnitt 4.1). In diesem Fall entfällt die Synchronisation der Daten und der Zugriff auf die Kommunikationskomponenten.

Das Model stellt anderen Komponenten, konkret dem Presenter, eine Schnittstelle zur Verfügung, um die repräsentierten Daten zu manipulieren, je nach Anwendungsfall neue Datenobjekte anzulegen oder zu löschen. Weil Manipulationen der Daten nicht nur lokal ausgelöst werden sondern auch von anderen Systemen kommen können, greift es auf einen Event-Mechanismus zurück, um den Presenter asynchron über Änderungen zu informieren. Es handelt sich hierbei um eine blinde Interaktion. Das Model muss keinerlei Informationen über den Presenter besitzen und kann unabhängig von ihm implementiert werden.

Da das Model je nach Anwendungsfall andere Funktionalität bieten muss, muss es konkret für die jeweilige Kollaborationsanwendung implementiert werden. Allerdings kann

die Plattform generische oder oft benötigte Funktionalität für das Model so bereitstellen, dass konkrete Models darauf aufbauen können (vgl. Abschnitt 4.7). Ein Beispiel für eine solche generische Funktionalität sind Algorithmen zur Synchronisation der Daten auf verschiedenen Netzwerkknoten.

Im-/Exporter

Um die Interoperabilität mit anderen Softwaresystemen zu ermöglichen, existiert die Im-/Exporter-Komponente. Sie verwendet direkt die Schnittstelle des Model, um auf dessen Daten zuzugreifen und sie in ein beliebiges Datenformat zu überführen. Diese Daten werden mittels der in Abschnitt 4.2 angesprochenen HTML5-APIs als Dateien exportiert. Beim Import liest der Im-/Exporter Dateien ein und verwendet entsprechende Funktionen der Model-Schnittstelle, um die gewünschten Informationen zu reproduzieren.

Presenter

Der Presenter ist die wesentliche Komponente für die Interaktion mit dem Nutzer der Anwendung. Er enthält den Großteil der Präsentationslogik, steuert die View, reagiert auf Nutzeraktionen und übersetzt sie in Aktionen auf dem Model. Hierfür greift er auf die vom Model bereitgestellten Schnittstellen zu. Ebenso reagiert er auf Änderungen des Models und übersetzt diese in eine Änderung im User Interface. Da Presenter und View eng miteinander verwoben sind, existiert in der Regel ein Presenter pro View.

Für die Manipulation des User Interface greift der Presenter auf die standardisierten Methoden des Document Object Model (DOM) zurück, die vom Webbrowser implementiert werden. Die DOM-Schnittstelle bietet ihm zudem die Möglichkeit mittels Events auf Nutzeraktionen zu reagieren.

View

Die View realisiert den sichtbaren Teil des User Interface. Sie enthält selbst keine, oder nur sehr wenig, Logik. Der Presenter ist dafür verantwortlich, die View bei Bedarf anzupassen. Eine Kollaborationsanwendung kann mehrere Views beinhalten, beispielsweise um es dem Nutzer zu ermöglichen die gleichen Daten aus verschiedenen Perspektiven zu betrachten.

Umgesetzt wird die View mittels der Webtechnologien HTML und CSS. Beide sind reine Beschreibungssprachen. Um Logik in der View zu vermeiden, wird der JavaScript-Code in den Presenter ausgelagert.

Awareness Management

Um die Erfassung und den Austausch von Awarenessinformationen zu ermöglichen, wird eine eigene Komponente eingeführt, die sich grob an dem von Saucedo-Tejada und Mendoza vorgeschlagenen Eventmechanismus orientiert (vgl. Abschnitt 2.2.4). Sie stellt dem Presenter eine Schnittstelle zur Verfügung, um das Awareness Management über bestimmte Nutzeraktionen zu informieren und selbst per Event über Aktionen anderer Teilnehmer informiert zu werden. Je nachdem wie eng die Erfassung und Darstellung von Awarenessinformationen mit dem jeweiligen User Interface verknüpft ist, kann das Awareness Management über einen eigene Presenter und Views verfügen.

Der Austausch von Awarenessinformationen wird über die Kommunikationskomponenten der Architektur abgewickelt, auf dessen Schnittstellen das Awarenessmanagement zugreift.

5 Entwicklung eines Prototypen

Im vorherigen Kapitel wurde die Entwicklung und Konzeption einer Architektur für eine Face-to-Face-Kollaborationsplattform beschrieben. Diese Architektur soll in einem Prototypen umgesetzt werden, dessen Zweck einerseits eine Demonstration der Umsetzbarkeit der beschriebenen Architektur ist und der andererseits auch als Basis für eine Evaluation dienen kann. Wie die Architektur selbst, teilt sich auch der Prototyp in zwei große Komponenten auf: Den zentralen Server und die Webapplikation. Deren Implementierung und damit zusammenhängende Entscheidungen sollen in den folgenden Abschnitten erläutert werden.

5.1 Anwendungsfall

Die Architektur selbst gibt keine konkrete Form der Kollaboration vor. Zunächst stellt sich daher die Frage, welche Form von Kollaborationsanwendung im Prototypen implementiert wird.

Weil der Prototyp auch zur Evaluation verwendet werden soll, ist es sinnvoll, einen Anwendungsfall zu wählen, der sich an den spezifizierten High-Level-Anforderungen der Architektur orientiert. Gleichzeitig soll der Implementierungsaufwand in einem für die Masterarbeit akzeptablen Rahmen gehalten werden.

Ein Aspekt, welcher sich anhand eines Prototypen erproben lässt, sind die zu erwartende Performance und Geschwindigkeit einer Kollaborationsanwendung auf Basis des Architekturentwurfs. Um diese zu Überprüfen bietet sich ein System nach dem WYSIWIS-Prinzip an. Hierbei teilen alle Nutzer die gleiche Sicht auf die Daten, was außerdem den Implementierungsaufwand reduziert. Änderungen durch andere Teilnehmer sollten instantan im Interface sichtbar werden. Daruch wird eine Überprüfung der zu erwartenden Performance, im Sinne der Latenz, möglich.

Konkret fällt die Wahl auf eine kollaborative Mindmapping-Anwendung. Mindmaps ermöglichen eine grafische Repräsentation, die über eine reine Textdarstellung hinausgeht, was für die Betrachtung der Performance interessant ist. Gleichzeitig ist die Darstellung einer Mindmap aus Implementierungssicht ein relativ einfach zu lösendes Problem, das dem engen Zeitrahmen angemessen erscheint. Eine Mindmapping-Anwendung ermöglicht außerdem eine einfache Umsetzung weiterer Anforderungen, wie der Ermöglichung von individueller Arbeit oder der Erfassung und dem Austausch von Awarenessinformationen.

Zum gegenwärtigen Zeitpunkt unterstützen nur wenige mobile Browser den WebRTC-Standard. Für die Unterstützung von Touchscreens oder die Darstellung auf einem Smartphonebildschirm wären größere Anpassungen am User Interface notwendig. Daher ist die prototypische Anwendung vorerst für den Einsatz auf Desktop-Computern ausgelegt.

5.2 Funktionalität

Der Anwendungsfall, den der Prototyp adressieren soll, wurde im vorherigen Abschnitt festgelegt. Um nun seine konkrete Funktionalität abzustecken, soll an dieser Stelle eine Reihe von funktionalen Anforderungen festgelegt werden. Wie bereits angesprochen, handelt es sich bei dem Prototypen nicht um ein Produkt für den Einsatz in realen Kontexten sondern um eine Machbarkeitsstudie. Dementsprechend wäre der Aufwand einer vollständigen Anforderungserhebung unangemessen. Stattdessen wurden die Anforderungen unter der Prämisse festgelegt, möglichst viele Aspekte der Architektur mit vertretbarem Aufwand umzusetzen. Sie orientieren sich außerdem an den, in Abschnitt 4.1 bereits aufgeführten, High-Level-Anforderungen an die Architektur.

Die Liste der funktionalen Anforderungen findet sich in Tabelle 5.1. Sie umfasst Aspekte wie die Verwaltung einer Sitzung, die gemeinsame Erstellung und Bearbeitung einer Mindmap mit anderen Teilnehmern, die Unterstützung individueller Arbeit, den Im- und Export von Daten und die Erfassung und Bereitstellung von Awarenessinformationen. Grundlegende Anforderungen, die bereits durch die Architektur vorgegeben sind, wurden nicht noch einmal explizit aufgeführt. Hierzu gehört beispielsweise die Lauffähigkeit der Anwendung im Webbrowser.

5.3 Implementierung der Webapplikation

In diesem Abschnitt soll die Implementierung der Webapplikation beschrieben werden, die den Hauptteil des Prototypen ausmacht. Die Entwicklung findet, wie bereits angesprochen, in der Programmiersprache JavaScript statt und stützt sich auf weitere Webtechnologien, die in modernen Browsern verfügbar sind.

Tabelle 5.1: Funktionale Anforderungen an den Prototypen

F1	Das System sollte es dem Nutzer ermöglichen eine Kollaborationssitzung anzulegen
F2	Das System sollte es dem Nutzer ermöglichen, andere Teilnehmer zu der Kollaborationssitzung einzuladen
F3	Das System sollte es dem Nutzer ermöglichen, einer bestehenden Kollaborationssitzung beizutreten
F4	Das System sollte die in einer Kollaborationssitzung anfallenden Daten persistent speichern
F5	Das System sollte die persistent gespeicherten Daten der Kollborationssitzung wieder abrufen können
F6	Das System sollte strukturierte Daten in Form einer Mindmap darstellen können, deren Knoten kurze Begriffe in Textform sind, die untereinander durch Verbindungen in Beziehung gesetzt werden
F7	Das System sollte es dem Nutzer ermöglichen, Begriffe einzutragen, die der Mindmap hinzugefügt werden
F8	Das System sollte es dem Nutzer ermöglichen, Begriffe zu bearbeiten
F9	Das System sollte es dem Nutzer ermöglichen, Begriffe zu löschen
F10	Das System sollte es dem Nutzer ermöglichen, Verbindungen zwischen Begriffen anzulegen
F11	Das System sollte es dem Nutzer ermöglichen, Verbindungen zwischen Begriffen zu löschen
F12	Das System sollte die der Mindmap zugrunde liegenden Daten bei allen Teilnehmern synchronisieren
F13	Das System sollte die grafische Darstellung bei Änderungen der Daten anpassen
F14	Das System sollte es dem Nutzer ermöglichen, privat Begriffe anzulegen, die nicht zwischen allen Teilnehmern synchronisiert werden,
F15	Das System sollte es dem Nutzer ermöglichen, private Begriffe zu bearbeiten
F16	Das System sollte es dem Nutzer ermöglichen, private Begriffe zu löschen
F17	Das System sollte es dem Nutzer ermöglichen, private Begriffe nachträglich der öffentlichen Mindmap hinzuzufügen
F18	Das System sollte es dem Nutzer ermöglichen, die der Mindmap zugrunde liegenden Daten in Form einer Datei auf sein lokales Dateisystem zu exportieren
F19	Das System sollte es dem Nutzer ermöglichen eine exportierte Datei aus seinem lokalen Dateisystem zu importieren und die entsprechenden Daten der Mindmap hinzufügen
F20	Das System sollte es dem Nutzer ermöglichen seinen Namen einzutragen
F21	Das System sollte es dem Nutzer ermöglichen seinen eingetragenen Namen zu ändern
F22	Das System sollte dem Nutzer die Namen aller anderen Teilnehmer anzeigen
F23	Das System sollte dem Nutzer die Darstellung der Teilnehmernamen bei Änderungen aktualisieren
F24	Das System sollte dem Nutzer darstellen, welche anderen Teilnehmer die Anwendung zum aktuellen Zeitpunkt aktiv benutzen

5.3.1 Verwendete Bibliotheken

Für die Umsetzung des Prototypen wurde auf eine Reihe externer Softwarebibliotheken zurückgegriffen, die an dieser Stelle mit ihrem jeweiligen Einsatzzweck aufgeführt werden sollen.

Eventhandling

Ein Großteil der Kommunikation zwischen den einzelnen Komponenten des Systems läuft asynchron. Eine in JavaScript-Anwendungen weit verbreitete Möglichkeit, diese Kommunikation umzusetzen, sind Events, die von einer Komponente ausgelöst werden und auf die andere Komponenten bei Bedarf mittels einer Callback-Funktion reagieren können (vgl. auch Observer-Pattern in Abschnitt 4.3.2). Obwohl viele Schnittstellen im Browser ebenfalls auf Events basieren, stellen diese bisher keinen generischen Eventmechanismus bereit, der von einer Webapplikation genutzt werden könnte, um eigene Events zu realisieren. Daher wird in der prototypischen Umsetzung der Webapplikation auf eine externe Bibliothek namens **EventEmitter** zurückgegriffen. Sie realisiert die Funktionalität eines Event Emitters, die in eigene JavaScript-Module integriert werden kann [Cal14a]. Diese Integration wird mit mittels eines Moduls namens **Heir** [Cal14b] des gleichen Entwicklers realisiert.

Publish-Subscribe

Für die Kommunikation mit dem zentralen Server kommt, wie bereits festgelegt, das WebSocket-Protokoll zum Einsatz. Auf dieser Basis soll eine Publish-Subscribe-Lösung realisiert werden, die es einem Knoten ermöglicht beim initialen Verbindungsaufbau Kontakt zu anderen Knoten aufzunehmen (vgl. Abschnitt 4.4.2). Um diese Kommunikation umzusetzen, kommt das Softwarepacket **Faye** zum Einsatz. Bestandteil von Faye sind sowohl Serverbibliotheken für verschiedene Programmiersprachen als auch eine JavaScript-Clientbibliothek für den Einsatz im Browser [Cog15].

User Interface

Um eine schnelle Umsetzung des User Interface im Browser zu ermöglichen, greift die prototypische Umsetzung auf das **Bootstrap**-Framework von Twitter zurück. Dieses bietet vorgefertigte CSS-Stile für Standardelemente und eigene Komponenten, die für die Gestaltung des User Interface verwendet werden können. Außerdem stellt es eine Reihe

von Icons für die Verwendung in der Anwendung zur Verfügung [Twi15]. Einige Funktionen von Bootstrap benötigen zusätzlich die **JQuery**-Bibliothek [The15].

QR-Codes

Für die Generierung von QR-Codes innerhalb der Webanwendung wird die JavaScript-Bibliothek **QRCode.js** verwendet. Hiermit lassen aus Zeichenketten clientseitig QR-Codes generieren, ohne dass auf einen externen Service zurückgegriffen werden muss, für dessen Nutzung eine Internetverbindung notwendig wäre [San15].

SHA2

Um zufällige IDs zu generieren, die gleichmäßig im ringförmigen Adressraum verteilt sind, kommt eine SHA2-Hashfunktion zum Einsatz. Die Webapplikation verwendet hierfür die Bibliothek **js-sha512** [15].

URI-Parsing

Sessioninformationen werden über einen Query-Parameter innerhalb der URL der Webapplikation weitergegeben. Für die Verarbeitung der URL wird die Bibliothek **URI.js** [Reh13] verwendet.

Testing

Um Unit-Tests für einige Komponenten der Webapplikation zu realisieren, kommen mehrere Bibliotheken zum Einsatz. Das **Mocha**-Framework [Hol11], die Assertion-Bibliothek **should.js** [Hol15] und eine Bibliothek für Steuerung von von asynchron ausgeführten Aufgaben namens **async.js** [McM15].

5.3.2 Klassen

JavaScript ist eine Multiparadigmen-Programmiersprache und unterstützt sowohl prozedurale, objektorientierte als auch funktionale Programmierung. Es unterscheidet es sich von klassischen objektorientierten Sprachen wie Java unter anderem durch das Fehlen eines Klassen-Konzepts. Stattdessen kommt prototypische Vererbung zum Einsatz. Dennoch wurde ein Großteil der Webapplikation in Form von klassenähnlichen Modulen (im

Folgenden der Einfachheit halber dennoch als „Klasse“ bezeichnet) implementiert, die in Abbildung 5.1 als Klassendiagramm dargestellt sind. Die Klassenblöcke des Diagramms enthalten nur die jeweils wichtigsten Methoden.

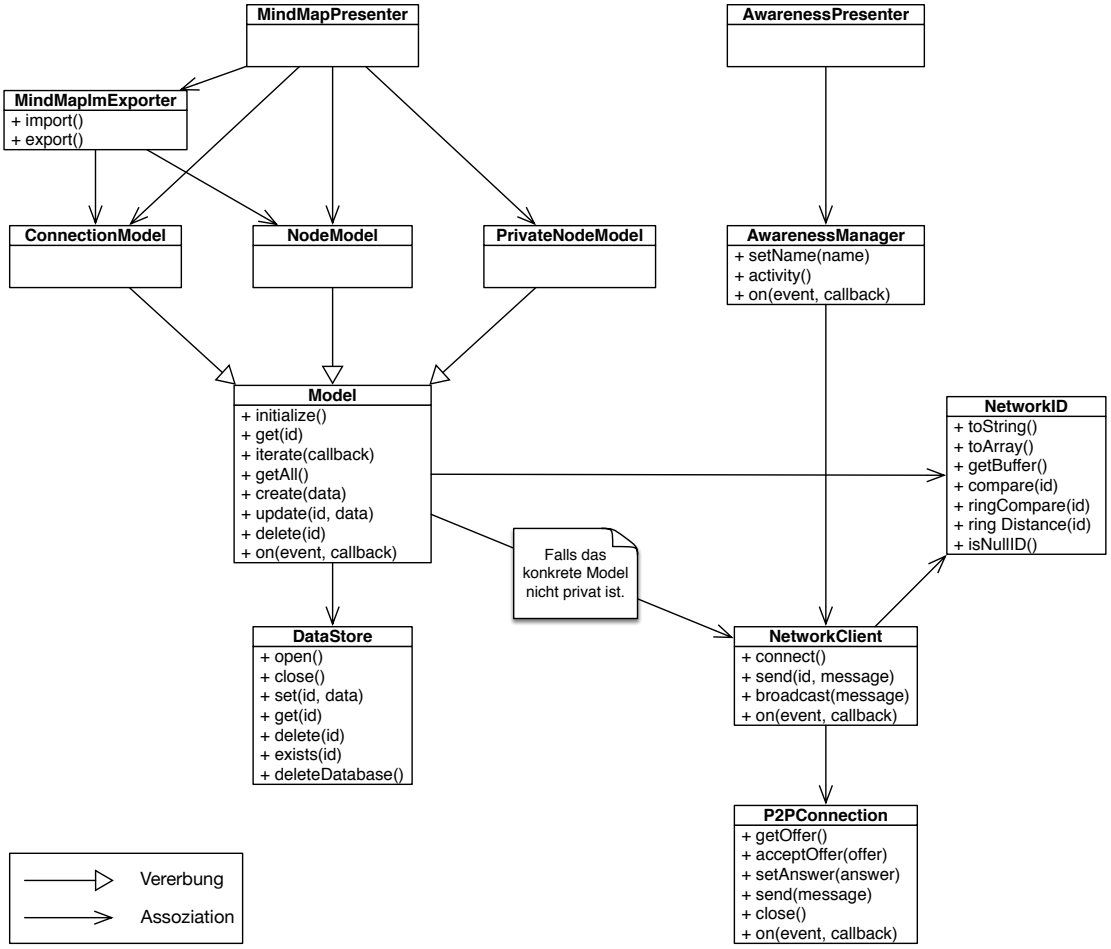

Abbildung 5.1: Klassendiagramm der Webapplikation

Von vielen der aufgeführten Klassen existiert zur Laufzeit der Anwendung nur eine Instanz, da sie eine bestimmte Komponente der Architektur umsetzen die nur einmal benötigt wird. Ausnahmen sind beispielsweise die Klassen **NetworkID** und **P2PConnection**, die mehrfach instanziiert werden. Die einzelnen Klassen werden im Folgenden genauer erläutert.

NetworkID

Knoten im Peer-to-Peer-Netzwerk und Datenobjekte müssen eindeutig identifizierbar sein. Aus diesem Grund wurde die Klasse **NetworkID** entwickelt. Sie realisiert eine eindeutige Kennung in Form einer n-stelligen Binärzahl. Der Wert n ist dabei frei wählbar

und wurde für den Prototypen auf 128 festgelegt. Je länger die Binärzahl ist, desto geringer ist die Wahrscheinlichkeit einer zufälligen Kollision zweier Kennungen.

Da JavaScript Zahlenwerte intern immer als 64-bit Fließkommazahl ablegt, können größere Zahlen nativ nicht dargestellt und auch keine arithmetischen Operatoren auf sie angewendet werden. Um dieses Problem zu umgehen, speichert NetworkID Werte als Binärzahl in einem ArrayBuffer. Um den Aufbau des Peer-to-Peer-Netzwerks und das Routing von Nachrichten, wie in Abschnitt 4.4 beschrieben, zu ermöglichen sind zudem einige Berechnungen notwendig, darunter ein Größenvergleich der Werte und die Bestimmung des Abstands zweier Werte im ringförmigen Adressraum. Diese wurden in Form von Methoden implementiert. Um den Abstand zu bestimmen, ist beispielsweise eine Subtraktion zweier IDs notwendig. Hierfür wird der ArrayBuffer intern in kleinere Abschnitte zerlegt, die von JavaScript noch arithmetisch verarbeitet werden können. Diese werden dann einzeln, unter Berücksichtigung des Übertrags, subtrahiert und zu einem neuen Wert zusammengesetzt.

Die **NetworkID**-Klasse bietet zudem einige Methoden zur Umwandlung in andere Datentypen, wie ein Array von Zahlen oder einen String, der eine Hexadezimale Repräsentation enthält. Letzteres ist für die Darstellung in JSON-Strings von Nutzen.

P2PConnection

Die Klasse **P2PConnection** repräsentiert Direktverbindungen zu anderen Knoten mittels WebRTC Data Channels. Hierfür kapselt sie die WebRTC-API des Browsers und stellt dem Entwickler eine einfache, asynchrone Schnittstelle für den Aufbau von Verbindungen und den Versand von Nachrichten bereit.

Wie sich während der Entwicklung herausstellt, bieten die aktuellen Implementierungen von WebRTC keine zuverlässige Möglichkeit, den Abbruch einer Verbindung festzustellen. Daher implementiert **P2PConnection** eine durchgehende Überprüfung der Konnektivität mittels Heartbeat-Nachrichten [vgl. Li+04], die in regelmäßigen Abständen versandt werden. Geht bis zum Ablauf eines Timeouts keine Antwort ein, ist davon auszugehen, dass die Verbindung getrennt wurde und ein entsprechendes Event wird ausgelöst.

NetworkClient

NetworkClient ist zuständig für den Aufbau und das Management des Peer-to-Peer-Netzwerks und das Routing und den Versand von Nachrichten. Eine Instanz von **NetworkClient** repräsentiert somit einen Knoten im Netzwerk. Sie enthält die dafür notwen-

dig Funktionalität und hält die notwendigen Datenstrukturen vor, darunter die eigene ID, das Leaf Set oder eine Liste bereits erhaltener Broadcast-Nachrichten. Für den Austausch von Daten mit anderen Knoten, greift es auf die Klasse **P2PConnection** zurück. Zudem verwendet es den Faye-Client, um eine Verbindung zum zentralen Server herzustellen.

Die **NetworkClient**-Klasse abstrahiert die mit dem Peer-to-Peer-Netzwerk zusammenhängende Logik und stellt anderen Modulen eine einfache Schnittstelle zur Verfügung, die im wesentlichen nur den Aufbau einer Verbindung, den Versand von Nachrichten an einzelne andere Knoten und den Versand von Broadcast-Nachrichten ermöglicht. Über Ereignisse, wie den Eingang einer neuen Nachricht, werden andere Module mittels des bereits erwähnten Event-Mechanismus informiert.

DataStore

Datastore realisiert die persistente Datenhaltung der Webapplikation und entspricht dem in Abschnitt 4.10.2 angesprochenen Key-Value-Interface. Während es intern auf eine IndexedDB zurückgreift, stellt es nach außen eine Schnittstelle zur Verfügung, die einem einfachen Key-Value-Store gleicht. Hierzu gehören Methoden zum speichern, abrufen, löschen von Werten unter einem bestimmten Schlüssel. Zudem existiert die Möglichkeit, die IndexedDB vollständig zu löschen. Diese Funktion kam während der Entwicklung zu Debugging-Zwecken zum Einsatz.

Die Schnittstelle von **DataStore** ist asynchron. Methodenaufrufe liefern Ergebnisse nicht direkt als Rückgabewert. Stattdessen kann die aufrufende Instanz mittels eines Callbacks reagieren, sobald die Ergebnisse vorliegen.

Model

Die Klassen des Model repräsentieren die eigentliche Mindmap innerhalb der Anwendung. Es existieren verschiedene Modelklassen: Die Klasse **NodeModel** repräsentiert Knoten, d.h. Begriffe innerhalb der Mindmap, die Klasse **ConnectionModel** repräsentiert Verbindungen zwischen den Knoten und die Klasse **PrivateNodeModel** repräsentiert Knoten, die der Nutzer in einem privaten Bereich lokal anlegen und bearbeiten kann. Da allen Models ähnliche Funktionalitäten zugrunde liegen, existiert eine generische Klasse namens **Model**, deren Funktionalität von den oben genannten konkreten Modelklassen geerbt wird.

Die Modelklassen stellen eine Schnittstelle zur Verfügung, um auf Objekte, das heißt in diesem Fall Knoten oder Verbindungen in der Mindmap, zuzugreifen. Diese Objekte

werden durch eine eindeutige Kennung identifiziert, die bei der Erstellung des Objekts zufällig generiert wird. Hierfür kommt wieder die Klasse **NetworkID** zum Einsatz.

Objekte können abgerufen, angelegt, verändert und gelöscht werden. Das Model speichert diese Änderungen in der persistenten Datenhaltung und benachrichtigt andere Instanzen der Webapplikation darüber, indem es entsprechende Nachrichten über das Peer-to-Peer-Netzwerk versendet. Bei privaten Modelklassen, wie **PrivateNodeModel**, ist die Netzwerkkommunikation deaktiviert. Ein zusätzlicher Mechanismus, um die Daten zwischen den Teilnehmern Konsistenz zu halten, ist nicht Teil der prototypischen Implementierung. Im realen Anwendungsfall wäre dies allerdings angebracht, da nicht davon ausgegangen werden kann, dass alle Teilnehmer über die gesamte Sitzung hinweg anwesend und mit dem Peer-to-Peer-Netzwerk verbunden sind. Mögliche Techniken hierfür wurden in Abschnitt 4.7 bereits angesprochen.

Tritt eine Änderung der Daten ein, informiert das Model andere Module mittels eines Events darüber. Dabei macht es keinen Unterschied, ob die Änderung lokal oder von einem anderen Teilnehmer angestoßen wurde. Ein Vorteil dieser Vorgehensweise ist, dass der entsprechende Code für die Reaktion auf das Event und die Anpassung des User Interface nur einmal geschrieben werden muss.

MindMapImExporter

Die Klasse **MindMapImExporter** realisiert den Im- und Export von Daten aus dem Model. Ihr Schnittstelle besteht aus lediglich zwei Methoden, **import()** und **export()**. Intern greift es auf die Methoden des Model zurück.

Als Datenformat für den Im- und Export kommen UTF-8 kodierte JSON-Textdateien zu Einsatz. Das JSON Format wird von allen gängigen Browsern unterstützt. Eine Umwandlung von JavaScript-Objekten in JSON-Zeichenketten und umgekehrt ist somit sehr einfach möglich. Um die Dateien zu laden und zu speichern, werden die HTML5 File API und das enquotedownload-Attribut der Anker-Elements verwendet (vgl. Abschnitt 4.2).

MindMapPresenter

Der **MindMapPresenter** enthält die mit der Mindmap zusammenhängende Präsentationslogik. Er ist zuständig für die Darstellung von Inhalten und die Interaktion mit dem Nutzer. Hierfür steuert er die Darstellung im Browser die Schnittstelle des DOM und interagiert mit dem Model, um Aktionen des Nutzer in Änderungen der Daten zu überführen. Außerdem kontrolliert er den **MindMapImExporter**.

AwarenessManager

Der **AwarenessManager** ist zuständig für die Verwaltung von Awarenessinformationen und deren Austausch über das Peer-to-Peer-Netzwerk. In der prototypischen Implementierung beschränken sich die Awarenessinformationen auf den Namen eines Teilnehmers sowie die Feststellung einer generellen Aktivität im User Interface anhand von Klicks und Mausbewegungen, die anderen Nutzern dabei helfen soll, abzuschätzen welche Teilnehmer gerade aktiv mit dem System arbeiten. Da diese Informationen direkt im User Interface generiert werden müssen, ist der **AwarenessManager** nicht selbst zuständig für ihre Erhebung. Stattdessen existiert ein separater Presenter, der im nachfolgenden Abschnitt erläutert wird.

AwarenessPresenter

Der **AwarenessPresenter** enthält den Teil der Anwendungslogik, der mit der Erhebung und Darstellung von Awarenessinformationen zusammenhängt. Der **AwarenessPresenter** ruft entsprechende Methoden des **AwarenessManager** auf, wenn bestimmte Ereignisse im User Interface eintreten. Andererseits reagiert er auf Events, die vom **AwarenessManager** gefeuert werden, und passt das User Interface entsprechend an.

5.3.3 User Interface

Das User Interface der prototypischen Webapplikation ist in Abbildung 5.2 dargestellt. Es gliedert sich im wesentlichen in vier Bereiche: Eine Titelzeile, zwei Seitenleisten sowie den eigentlichen Arbeitsbereich.

Die Titelzeile enthält für Debugging-Zwecke einen Button, um die Datenbank der persistenten Datenhaltung zu löschen.

Die linke Seitenleiste dient der Darstellung von Awarenessinformationen. In diesem Fall ist das eine Liste der anderen Teilnehmer. Ist ein Teilnehmer aktiv, so wird neben seinem Namen ein entsprechendes Symbol angezeigt. Zusätzlich kann der Nutzer seinen eigenen Namen eintragen und über einen Button neue Teilnehmer zur Kollaborationssitzung einladen. Dieser öffnet einen Modalen Dialog, welcher neben einem erklärenden Text eine URL enthält, die an andere weitergegeben werden kann. Die URL ist zusätzlich in einem QR-Code kodiert. Der Dialog ist in Abbildung 5.3 abgebildet.

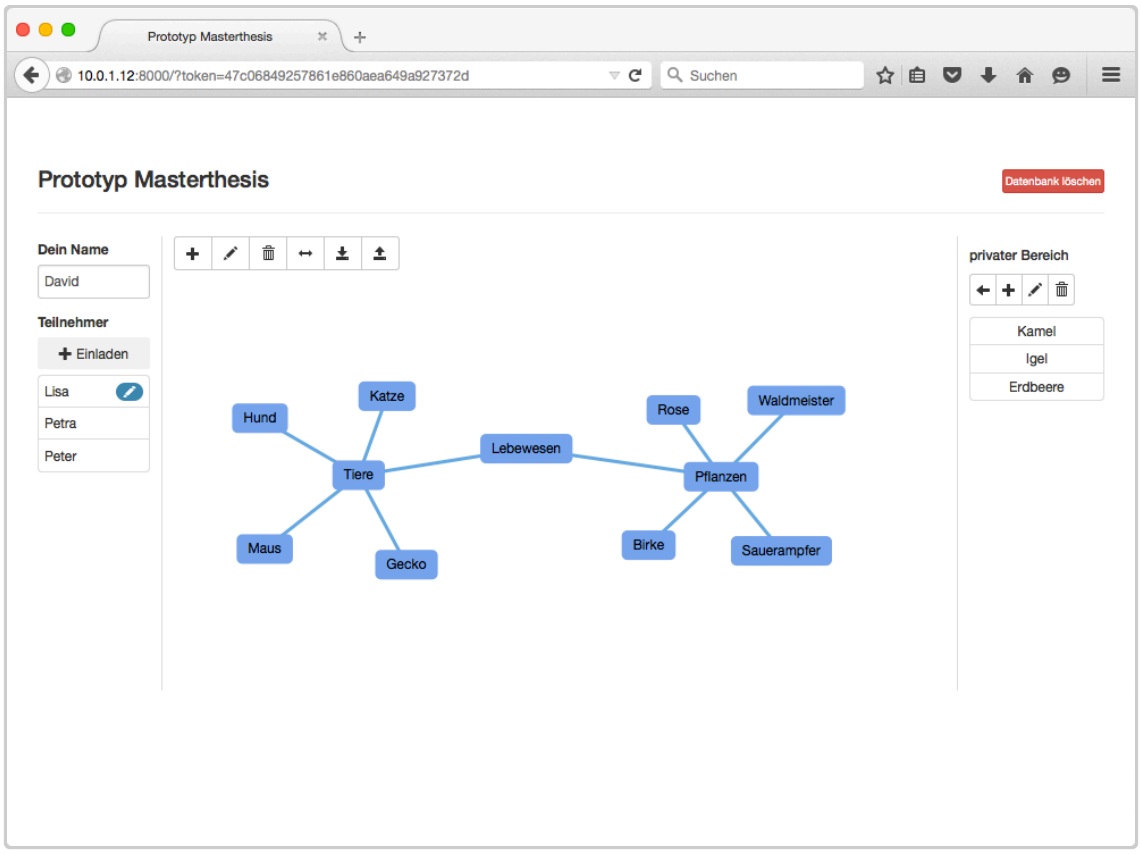

Abbildung 5.2: Die Webapplikation

Der Arbeitsbereich nimmt den größten Teil des Bildschirms ein und enthält die Darstellung der Mindmap. Zusätzlich enthält er eine Werkzeugleiste mit Buttons, über die Begriffe und Verbindungen angelegt, bearbeitet und gelöscht werden und die gesamte Mindmap exportiert bzw. importiert werden kann. Knoten können unter Verwendung der Maus frei verschoben werden. Da sich die Anwendung an dem WYSIWIS-Paradigma orientiert, sind Änderungen instantan für alle anderen Teilnehmer sichtbar.

Die Darstellung der Mindmap erfolgt mittels Scalable Vector Graphics (SVG), einem Standard für Vektorgrafiken, der von vielen modernen Browsern unterstützt wird. SVG basiert auf XML und kann direkt in den HTML-Quelltext eingebunden und über die Schnittstellen des DOM manipuliert werden. Durch die Verwendung von Vektorgrafiken ist die Darstellung zudem ohne Qualitätsverlust skalierbar.

Die rechte Seitenleiste enthält einen privaten Bereich, in dem der Nutzer eigene Ideen für neue Begriffe sammeln kann. Die Begriffe werden in einer Liste dargestellt. Der private Bereich verfügt über eine eigene Werkzeugleiste, die dem im Arbeitsbereich ähnelt. Zusätzlich enthält sie noch einen Button, um den aktuell ausgewählten Begriff aus dem privaten Bereich in die öffentliche Mindmap zu bewegen.

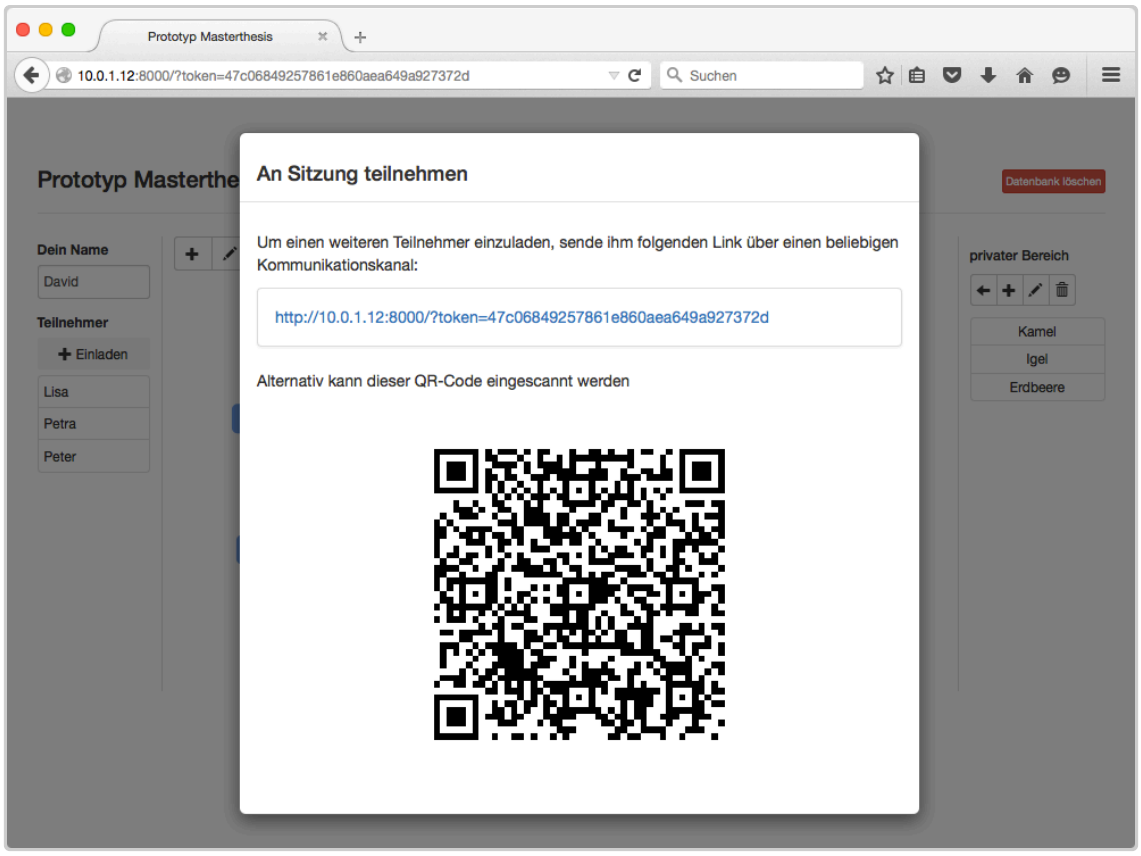

Abbildung 5.3: Einladen neuer Teilnehmer

5.4 Implementierung des zentralen Servers

In diesem Abschnitt soll die Implementierung des zentralen Servers beschrieben werden.

5.4.1 Technische Basis

Da die Zentrale Komponente leicht austauschbar sein soll, wurde in Kapitel 4 lediglich die grundlegende Funktionalität des Servers spezifiziert, nicht jedoch seine konkrete technische Basis. Für die prototypische Implementierung wird diese Entscheidung daher hier nachgeholt.

Die Entwicklung der Webapplikation findet in JavaScript statt. Zudem wird für den Nachrichtenaustausch das JSON-Format verwendet, welches sich mit JavaScript sehr einfach verarbeiten lässt. Im realen Anwendungsfall sollte der Server keinen Zugriff auf die Inhalte der über ihn ausgetauschten Nachrichten nehmen. Für Debugging-Zwecke ist ein solcher Zugriff allerdings sinnvoll. Daher fällt die Wahl der Programmiersprache für die

Implementierung des zentralen Servers ebenfalls auf JavaScript. Als Plattform kommt die Laufzeitumgebung zum Einsatz [Nod15].

5.4.2 Webserver

Um die Webapplikation an den Browser des Nutzers ausliefern zu können, muss der zentrale Server einen HTTP-Server betreiben. Node.js bietet liefert zwar ein entsprechendes Modul mit, das allerdings nur sehr grundlegende Funktionalitäten bereitstellt. Um die Auslieferung statischer Dateien zu vereinfachen, kommt das externe Modul zum Einsatz, welches auf dem mitgelieferten HTTP-Server basiert [Str15].

5.4.3 Publish-Subscribe

Auf Seiten der Webapplikation wurde bereits **Faye** als Publish-Subscribe-Lösung ausgewählt. Faye bietet eine entsprechende Server-Komponente für Node.js an, die hier auch verwendet wird. Da **Faye** auf WebSockets basiert, kann es unter einem bestimmten Pfad in den mittels Express umgesetzen Webserver integriert werden.

5.4.4 User Interface

Ein grafisches User Interface ist für den Betrieb des Servers nicht unbedingt notwendig. Allerdings ist ein Anspruch an die Architektur die einfache Initialisierung einer Kollaborationssitzung. Hierzu gehört auch der Betrieb eines Servers auf dem Endgerät eines der Teilnehmer. Deshalb wurde die Entscheidung getroffen, den Server um ein grafisches User Interface zu erweitern. Dieses wird mit Hilfe von **NW.js** realisiert. Dabei handelt es sich um eine Laufzeitumgebung auf Basis des Chromium-Browsers von Google in Kombination mit Node.js. Diese ermöglicht es, sowohl die Methoden des DOM als auch alle Node.js-Module innerhalb des selben JavaScript-Codes zu verwenden. Mittels **NW.js** können somit native Applikationen auf Basis von Webtechnologien entwickelt werden [Wan15].

Das finale User Interface des zentralen Servers ist in Abbildung 5.4 zu sehen. Es bietet dem Nutzer die Möglichkeit, den Server zu starten und zu stoppen, eine Historie der über Publish-Subscribe versandten Nachrichten einzusehen und den Inhalt einzelne Nachrichten im Detail zu betrachten (vgl. Abbildung 5.5). Zudem unterstützt die Serveranwendung den Nutzer dabei, die Webapplikation im Browser zu öffnen. Hierfür generiert die Software eine URL aus der lokalen IP-Adresse des Systems und kopiert diese in die Zwi-

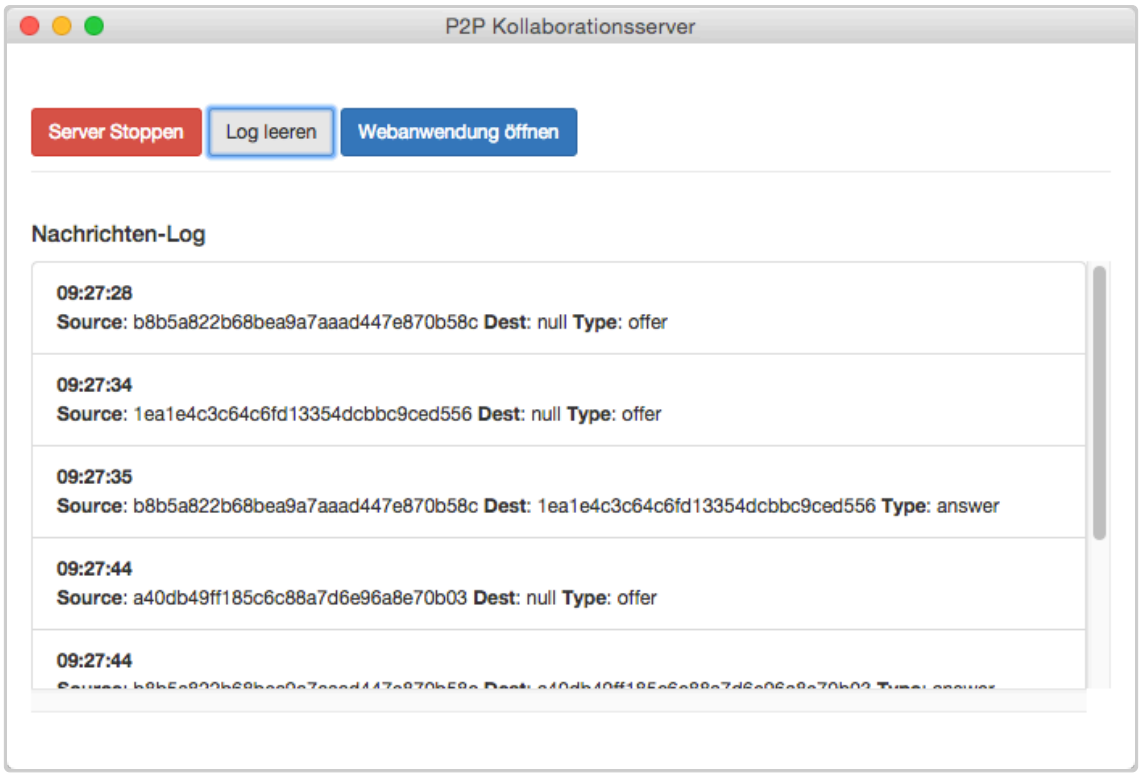

Abbildung 5.4: Die Serversoftware

schenablage. Zudem wird dem Nutzer eine kurze Anleitung gezeigt, die in Abbildung 5.6 dargestellt ist.

5.5 Abweichungen von der Architektur

In der prototypischen Implementierung konnten nicht alle Aspekte der in Kapitel 4 entworfenen Architektur umgesetzt werden. Diese Änderungen sollen im Folgenden erläutert werden.

5.5.1 Nachrichtenformat

Im Architekturentwurf ist ein binäres Nachrichtenformat für den Austausch innerhalb des Peer-to-Peer-Netzwerks vorgesehen (vgl. Abschnitt 4.4.5). Tatsächlich wird es im Prototypen nicht eingesetzt. Es wurde ein entsprechendes JavaScript-Modul entwickelt und getestet, welches Nachrichten im vorgesehenen Format generieren und verarbeiten kann. Es wurde allerdings aus zeitlichen Gründen nicht mehr in die restliche Implementierung integriert. Stattdessen setzt der Prototyp auf Nachrichten in Form von JSON-Strings. Da

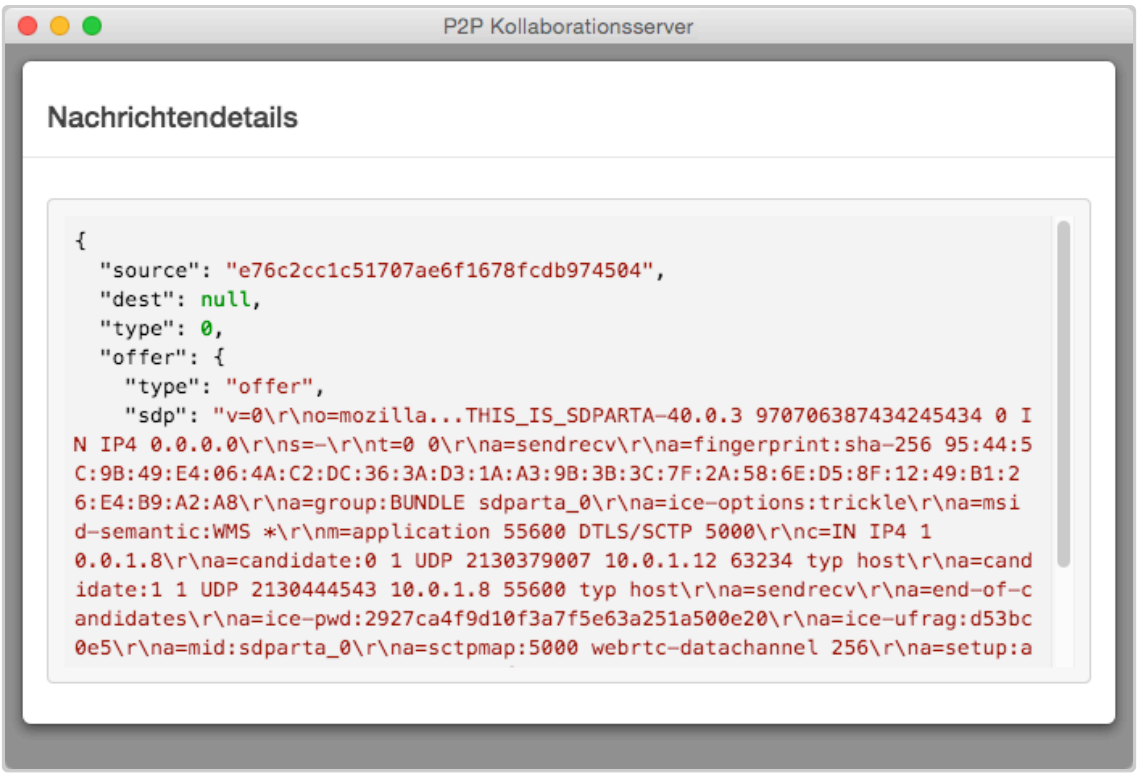

Abbildung 5.5: Detailansicht einer Nachricht

die Verwendung des Binärformats vor allem aus Performancevorteilen heraus begründet ist, stellt diese Änderung für die Demonstration der Funktionalität keine Einschränkung dar.

5.5.2 Konsistenzmanagement

Wie bereits oben in Abschnitt 5.3.2 angesprochen, wurde kein vollständiges Konsistenzmanagement für die Daten der Kollaborationsanwendung umgesetzt. Die Instanzen der Webapplikation synchronisieren Änderungen untereinander, indem sie, bei deren Auftreten, Nachrichten über das Peer-to-Peer-Netzwerk austauschen. Verliert ein Teilnehmer allerdings die Verbindung zum Netzwerk oder kommt erst später zu einer Sitzung hinzu, ist der Bearbeitungsstand inkonsistent. Während die vorhandene Funktionalität für eine Demonstration ausreicht, ist ein Konsistenzmanagement für eine sinnvolle Nutzung im realen Kontext notwendig.

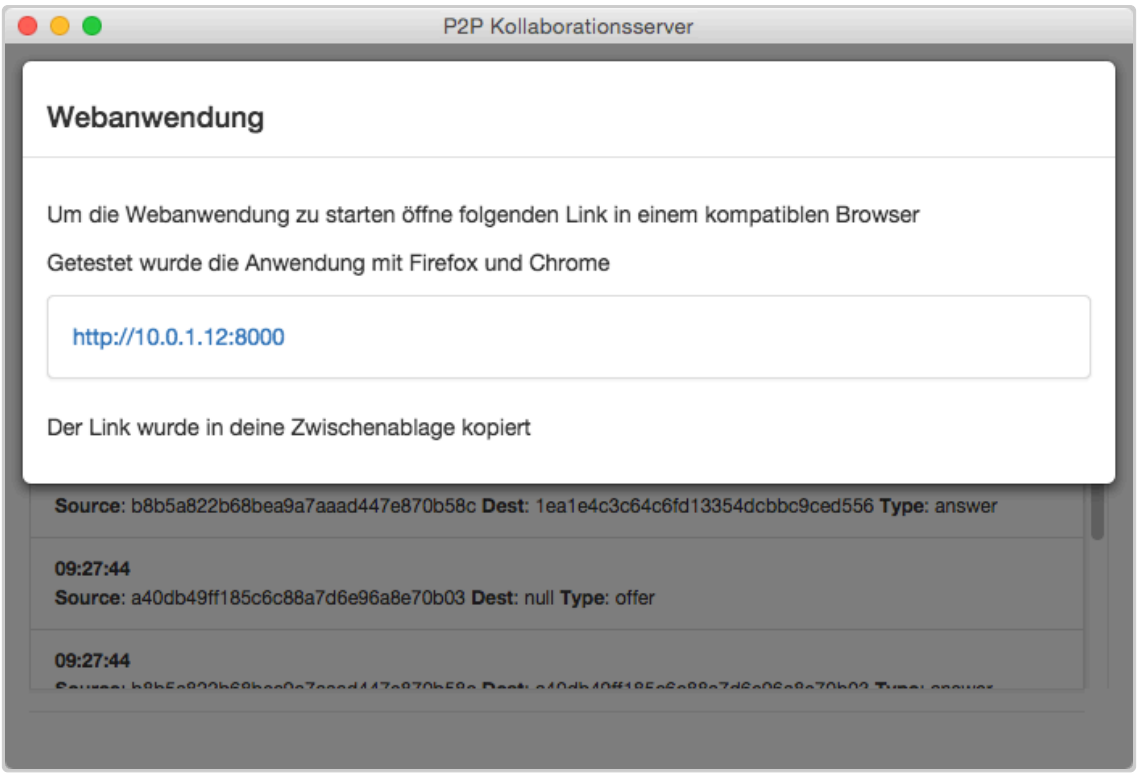

Abbildung 5.6: Anleitung zum Öffnen der Webapplikation

5.6 Herausforderungen bei der Implementierung

Während der Entwicklung des Prototypen ergaben sich erwartungsgemäß Herausforderungen, die in diesem Abschnitt kurz geschildert werden sollen.

5.6.1 Experimentelle Webtechnologien

Eine wesentliche Herausforderung während der Entwicklung stellte die experimentelle Natur einiger Webtechnologien und ihrer konkreten Implementierungen in einigen Webbrowsern dar. Dies betrifft vor allem WebRTC und IndexedDB und äußerte sich in Schwierigkeiten beim Debugging aufgrund irreführender Fehlermeldungen, unterschiedlichem Verhalten des selben Codes in verschiedenen Browsern und nicht vollständig implementierten Schnittstellen.

Ein Beispiel für eine unvollständige Implementierung ist die Möglichkeit, den Abbruch einer WebRTC-Verbindung festzustellen. Zwar definiert die API ein entsprechendes Event, auf das die Anwendung reagieren kann, dieses wurde allerdings in den getesteten Browserversionen nicht oder nur unter bestimmten Bedingungen ausgelöst, beispielsweise nur bei einem ordnungsgemäßen Schließen der Webbrowsers, nicht aber bei einer Trennung

der zugrunde liegenden Netzwerkverbindung. Aus diesem Grund musste, wie in Abschnitt 5.3.2 bereits erläutert, ein eigener Mechanismus zur Erkennung von Verbindungsabbrüchen implementiert werden.

5.6.2 Stabilität des Peer-to-Peer-Netzwerks

Eine weitere Herausforderung betrifft die Stabilität des Peer-to-Peer-Netzwerks. Hier wurde die Komplexität der Implementierung unterschätzt. Zwar funktionierten die in Abschnitt 4.4 angesprochenen Mechanismen wie geplant, aber es zeigte sich, dass unter bestimmten Bedingungen unvorhergesehene Zwischenzustände auftreten. Beispielsweise ist es möglich, dass ein Knoten zwar bereits eine initiale Verbindung zu einem anderen Knoten des Netzwerks hergestellt, aber dessen Leaf Set noch nicht erhalten hat. Somit ist sein Leaf Set kurzzeitig bis auf einen Knoten leer. Versucht während dieses Zeitfensters ein weiterer Knoten, eine Verbindung herzustellen, kann der erste Knoten nicht zuverlässig feststellen, ob er dafür zuständig ist, auf die Nachricht des neuen Knotens zu reagieren. Reagiert er fälschlicherweise auf die Nachricht und erreicht seine Antwort den neuen Knoten zuerst, so erhält dieser neue Knoten ebenfalls kein vollständiges Leaf Set, das Netzwerk wird instabil.

Während der Implementierung konnte nicht genug Zeit aufgewendet werden, um alle derartigen Probleme zu lösen. Unter realen Bedingungen kann es zu Instabilitäten kommen, aufgrund von Verbindungsabbrüchen oder mehreren Knoten, die in kurzer Folge einen Verbindungsversuch unternehmen kommen.

6 Evaluation

In den vorangegangenen Kapiteln wurde eine Architektur für eine Ad-Hoc-Kollaborationsplattform auf Basis von Browsertechnologien entworfen und beispielhaft in einem technischen Prototypen umgesetzt. In hier vorliegenden Kapitel sollen nun diese Ergebnisse einer Evaluation unterzogen werden. Hierfür wird einerseits der Prototyp in einer Fallstudie evaluiert und andererseits der Architekturentwurf an sich einer kritischen Reflexion unterzogen.

6.1 Evaluation des Prototypen

Während der Entwicklung des in Kapitel 5 beschriebene Prototypen, wurden einzelne technische Aspekte der Implementierung immer wieder getestet. Die generelle Funktion der Kommunikationskomponenten konnte durch automatisierte Tests überprüft werden. Die gesamte Anwendung wurde zudem manuell auf verteilten Computern mit verschiedenen Betriebssystemen probeweise ausgeführt, teilweise unter Zuhilfenahme von virtuellen Maschinen. Allerdings sollte der Prototyp, für die finale Evaluation, in einem realistischen Kontext ausgeführt werden. Diese Evaluation wird im Folgenden näher erläutert. Zuvor soll allerdings noch darauf hingewiesen werden, dass sich der Prototyp in einem frühen Stadium befindet und noch nicht alle Aspekte der geplanten Architektur umsetzt (vgl. auch Abschnitt 5.5). Er eignet sich somit für eine Bewertung der Machbarkeit des Konzepts, nicht aber für eine vollständige Evaluation der Architektur.

6.1.1 Methodik und Durchführung

Für die Evaluation des Prototypen wurde die Durchführung einer Fallstudie ausgewählt. Diese bietet die Möglichkeit, das System unter realistischen technischen und Nutzungsbedingungen zu erproben. Sechs Personen wurden gebeten, die Kollaborationsanwendung auf eigenen Endgeräten auszuführen und ein gemeinsames Artefakt zu erarbeiten. Hierfür erhielten sie die Aufgabe, eine Mindmap zum Thema *Gesundheit* zu erstellen. In einem ersten Schritt sollten Begriffe, ähnlich wie bei der Brainstorming-Technik, lediglich gesammelt werden. Im zweiten Schritt sollten diese Begriffe dann nach Kategorien geordnet und zu einer Mindmap verbunden werden.

Da es sich vordergründig um eine technische Evaluation handelt, wurden neben der reinen Beobachtung weitere Daten gesammelt. Die Webapplikation wurde um die Ausgabe von Debugging-Nachrichten erweitert, welche die internen Abläufe bei der Verwaltung des Peer-to-Peer-Netzwerks nachvollziehbar machen. Diese Nachrichten können über die

bestehende Websocket-Verbindung an die Serversoftware gesandt werden, um sie in einer zentralen Logdatei zu sammeln. Auf einem der Endgeräte wurde zudem der Bildschirminhalt während der Nutzung der Webapplikation aufgezeichnet.

Während der Entwicklung des Prototypen kamen in der Regel kabelgebundene Netzwerkverbindungen oder das Loopback-Interface zum Einsatz. Bei der Durchführung der Fallstudie, wurde der Versuch unternommen, eine realistische Netzwerksituation nachzubilden. Um eine typische Netzwerkanbindung zu simulieren wurden alle Endgeräte mit einem WLAN-Netzwerk verbunden, welches von zwei Access-Points zur Verfügung gestellt wurde, die im 2,4 GHz-Spektrum arbeiten (IEEE 802.11n). Beide Access-Points waren jeweils über eine Fast-Ethernet-Schnittstelle (100BASE-TX 100 MBit/s Vollduplex) und einen oder mehrere Netzwerkswitches mit einem handelsüblichen Heimrouter verbunden, der ihnen per DHCP eine IPv4-Adresse zur Verfügung stellte.

Nach einer kurzen Erläuterung verwendeten vorerst alle Probanden ein selbst mitgebrachtes Notebook. Drei davon waren mit einem Windows-Betriebssystem ausgestattet, zwei mit Apple OS X und eines mit einer Linux-Distribution. Zum Einsatz kamen die Webbrowser Mozilla Firefox und Google Chrome. Allerdings ergaben sich während der Nutzung Probleme in Form von hohen Latenzen und Verbindungsabbrüchen, die vor allem bei größerer Nutzeraktivität auftraten. Damit dennoch das geplante Nutzungsszenario durchgeführt werden konnte, mussten einige Änderungen am Code vorgenommen werden. Um die Nachrichtenlast im Peer-to-Peer-Netzwerk zu verringern, wurde die Frequenz verringert, mit der Positionsupdates beim Verschieben von Knoten im User Interface übertragen werden. Zudem wurden einige Timeout-Zeiten erhöht, beispielsweise für die Feststellung eines Verbindungsabbruchs. Um eine kontrolliertere Umgebung zu schaffen, wurden alle Teilnehmer gebeten Firefox als Webbrowser zu verwenden. Und da während der Entwicklung keine Tests auf Linux durchgeführten wurden, wurde das Notebook mit Linux-Betriebssystem gegen eines mit OS X getauscht.

Nachdem die beschriebenen Änderungen vorgenommen waren, wurde ein weiterer Anlauf gestartet, in dem das System stabiler funktionierte und die Kollaboration ohne größere Schwierigkeiten ablief. Die Probanden erstellten unter Verwendung des Prototypen die in der Aufgabenstellung geforderte Mindmap. Diese wurde auf einem der Systeme als Datei exportiert. Die während der Durchführung entstandenen Logdateien wurden gesammelt und analysiert.

6.1.2 Analyse der Ergebnisse

Wie sich bei der Analyse der Logdateien zeigte, meldete die Anwendung zu Beginn der Evaluation, vor Anpassung der Software, häufige Verbindungsabbrüche. Diese waren während Testläufen in kabelgebundenen Netzwerken nicht aufgetreten. Der wahrscheinlichste Grund dafür liegt in einer zu hohen Nachrichtenlast, durch häufige Updates beim verschieben von Knoten der Mindmap. Hierfür spricht auch die zu Beginn hohe Latenz, welche sich durch die verspätete Darstellung bestimmter Änderungen im User Interface äußerte. Da die Feststellung eines Verbindungsabbruchs auf dem Versenden und Empfangen von Heartbeat-Nachrichten basiert (vgl. Abschnitt 5.3.2), kann eine zu hohe Latenz dazu führen, dass fälschlicherweise eine getrennte Verbindung angenommen wird. Hierdurch wird das Peer-to-Peer-Netzwerk instabil.

Nachdem die oben beschriebenen Änderungen am Code vorgenommen wurden, traten keine ungewöhnlichen Verbindungsabbrüche mehr auf. Es wurde eine neue Logdatei angelegt, welche die erfolgreiche Kollaborationssitzung protokollierte. Um die Analyse zu vereinfachen wurden die einzelnen Nachrichten je nach Ausgangsknoten eingefärbt und die 128-Bit-Identifier der Knoten durch Kürzel ersetzt. Ein Ausschnitt der Logdatei, der einen den Beginn des Verbindungsaufbaus abdeckt findet sich in Abbildung 6.1. Die gesamte Logdatei findet sich im Anhang. Wie sich zeigte, liefen der Verbindungsaufbau und das Management des Peer-to-Peer-Netzwerks wie geplant ab.

```
19:24:03:168: server: ------------------------ Server gestartet ------------------------
19:24:13:265: node1 (1441992253168): my id is node1
19:24:13:269: node1 (1441992253260): generated initial offer
19:24:13:273: server: Initial offer from node1 to null
19:24:13:291: node1 (1441992253165): my user-agent is: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.10; rv:40.0)
Gecko/20100101 Firefox/40.0
19:24:13:299: node1 (1441992253293): published inital offer
19:24:14:299: node1 (1441992254295): no answer - seems i'm the first node
19:24:33:697: node2 (1441992273230): my id is node2
19:24:33:700: node2 (1441992273535): generated initial offer
19:24:33:701: node2 (1441992273226): my user-agent is: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:40.0) Gecko/20100101
Firefox/40.0
19:24:33:705: server: Initial offer from node2 to null
19:24:33:710: node2 (1441992273784): published inital offer
19:24:33:831: node1 (1441992273711): received initial offer from node2
19:24:33:832: node1 (1441992273712): i'm the nearest node for node2
19:24:33:834: node1 (1441992273831): generated inital answer for node2
19:24:33:835: server: Initial answer from node1 to node2
19:24:33:837: node1 (1441992273834): published inital answer for node2
19:24:33:884: node2 (1441992273919): received initial answer from node1
19:24:33:912: node1 (1441992273906): established the initial connection for node2
19:24:33:913: node1 (1441992273906): adding node2 to my Leaf Set
19:24:33:914: node1 (1441992273908): my new Leaf Set: ( left: null, null, null | right: node2, null, null )
19:24:33:919: node2 (1441992273981): established initial connection to node1
19:24:33:921: node2 (1441992273982): adding node1 to my Leaf Set
19:24:33:922: node2 (1441992273987): my new Leaf Set: ( left: node1, null, null | right: null, null, null )
19:25:13:619: node3 (1441992317842): my id is node3
19:25:13:621: node3 (1441992318480): generated initial offer
19:25:13:622: node3 (1441992317836): my user-agent is: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:40.0) Gecko/20100101
Firefox/40.0
19:25:13:637: server: Initial offer from node3 to null
19:25:13:643: node1 (1441992313637): received initial offer from node3
19:25:13:644: node1 (1441992313640): i'm not the nearest node for node3
19:25:13:674: node3 (1441992319137): published inital offer
19:25:13:824: node2 (1441992313717): received initial offer from node3
19:25:13:825: node2 (1441992313724): i'm the nearest node for node3
19:25:13:830: node2 (1441992313899): generated inital answer for node3
19:25:13:831: server: Initial answer from node2 to node3
19:25:13:838: node2 (1441992313911): published inital answer for node3
```

Abbildung 6.1: Ausschnitt aus der bei der Evaluation entstandenen Logdatei

Die Nachrichten für die Synchronisation der Inhalte wurden ebenfalls wie erwartet ausgetauscht. Allerdings fanden sich in der Logdatei doppelte Einträge für bestimmte Nachrichten. Dieses Phänomen konnte in einem späteren Versuch ebenfalls nachvollzogen werden. Es handelt sich höchstwahrscheinlich um einen Fehler in der Implementierung der Broadcastfunktion. Da dieser Fehler die Funktionalität des Prototypen und die Demonstration der Machbarkeit des Konzepts nicht signifikant beeinflusst, wurde seiner Lokalisierung und Behebung allerdings keine Priorität eingeräumt.

6.1.3 Erkenntnisse

Trotz der anfänglichen Schwierigkeiten zeigen die Ergebnisse der Evaluation des Prototypen, dass ein stabiler Betrieb einer Peer-to-Peer-Kollaborationslösung im Webbrowser auf Basis der vorgestellten Systemarchitektur möglich ist.

Deutlich wurde zudem, dass die Gewährleistung einer guten Performance des Systems nicht so trivial ist, wie vorerst angenommen. Die angestrebte hohe Geschwindigkeit und geringe Latenz ließ sich mit dem vorliegenden Prototypen, trotz relativ geringer Datenmengen, unter realen Netzwerkbedingungen nicht ohne Einschränkungen erreichen. Eine Möglichkeit der Optimierung liegt in der Verwendung des in Abschnitt 4.4.5 beschriebenen Binärformats für den Datenaustausch. Ob die Performanceprobleme auch dem relativ frühen Stadium der WebRTC-Implementierungen geschuldet sind, bleibt vorerst ungeklärt. In zukünftigen Arbeiten könnte dieser Aspekt weitergehend betrachtet werden.

6.2 Reflexion des Architekturentwurfs

Eine vollständige Evaluation des im Laufe dieser Arbeit entwickelten Architekturentwurfs ist, wie bereits angesprochen, zum gegenwärtigen Zeitpunkt und auf Basis des vorliegenden Prototypen nicht möglich. Stattdessen wird an dieser Stelle eine theoretische Reflexion anhand der in Kapitel 4 identifizierten High-Level-Anforderungen vorgenommen. Erkenntnisse aus der Entwicklung und Evaluation des Prototyps können, wo sinnvoll, einfließen.

6.2.1 Erfüllungsgrad der Anforderungen

Unabhängigkeit von zentraler Infrastruktur

Eine der Kernanforderungen an die Architektur stellt die Unabhängigkeit von zentraler Infrastruktur dar. Auf Basis der momentan verfügbaren Browsertechnologien ist eine vollständige Unabhängigkeit nicht zu erreichen, weil für die Vermittlung einer WebRTC-Verbindung immer ein Signaling-Kanal benötigt wird. Deshalb verfügt auch der vorgestellte Architekturentwurf über eine zentrale Komponente. Die Rolle dieser Komponente wurde jedoch so weit wie möglich eingeschränkt. Sie enthält keine Anwendungslogik, benötigt keine persistente Datenhaltung und ist aufgrund der Verwendung standardisierter Technologien (WebSockets) und Paradigmen (Publish Subscribe) leicht austauschbar. Die Interaktion der Webapplikation mit der zentralen Komponente wurde auf ein Minimum reduziert, sodass sie, eine entsprechende Weiterentwicklung der Browsertechnologien vorausgesetzt, mit geringfügigen Änderungen aus der Architektur entfernt werden könnte. Der Erfüllungsgrad der Anforderung ist somit, nach gegenwärtigen Maßstäben, hoch.

Eine Möglichkeit für den Verzicht auf zentrale Infrastruktur wäre die Integration von Multicast DNS [vgl. CK13] durch die Browserhersteller. Diese Technologie könnte Webapplikationen die Möglichkeit einräumen, im lokalen Netzwerk andere Instanzen zu finden und grundlegende Informationen für einen Verbindungsaufbau mit ihnen auszutauschen.

Performance und Geschwindigkeit

Bei der Erarbeitung des Architekturentwurfs wurde der Aspekt der Performance und Geschwindigkeit von Kollaborationsanwendungen berücksichtigt und floss in einige Teile der Architektur mit ein. Hierzu gehören die weitgehend asynchrone Interaktion der Systemkomponenten untereinander, die Entwicklung eines effizienten Broadcasting-Mechanismus und die Verwendung eines binären Datenformats. Wie sich allerdings bei der Evaluation des Prototypen herausstellte, bleibt die Performance trotz dieser Maßnahmen ein kritischer Faktor (vgl. Abschnitt 6.1.3).

Die Architektur alleine kann keine hohe Performance gewährleisten. Sie stellt nur eine relativ generische Plattform zur Verfügung und räumt den konkreten Kollaborationsanwendungen weitgehende Freiheiten ein. Diese müssen daher ebenfalls Performancefaktoren berücksichtigen. Das zeigte sich beispielsweise in der prototypischen Implementierung. Eine zu häufige Anpassung des Models durch eine Presenter-Klasse führte hier

zu einer Überlastung der Netzwerkverbindungen mit entsprechenden Nachrichten. Kollaborationsanwendungen sollten effiziente Mechanismen zur Synchronisation der Daten implementieren.

Flexibilität und Wiederverwendbarkeit

Die Flexibilität und Wiederverwendbarkeit von Systemkomponenten spielte bei der Ausarbeitung der Architektur ebenfalls eine wichtige Rolle. Ob dieses Ziel erreicht wurde, lässt sich anhand eines einzelnen Prototypen kaum objektiv abschätzen. Während der Entwicklung gestaltete sich die Umsetzung von Änderungen und Erweiterungen am Prototypen relativ unproblematisch, was dafür spricht, dass die Anforderung der Flexibilität zumindest in Teilen erfüllt wurde. Eine weitere Praxiserprobung des Konzepts könnte hier mehr Aufschluss geben.

Persistenz

Der Architekturentwurf ermöglicht eine persistente Speicherung von Daten. Es kommt, wie in Abschnitt 4.6 beschrieben, ein Key-Value-Store auf Basis der Indexed Database API zum Einsatz. Für den Anwendungsfall des Prototypen, stellten sich die gewählten Technologien und Paradigmen als geeignet heraus, was allerdings nicht bedeutet, dass sie sich für alle Arten von Kollaborationsanwendungen gleichermaßen eignen. IndexedDB ermöglicht es auch, komplexere Datenbankstrukturen zu realisieren. Ob bestimmte Anwendungsfälle hiervon profitieren könnten, bleibt an dieser Stelle offen.

Um für einzelne Anwendungen eine größere Flexibilität im Bezug auf die persistente Datenhaltung zu erreichen, wäre es denkbar, diese nicht als Teil der Plattform zu deklarieren sondern in anwendungsspezifischen Code auszulagern. Eine Möglichkeit hierfür wäre die Integration der Datenhaltung in das Model. Dieser Schritt hätte allerdings wiederum Nachteile bezüglich der Flexibilität und Wiederverwendbarkeit von Systemkomponenten.

Konsistenz

Die Anforderung der Konsistenz von Daten wird durch den vorgestellten Architekturentwurf nur indirekt abgedeckt. Die konkrete Ausgestaltung der Datensynchronisation wurde nicht weiter betrachtet und ist Teil des anwendungsspezifischen Implementierung des Models. Es liegt allerdings nahe, dass sich über die, von der Plattform zur Verfügung gestellte, Kommunikationsinfrastruktur entsprechende Methoden umsetzen lassen. Es exis-

tieren viele Forschungsarbeiten, die sich mit der Entwicklung oder Anpassung entsprechender Algorithmen für Peer-to-Peer-Netzwerke beschäftigen [vgl. z.B. Ost+06; CF07]

Unterstützung verschiedener Datentypen

Beim Entwurf der Architektur wurde die Unterstützung verschiedener Datentypen insofern berücksichtigt, dass ein binäres Nachrichtenformat und eine persistente Datenhaltung gewählt wurden, die die Übertragung bzw. Speicherung beliebiger Datentypen ermöglichen. Auch wenn in der prototypischen Umsetzung nur textbasierte Daten übertragen wurden, legt das Spektrum von bereits verfügbaren Webapplikationen nahe, dass eine Verarbeitung nahezu beliebiger Datenformate im Browser möglich ist. Die Anforderung ist somit grundlegend erfüllt. Dabei wäre es in Zukunft von Interesse zu untersuchen, inwiefern sich größere Datenmengen auf die Performance des Systems auswirken.

Als problematisch könnten sich solche Datenformate herausstellen, deren Verarbeitung komplexe Berechnungen oder die schnelle Verarbeitung großer Datenmengen erfordern, beispielsweise Videoformate. Die JavaScript-Runtimes in Webbrowsern wurden für solche Aufgaben nicht ausgelegt. In klassischen Webapplikationen werden sie oft auf externe Server ausgelagert, was in einer Peer-to-Peer-Anwendung nicht möglich ist.

Interoperabilität und Integration

Die Interoperabilität mit anderen Softwaresystemen ist im Architekturentwurf in Form einer Im- und Exportmöglichkeit für Dateien auf dem lokalen Dateisystem des Nutzer abgedeckt. Diese Funktionalität wurde in der prototypischen Webapplikation erprobt. Die Anforderung ist somit erfüllt, zum gegenwärtigen Zeitpunkt allerdings in einem Maße, welches hinter den Möglichkeiten nativer Anwendungen zurückbleibt. Native Software verfügt über Möglichkeiten zur Interprozesskommunikation, welche im Browser vor allem aus Sicherheitsgründen nicht verfügbar sind. Hier können zukünftige technische Entwicklungen möglicherweise Abhilfe schaffen.

Datenschutz und Sicherheit

Wie in Abschnitt 4.8 erläutert, bietet die verwendete Architektur durch die Verwendung einer Transportverschlüsselung ein gewisses Mindestmaß an Sicherheit. Mögliche weitere Sicherheitsmaßnahmen wurden ebenfalls angesprochen, sind jedoch bisher nicht Teil

des Architekturentwurfs und wurden auch nicht im Prototypen implementiert. In dieser Hinsicht besteht Bedarf für eine weitere Auseinandersetzung mit Sicherheitsaspekten.

Einfache Initialisierung

Die Initialisierung des Systems erfordert den Start einer Serversoftware auf dem Endgerät eines Nutzers und die Verteilung einer URL an alle anderen Teilnehmer der Kollaborationssitzung. Da die URL eine lokale IP-Adresse und einen Sitzungsidentifier enthält, ist sie relativ komplex und nur bedingt menschenlesbar. Es ist deshalb sinnvoll, die URL über ein technisches Medium zu übertragen. Hierfür generiert die prototypische Webapplikation einen QR-Code. Auf Desktop-Geräten ist die Verwendung eines QR-Codes nur bedingt sinnvoll. Alternativ kann die URL daher auch mittels einer Instant-Messaging-Software oder per Email ausgetauscht werden.

Die Initialisierung benötigt also nur wenige Schritte, erfordert aber in vielen Fällen einen bestehenden Kommunikationskanal, über den die URL zur Verfügung gestellt werden kann. Abhilfe könnte die Verwendung eines besser menschenlesbaren Sitzungsidentifiers schaffen. Allerdings bleibt dabei das Problem der lokalen IP-Adresse bestehen, die gerade in IPv6-Netzwerken sehr komplex sein kann. Falls die beteiligten Betriebssysteme dies unterstützen, wäre eine serverseitige Verwendung von Multicast DNS [vgl. CK13] möglich, um dem System einen menschenlesbaren und im lokalen Netzwerk gültigen Hostnamen zuzuweisen. Hier sind in Zukunft einige Optimierungen denkbar.

Awareness

Der Architekturentwurf sieht eine eigene Komponente für die Erfassung und den Austausch von Awarenessinformationen vor, die auch in der prototypischen Webapplikation umgesetzt und erprobt wurde. Die Anforderung der Unterstützung von Awarenessinformationen ist also erfüllt. Relevant für folgende Forschungsarbeiten wäre vor allem die Frage, welche Awarenessinformationen in einem lokalen Kontext von Bedeutung sind.

Unterstützung individueller Arbeit

Die Architektur sieht explizit eine separate Modelkomponente für private Informationen vor. Diese wurde im Prototypen erfolgreich dazu verwendet, eine individuelle Arbeit von einzelnen Teilnehmern an eigenen Artefakten zu ermöglichen. Somit ist die Anforderung erfüllt.

6.2.2 Erkenntnisse

Bei der Reflexion des Architekturentwurfs wurde deutlich, dass ein Großteil der High-Level-Anforderungen erfüllt oder zu einem gewissen Grad erfüllt ist. Dennoch bietet der aktuelle Stand der Architektur in einigen Bereichen Potential für Optimierungen und Erweiterungen. Hierzu gehören vor allem die Bereiche der Performance, Sicherheit und der Initialisierung von Kollaborationssitzungen. Einige Möglichkeiten für Verbesserungen wurden im vorherigen Abschnitt bereits angesprochen.

7 Fazit & Ausblick

7.1 Fazit

Die vorliegende Arbeit setzt sich mit der Frage auseinander, ob es möglich ist, auf Basis von heute in Webbrowsern verfügbaren Technologien, eine Plattform zur Unterstützung von Face-to-Face-Kollaboration umzusetzen, die weitestgehend auf zentrale Infrastruktur, eine Installation und komplexe Initiationsschritte verzichtet und dennoch einen mit bestehenden Kollaborationslösungen vergleichbaren Funktionsumfang bietet. Zu diesem Zweck wurde, auf Basis vorher erarbeiteter Kriterien, eine Softwarearchitektur für eine browserbasierte Peer-to-Peer-Kollaborationplattform entworfen. Diese Architektur wurde in Form einer prototypischen Webapplikation für die kollaborative Erstellung von Mindmaps umgesetzt. Sowohl die Architektur als auch der Prototyp wurden einer Evaluation bzw. kritischen Reflexion unterzogen.

Die Beantwortung der Forschungsfrage sollte, auf Basis der gewonnenen Erkenntnisse, differenziert betrachtet werden. Es stellt sich heraus, dass bestehende Ansätze aus dem Peer-to-Peer-Bereich für Browsertechnologien adaptiert und für die Implementierung von Kollaborationslösungen verwendet werden können. Ein vollständiger Verzicht auf zentrale Infrastruktur ist, aufgrund der Beschränkungen dieser Technologien, zum momentanen Zeitpunkt nicht möglich. Allerdings konnte mit der dargelegten Softwarearchitektur ein Ansatz präsentiert werden, welcher die Rolle einer zentralen Komponente stark eingrenzt. Im gleichen Zuge wurde gezeigt, dass auf eine Installation und Konfiguration des Systems in weiten Teilen verzichtet werden kann. Zudem sind für die Initialisierung einer Kollaborationssitzung mit mehreren Teilnehmern nur wenige Schritte notwendig.

Die Betrachtung der verfügbaren Browsertechnologien und deren Einsatz im vorgestellten Prototypen haben gezeigt, dass sich mit Peer-to-Peer-Kollaborationsanwendungen im Browser in vielen Fällen ein vergleichbarer Funktionsumfang erreichen lässt, wie mit nativen oder „cloud"-basierten Anwendungen. Das schließt eine Integration mit anderen Anwendungen ein. Allerdings wurde auch deutlich, dass spezifische Herausforderungen existieren, insbesondere im Bereich der Performance. Bestimmte Arten von Kollaborationsanwendungen, die mit großen Datenmengen oder aufwändigen Berechnungen agieren müssen, sind daher in einer Webapplikation nur bedingt umzusetzen.

7.2 Ausblick

Während der Ausarbeitung ergaben sich weitergehende Fragen, die Potential für zukünftige Forschungsarbeiten bieten und von denen einige an dieser Stelle genannt werden sollen.

Ein konkretes Verfahren zur Synchronisation und Gewährleistung der Konsistenz von Daten während der gleichzeitigen Bearbeitung wurde in der vorgestellten Softwarearchitektur nicht genauer spezifiziert. Daher stellt sich die Frage, welche Verfahren für welchen Anwendungsfall geeignet sind, inwiefern sie sie auf Basis von Technologien wie WebRTC umgesetzt werden können und welche Anpassungen hierfür notwendig wären.

Ein weiterer Aspekt, der sich für eine tiefergehende Betrachtung eignen würde, sind Datenschutz und Sicherheit. Es existieren zahlreiche Konzepte für die Gewährleistung von Datensicherheit in Peer-to-Peer-Netzwerken. Hier wäre ebenfalls untersuchungswürdig, welche dieser Konzepte sich für den Einsatz in browserbasierten Peer-to-Peer-Netzwerken eignen und wie diese angepasst werden müssten.

Geschwindigkeit und Performance als kritische Faktoren in Kollaborationsanwendungen wurden bereits angesprochen. Da sich dieser Faktor bei der Implementierung des auf WebRTC basierenden Peer-to-Peer-Netzwerks als Herausforderung herausstellte, wäre es angebracht, sich in zukünftigen Arbeiten mit diesbezüglichen Optimierungspotentialen auseinanderzusetzen.

Die vorliegende Arbeit konnte die Machbarkeit von dezentral organisierten Kollaborationsanwendungen auf Basis von Peer-to-Peer Browsertechnologien darlegen und Herausforderungen aufzeigen, die bei deren Implementierung auftreten können. Es ist zu hoffen, dass zukünftige technische Entwicklungen einige dieser Herausforderungen aus dem Weg räumen und derartigen Anwendungen weiterhin neue Einsatzgebiete ermöglichen, die bisher nativen Applikationen vorbehalten waren.

Literaturverzeichnis

[15] *js-sha512 - A simple SHA-512, SHA-384, SHA-512/224, SHA-512/256 hash functions for JavaScript supports UTF-8 encoding*. 2015. URL: https://github.com/emn178/js-sha512 (besucht am 29. 09. 2015).

[BC06] V. Braun und V. Clarke. „Using thematic analysis in psychology“. In: July 2014 (2006), S. 37–41. ISSN: 1478-0887. DOI: 10.1191/1478088706qp063oa. URL: http://eprints.uwe.ac.uk/11735/1/thematic%5C_analysis%5C_revised%5C_-%5C_final.doc.

[Bel14] David Bellingroth. „Dezentrale soziale Netzwerke“. 2014.

[Ber+14] Robin Berjon u. a. *HTML5*. Candidate Recommendation. W3C, 2014. URL: http://www.w3.org/TR/2014/CR-html5-20140731/.

[Ber+15] Adam Bergkvist u. a. *Webrtc 1.0: Real-time communication between browsers*. W3C Working Draft. W3C, Feb. 2015. URL: http://www.w3.org/TR/2015/WD-webrtc-20150210/.

[Cal14a] Oliver Caldwell. *EventEmitter - Event based JavaScript for the browser*. 2014. URL: https://github.com/Olical/EventEmitter (besucht am 29. 09. 2015).

[Cal14b] Oliver Caldwell. *Heir - Makes prototypical inheritance easy and robust*. 2014. URL: https://github.com/Olical/Heir (besucht am 29. 09. 2015).

[CF07] M Cart und J Ferrie. „Asynchronous reconciliation based on operational transformation for P2P collaborative environments“. In: *Collaborative Computing: Networking, Applications and Worksharing, 2007. CollaborateCom 2007. International Conference on*. Nov. 2007, S. 127–138. DOI: 10.1109/COLCOM.2007.4553821.

[CK13] S Cheshire und M Krochmal. *Multicast DNS*. RFC 6762 (Proposed Standard). 2013. URL: http://www.ietf.org/rfc/rfc6762.txt.

[Cog15] James Coglan. *Faye: Simple pub/sub messaging for the web*. 2015. URL: http://faye.jcoglan.com (besucht am 29. 09. 2015).

[DB92] Paul Dourish und Victoria Bellotti. „Awareness and Coordination in Shared Workspaces“. In: *Proc. Intl. Conf. on Computer-Supported Cooperative Work* November (1992), S. 107–114. ISSN: 15365050. DOI: 10.1145/143457.143468. arXiv: 0812.1045v1. URL: http://portal.acm.org/citation.cfm?doid=143457.143468.

[Dev15] Alexis Deveria. *Can I use... Support tables for HTML5, CSS3, etc*. 2015. URL: http://caniuse.com (besucht am 29. 09. 2015).

[EG89] C. a. Ellis und S. J. Gibbs. „Concurrency control in groupware systems“. In: *ACM SIGMOD Record* 18.2 (1989), S. 399–407. ISSN: 01635808. DOI: 10.1145/66926.66963.

[EGR91] Clarence A. Ellis, Simon J. Gibbs und Gail Rein. „Groupware: some issues and experiences“. In: *Communications of the ACM* 34.1 (Jan. 1991), S. 39–58. ISSN: 00010782. DOI: 10.1145/99977.99987. URL: http://dl.acm.org/citation.cfm?id=99977.99987.

[FM11] I Fette und A Melnikov. *The WebSocket Protocol*. RFC 6455 (Proposed Standard). 2011. URL: http://www.ietf.org/rfc/rfc6455.txt.

[Hic15] Ian Hickson. *Web Storage (Second Edition)*. Techn. Ber. June. W3C, 2015. URL: http://www.w3.org/TR/2015/CR-webstorage-20150609/.

[Hol11] TJ Holowaychuk. *Mocha - the fun, simple, flexible JavaScript test framework*. 2011. URL: http://mochajs.org (besucht am 29. 09. 2015).

[Hol15] TJ Holowaychuk. *should.js - BDD style assertions for node.js*. 2015. URL: https://github.com/shouldjs/should.js (besucht am 29. 09. 2015).

[Hor+91] M. Horton u. a. „The impact of face-to-face collaborative technology on group writing". In: *Proceedings of the Twenty-Fourth Annual Hawaii International Conference on System Sciences* iv (1991). ISSN: 07421222. DOI: 10.1109/HICSS.1991.184073.

[KGB02] Gerd Kortuem, Hans-Werner Gellersen und Mark Billinghurst. „Mobile ad hoc collaboration". In: *CHI '02 extended abstracts on Human factors in computing systems - CHI '02*. New York, New York, USA: ACM Press, Apr. 2002, S. 931. ISBN: 1581134541. DOI: 10.1145/506443.506665. URL: http://dl.acm.org/citation.cfm?id=506443.506665.

[Lar10] James R. Larson. „Idea Generation - Creative Thinking in Groups". In: *In Search of Synergy in Small Group Performance*. 2010. Kap. 3, S. 73–120. ISBN: 0805859438. URL: https://books.google.com/books?hl=de%5C&lr=%5C&id=zloA3bMffI4C%5C&pgis=1.

[Li+04] Yong Li Yong Li u. a. „Reliable communication based on P2P architecture on real-time collaborative editing system". In: *8th International Conference on Computer Supported Cooperative Work in Design*. Bd. 1. 2004, S. 244–249. ISBN: 0-7803-7941-1. DOI: 10.1109/CACWD.2004.1349023.

[McM15] Caolan McMahon. *Async.js - Async utilities for node and the browser*. 2015. URL: https://github.com/caolan/async (besucht am 29. 09. 2015).

[Meh+15] Nikunj Mehta u. a. *Indexed Database API*. Techn. Ber. W3C, 2015. URL: http://www.w3.org/TR/2015/REC-IndexedDB-20150108/.

[Mic14] Microsoft Corporation. *Bringing Interoperable Real-Time Communications to the Web*. 2014. URL: http://blogs.msdn.com/b/ie/archive/2014/10/27/bringing-interoperable-real-time-communications-to-the-web.aspx (besucht am 29. 09. 2015).

[Moz13] Mozilla Developer Network. *IndexedDB Storage limits*. 2013. URL: https://developer.mozilla.org/de/docs/IndexedDB%5C#Storage%5C_limits (besucht am 10. 09. 2015).

[Moz15a] Mozilla Developer Network. *JavaScript typed arrays*. 2015. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Typed%5C_arrays (besucht am 13. 08. 2015).

[Moz15b] Mozilla Developer Network. *Same-origin policy*. 2015. URL: https://developer.mozilla.org/en-US/docs/Web/Security/Same-origin%5C_policy (besucht am 29. 09. 2015).

[Moz15c] Mozilla Developer Network. *Structured Clone Algorithm - Benefits over JSON*. 2015. URL: https://developer.mozilla.org/en-US/docs/Web/API/Web%5C_Workers%5C_API/Structured%5C_clone%5C_algorithm%5C#Benefits%5C_over%5C_JSON (besucht am 10. 09. 2015).

[Ney+09] Andrés Neyem u. a. „An Architectural Pattern for Mobile Groupware Platforms“. In: *Proceedings of the Confederated International Workshops and Posters on On the Move to Meaningful Internet Systems: ADI, CAMS, EI2N, ISDE, IWSSA, MONET, OnToContent, ODIS, ORM, OTM Academy, SWWS, SEMELS, Beyond SAWSDL, and COMBEK 2009*. OTM '09. Berlin, Heidelberg: Springer-Verlag, 2009, S. 401–410. ISBN: 978-3-642-05289-7. DOI: 10.1007/978-3-642-05290-3_52. URL: http://dx.doi.org/10.1007/978-3-642-05290-3%5C_52.

[Nod15] Node.js Foundation. *Node.js*. 2015. URL: https://nodejs.org/en/ (besucht am 29. 09. 2015).

[Ols+93] Judith S. Olson u. a. „Groupwork close up: a comparison of the group design process with and without a simple group editor“. In: *ACM Transactions on Information Systems* 11.4 (1993), S. 321–348. ISSN: 10468188. DOI: 10.1145/159764.159763.

[Ost+06] Gérald Oster u. a. „Data consistency for P2P collaborative editing“. In: *Proceedings of the 2006 20th ...* (2006), S. 259–267. DOI: 10.1145/1180875.1180916. URL: http://dl.acm.org/citation.cfm?id=1180916.

[Pat91] John F Patterson. „Comparing the Programming Demands of Single-user and Multi-user Applications“. In: *Proceedings of the 4th Annual ACM Symposium on User Interface Software and Technology*. UIST '91. New York, NY, USA: ACM, 1991, S. 87–94. ISBN: 0-89791-451-1. DOI: 10.1145/120782.120792. URL: http://doi.acm.org/10.1145/120782.120792.

[PE10] Timothy Perfitt und Burkhard Englert. „Megaphone: Fault tolerant, scalable, and trustworthy P2P microblogging“. In: *5th International Conference on Internet and Web Applications and Services, ICIW 2010* (2010), S. 469–477. DOI: 10.1109/ICIW.2010.77.

[QB06] Yi Qiao und E Bustamante. „A measurement-based view of two P2P systems that people use“. In: *Electrical Engineering* (2006), S. 341–355.

[RD01] Antony Rowstron und Peter Druschel. „Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems“. In: *Middleware 2001* 2218.November 2001 (2001), S. 329–350. ISSN: 03029743. DOI: 10.1007/3-540-45518-3. URL: http://www.springerlink.com/index/10.1007/3-540-45518-3.

[Reh13] Rodney Rehm. *URI.js*. 2013. URL: http://medialize.github.io/URI.js/ (besucht am 29. 09. 2015).

[RG92] M. Roseman und S. Greenberg. „GroupKit: {A} groupware toolkit for building real-time conferencing applications“. In: *Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW'92)* November (1992), S. 43–50. DOI: 10.1145/143457.143460.

[Ros+02] J Rosenberg u. a. *SIP: Session Initiation Protocol*. Internet Requests for Comments. 2002. URL: http://www.ietf.org/rfc/rfc3261.txt.

[RPR06] V Roussev, G P Priego und G G Richard. „TouchSync: Lightweight Synchronization for Ad-Hoc Mobile Collaboration“. In: *Collaborative Technologies and Systems, 2006. CTS 2006. International Symposium on*. 2006, S. 181–188. DOI: 10.1109/CTS.2006.67.

[RS15] Arun Ranganathan und Jonas Sicking. *File API*. Last Call {WD}. W3C, Apr. 2015. URL: http://www.w3.org/TR/2015/WD-FileAPI-20150421/.

[San15] Shim Sangmin. *QRCode.js - Cross-browser QRCode generator for javascript*. 2015. URL: https://github.com/davidshimjs/qrcodejs (besucht am 29. 09. 2015).

[SM11] G Saucedo-Tejada und S Mendoza. „An architecture for supporting face-to-face mobile interaction". In: *Electrical Engineering Computing Science and Automatic Control (CCE), 2011 8th International Conference on*. 2011, S. 1–6. DOI: 10.1109/ICEEE.2011.6106576.

[SMA15] SMART Technologies. *SMART Notebook® Collaborative Learning Software*. 2015. URL: http://education.smarttech.com/de-de/products/notebook (besucht am 29. 09. 2015).

[Str15] Inc. StrongLoop. *Express - Fast, unopinionated, minimalist web framework for Node.js*. 2015. URL: http://expressjs.com (besucht am 29. 09. 2015).

[TBB58] Donald W Taylor, Paul C Berry und Clifford H Block. „Does Group Participation When Using Brainstorming Facilitate or Inhibit Creative Thinking?" In: *Administrative Science Quarterly* 3.1 (1958), pp. 23–47. ISSN: 00018392. URL: http://www.jstor.org/stable/2390603.

[The15] The jQuery Foundation. *jQuery*. 2015. URL: http://jquery.com (besucht am 29. 09. 2015).

[Twi15] Inc. Twitter. *Bootstrap · The world's most popular mobile-first and responsive front-end framework*. 2015. URL: http://getbootstrap.com.

[Wan15] Roger Wang. *NW.js*. 2015. URL: http://nwjs.io (besucht am 03. 09. 2015).

[WSF05] Alf Inge Wang, Carl-Fredrik Sørensen und Thomas Fossum. „Mobile peer-to-peer technology used to promote spontaneous collaboration". In: (Mai 2005), S. 48–55. URL: http://dl.acm.org/citation.cfm?id=1895420.1895440.

[ZA15] Juriy Zaytsev und Leon Arnott. *ECMAScript 6 compatibility table*. 2015. URL: http://kangax.github.io/compat-table/es6/ (besucht am 29. 09. 2015).

Anhang

Aus praktischen Erwägungen, vor allem der Einsparung von Papier, liegt der Anhang ausschließlich in digitaler Form vor. Er befindet sich auf dem beigelegten Datenträger. Die Ordnerstruktur inklusive einiger wichtiger Dateien wird im Folgenden erläutert:

- /
  - **Kontextanalyse**
    - `Interviewtranskripte` ..... Transkripte der Interview inklusive Codings
    - `Codings` ..... Einzelne Codings zu bestimmten Themen
    - `Transkript Fallstudie.pdf` ..... Transkript der Fallstudie
  - **Prototyp**
    - `Ausführbar` ..... Ausführbare Versionen des Prototypen
    - `Quelltext` ..... Gesamter Quelltext des Prototypen
      - `README.md` ..... Erläuterungen zum Quelltext (**wichtig**)
      - `webapp` ..... Quelltext der Webanwendung
  - **Evaluation**
    - `Logfile.log` ..... Bei der Evaluation erstellte Logdatei
    - `Logfile.html` ..... Visuell aufbereitete Version der Logdatei
    - `Logfile Fehlschläge.log` ..... Logdatei der fehlgeschlagenen Durchläufe
    - `Mindmap.json` ..... JSON-Export der erstellen Mindmap
    - `Screenshot.png` ..... Screenshot der finalen Mindmap
    - `Screenvideo.mp4` ..... Zeitraffer-Screenvideo der Evaluation

Eidesstattliche Erklärung

Ich versichere, die von mir vorgelegte Arbeit selbständig verfasst zu haben.

Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder nicht veröffentlichten Arbeiten anderer entnommen sind, habe ich als entnommen kenntlich gemacht. Sämtliche Quellen und Hilfsmittel, die ich für die Arbeit benutzt habe, sind angegeben.

Die Arbeit hat mit gleichem Inhalt bzw. in wesentlichen Teilen noch keiner anderen Prüfungsbehörde vorgelegen.

Gummersbach, 01. Oktober 2015

David Bellingroth