Dies ist der Titel der Bachelorarbeit und bitte ohne Rechtschreibfehler

Bachelorarbeit

für die Prüfung zum

Bachelor of Science

des Studiengangs Informatik

an der Dualen Hochschule Baden-Württemberg Heidenheim

von

Michael Lustig

Septemper 2017

Bearbeitungszeitraum	12 Wochen
Matrikelnummer, Kurs	1234510, TINF2014MI
Ausbildungsbetrieb	Firma GmbH, Firmenort
Erstgutachter	Dipl.-Ing. (FH) Peter Pan
Zweitgutachter	Prof. Dr. Rolf Assfalg

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Listings

1 Einführung

Wir leben in einer Zeit, in der der Einsatz von Technik im Alltag nicht mehr weg zu denken ist. Die Menschheit ist so stark vom technischen Fortschritt und den daraus resultierenden Entwicklungen geprägt wie nie zuvor. Technologien wie Autos, zu deren Betrieb eine körperliche Interaktion nicht mehr von Nöten ist, wurden vor wenigen Jahren noch als Science Fiction klassifiziert und sind trotz allem heute realisierbar. Auch In Fabriken werden selbst komplexere, erfolgsentscheidende Tätigkeiten von Robotern umgesetzt und zu Hause gehört das eigenständige Staubsaugen der Vergangenheit an, weil dies eine programmierte Maschine zuverlässig übernehmen kann.
Neue Technologien wirken sich in vielen Alltagssituationen bereichernd und erleichternd auf unser Leben aus. Ein hiervon stark betroffener Wirtschaftssektor ist das wissens- und technologiegetriebene Gesundheitswesen. Der medizintechnische Fortschritt, der sich in den letzten Jahrzehnten ereignet hat, ermöglicht weitaus zielführendere Diagnose- und Therapiemöglichkeiten als sie früher vorstellbar waren. So lassen sich beispielsweise in der diagnostischen Fachabteilung Radiologie mit dem Einsatz neuer Technologien, wie Computer- und Kernspintomographien, aussagekräftigere Befunde erzielen als sie ein Pathologe früher durch einen händischen Eingriff hätte anfertigen können [Buc13, S.3].
Die Statistiken des statistischen Bundesamtes zur durchschnittlichen Lebenserwartung der Bevölkerung Deutschlands zeigen einen stetigen Aufschwung im Verlauf der vergangenen Jahrzehnte und lassen daher auf einen erheblichen Anstieg der allgemeinen Patientenversorgungsqualität im Gesundheitswesen aufgrund der Entwicklung IT-basierter Medizintechnologien schließen.
Diese Tatsache gilt als ausschlaggebender Beweggrund dafür, die Studienarbeit im Rahmen eines medizininformatischen Entwicklungsprojektes zu absolvieren. Der Fokus ist hierbei auf Menschen gerichtet, die von Hörschädigungen betroffen sind und tagtäglich mit ihren schwerwiegenden Folgen konfrontiert werden. Soziale Integrationsschwierigkeiten und psychische Beeinträchtigungen bilden nur das Grundgerüst der gängigen Symptomatik. Das Ziel des Projekts besteht darin, den oben genannten Belastungen durch Hörstörungen mithilfe eines IT-Produktes entgegenzuwirken.
Als Entwicklungsgrundlage dient eine Technologie mit dem Namen Augmented Reality (zu Deutsch: Erweiterte Realität). Darunter lässt sich eine Variation der Ursprungstechnologie Virtual Reality verstehen, mit deren Hilfe Brillen entwickelt werden können, die dem User optisch das Gefühl vermitteln sich in einer virtuellen Welt zu befinden. Die reale Welt um ihn herum wird dabei vollständig ausgeblendet und ist für ihn somit nicht übersehbar.
Augmented Reality (kurz: AR) sah dies als Schwachstelle und versuchte dem entgegen-

zuwirken. Durch AR lassen sich digitale Daten oder computergenerierte Informationen auf Brillen visualisieren und in die reale Welt integrieren. Gegenüber einer VR-Brille ist die reale Welt Teil der AR-Technologie und somit für den User sichtbar. AR erlaubt die Erfassung und Verarbeitung von Daten, beispielsweise die Aufzeichnung und Analyse eines gesehenen Bildes, und ermöglicht die Projizierung der Ergebnisse auf die Brillengläser [KR12]. Gleichermaßen lässt sich diese Konvertierung auch in die entgegengesetzte Richtung vornehmen, wie es auch im hier behandelten Studienprojekt verwendet wird:
Der von einem Mobiltelefon aufgenommene Ton wird auf das Vorkommen von Sprachbausteinen untersucht und die erkannte Sprache in Text konvertiert. Der Text wird auf einer AR-Brille ausgegeben, die in Verbindung mit dem Mobiltelefon steht. So kann ein Mensch auch mit Hörschädigungen einem Vortrag oder einer Unterrichtsstunde akustisch und visuell problemlos folgen. Auch das kommunizieren mit Menschen, die der Gebärdensprache nicht mächtig sind, wird vereinfacht, da diese einfach in das Mikrofon des Mobiltelefons sprechen müssen. Durch die Integration des Konvertierers in ein Mobiltelefon, wäre es sogar möglich beim Telefonat die Stimme des Anrufers direkt zu verarbeiten und dem Hörgeschädigten live als Text auf die AR-Brillengläser zu projizieren.

2 Struktur

Die Applikation ist untergliedert in zwei Hauptbestandteile. Einen Recorder, welcher die Tonaufnahme durchführt und die erkannte Sprache in Text umwandelt, und ein Printer, welcher sich zu einer Augmenter Reality Brille verbindet und den vom Recorder erkannten Text forformatiert und darauf ausgibt. Das Bindeglied zwischen diesen beiden Komponenten ist das User Interface mit Klassenname MainActivity. Die MainActivity wird durch Usereingaben gesteuert. Sie erstellt bei bedarf neue Recorder oder Printer Instanzen, startet beziehungsweise stoppt die Aufnahme bzw die AR-Ausgabe und informiert den AR-Printer über neu verfügbare Ausgabetexte. Diese drei Komponenten werden im folgenden näher erleutert.

2.1 Main Activity

Wie bereits erwähnt ist die MainActivity das Zentrum der Applikation, ein UML-Klassendiagramm ist in XXX abgebildet.
Die bestandteile der MainActivity:

1. Ein Spinner zur Wahl einer verfügbaren Sprachkonvertierungsoption und ein beschreibendes Textfeld
2. Ein Spinner zur Wahl eines verfügbaren AR-Printers und ein beschreibendes Textfeld
3. Ein Button zum Starten bzw. Stoppens einer Konvertierung
4. Ein SideView Menü, welches Einstellungsmöglichkeiten enthält wie zum Beispiel das Festlegen der gesprochenen Sprache
5. Eine Recorder Instanz vom Typ IRecorder. IRecorder ist ein interface, welches von allen Recorder-Implementierungen verwendet wird, um eine einheitlich API zu gewährleisten. Dieses Interface wird im Abschnitt XXX näher erleutert.
6. Eine AR-Printer Instanz vom Typ IPrinter. Iprinter ist ebenfalls ein Interface, welches von allen AR-Printern zur Gewährleistung einer einheitlichen API implementiert wird und wird in Abschnitt XXX näher erleutert.
7. Eine boolsche Hilfsvariable, die Auskunft darüber gibt, ob aktuell ein Konvertierungsvorgang im Gange ist.

8. Zwei Integer-Variablen, die Auskunft über den aktuell gewählten Recorder bzw Printer geben.

Zu 1. Und 2.:
Die beiden Spinner, enthalten alle aktuell supporteten Printer bzw Recorder. Der Inhalt der Spinner wird dynamisch genereiert, hierbei spielt die Klasse Constants eine tragende Rolle.
!!Genaueres über die initialisierung eines spinner arrays!!
Jedem der beiden Spinner ist eine OnItemSelectListener Instanz zugeordnet. Der OnItemSelectListener, ruft im Falle des Printer Spinners die Methode setPrinterMode(int mode, String description) und im Falle des Recorder Spinners die Methode setRecorderMode(int mode, String description) auf. Diese beiden Methoden initialisieren den gewünschten AR-Printer beziehungsweise den Recorder/Converter. Die Funktionalität dieser Methoden wird XXXX näher beschrieben.
Zu 3.:
Der OnClickListener der Buttons fragt die Hilfsvariable (7) ab und ruft abhängig von deren Wert die Methoden IRecorder.startRecording() und IPrinter.startPrinting() oder die Methode notifyRecorderStop() der eigenen Klasse auf.
4. Menü: XXXX

Methoden der Klasse MainActivity:

onCreate() und initialize_components():
In der onCreate()-Methode, welche beim start der Applikation ausgeführt wird, werden zunächst all Ihre bestandteile initialisiert durch die initialize_components()-Methode.
setPrinterMode(int mode, String description)
Diese Methode prüft zuerst, ob im Moment ein Konvertierungsvorgang im Gange ist, durch das abfragen der boolschen Hilfsvariable isRecording. Im Falle einer laufenden Konvertierung wird eine Meldung ausgegeben, die den User darauf hinweißt, dass der AR-Printer während einer Konvertierung nicht gewechselt werden kann. Sonst wird nichts weiter unternommen. Ist keine Konvertierung im Gange wird überprüft, ob der aktuelle Wert in der Hilfsvariable PRINTER_MODE mit dem Wert des ausgewählten Printers in der übergebenen Variable mode übereinstimmt, denn auch dann muss keine weitere Aktion ausgeführt werden.
Wurde ein anderer Printer gewählt, als der gerade verwendete, wird die Methode generatePrinter(int mode, String description) der Klasse PrinterFactory aufgerufen. Die Methode generatePrinter prüft durch ein Switch Statement, welcher ARPrinter vom User gewünscht ist und gibt eine Instanz des gewünschten Printers and die aufrufende Methode zurück. Ist der übergebene Integer-Wert der Factory unbekannt, wird standardmäßig eine Instanz

der Klasse TextPrinter erzeugt. Dieser erzeugt ein TextFeld in der MainActivity, in dem Speech to Text Conversion Ergebnisse ausgegeben werden.
setRecorderMode(int mode, String description):
Diese Methode hat den gleichen Ablauf, wie die Methode setPrinterMode(), mit dem einzigen Unterschied, dass die Methode generateRecorder der Klasse RecorderFactory aufgerufen wird, welche eine IRecorder Instanz zurück gibt. Auch die Recorder Factory hat für unbekannte Eingaben einen Standardwert. Sie gibt einen TextFieldRecorder zurück, welcher ebenfalls ein Textfeld erzeugt und die Eingaben and den gewählten AR-Printer sendet. Dieser Recorder verfehlt zwar den Sinn der Speech to Text Conversion, ist aber eine Erleichterung um die Funktionalität eines einzelnes AR-Printer Moduls zu testen.

receiveResult()
notifyStopRecord()

2.2 Constants Klasse

In der Klasse Constants werden globale Applikationskonstanten abgelegt. So gibt es für jede IRecorder-Implementierung einen Integer Wert, welcher den Recorder repräsentiert. Außerdem gibt es für jeden Recorder eine beschreibende String-Variable. Eine Map, welche beim start der Anwendung erzeugt wird, bringt die String-Werte mit den jeweiligen Int-Werten in Verbindung. Analog dazu besteht die selbe Struktur für Instanzen des Typs IPrinter, die unterstützten AR-Printer verwalten zu können.

2.3 Interface für Augmented-Reality-Brillen-Printer

Das Interface IPrinter definiert ein Interface, welches den gesamten Kommunikationsverlauf zwischen MainActivity und dem Augmented Reality Gerätes ermöglicht. Für den AR-Connector wurde ein Interface erzeugt um dem Erweiterbarkeitsparadigma von Software gerecht zu werden:
Die Klasse MainActivity ist mit einer Instanz der Klasse IPrinter, also einer Klasse, welche das Interface IPrinter implementiert, assoziiert. Innerhalb der Klasse MainActivity werden ausschließlich Methoden des Interfaces aufgerufen, keine Methoden der Implementation selbt. Aus diesem folgt, dass um ein weiteres Augmented Realitty Gerät zu unterstützen, lediglich eine einzelne neue Klasse implementiert werden muss, welche das IPrinter-Interface implementiert und der Klasse MainActivity hinzugefügt werden muss. Durch das Interface

wird sichergestellt, dass alle Implementierungen über den selben Mindestmethodenumfang verfügen und Aufrufe innerhalb dieses Umfangs von allen Implementierungen verarbeitet werden können.

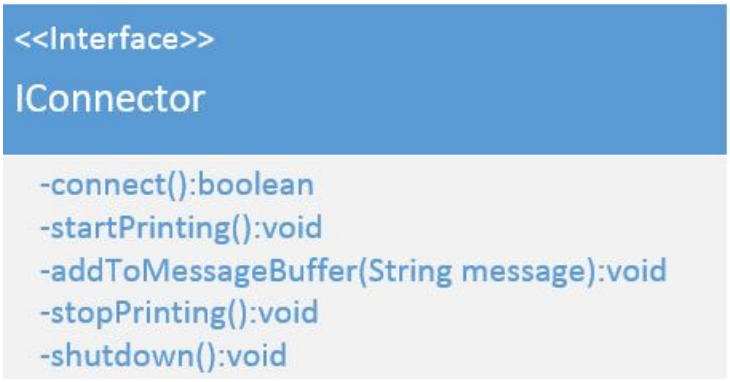

Abbildung 2.1: Interface für Augmented Reality Gerät Konnektoren

Interface-Methoden Das Interface verlangt die im Klassendiagramm **??** dargestellten Methoden von seinen Erben.
Die Methode connect() wird nach der Initialisierung einer neuen Connector Instanz von der MainActivity aufgerufen. Diese Methode, versucht eine Verbindung zu einem Augmented Reality Gerät aufzubauen. Ihr boolscher Rückgabewert gibt Auskunft über den Erfolg des Verbindungsaufbaus. Die startPrinting()-Methode versetzt den Connector in den Ausgabemodus. In diesem Zustand, wartet der Connector auf Nachrichten, die von der MainActivity Klasse durch die Methode addToMessageBuffer() in seinem Nachrichtenspeicher abgelegt werden und sendet diese an das Augmented Reality Gerät.
Die Methode stopPrinting() beendet diesen Zustand und mit der shutdown() Methode wird die Verbindung der App zum Augmented Reality Gerät beendet. Physikalisch kann die Verbindung dennoch weiterbestehen. Andere, Connectorspezifische Aktionen, welche bei Verbindungsende fällig werden, können ebenfalls in der shutdown()-Methode durchgeführt werden.

2.4 Interface für Speech-To-Text-Konvertierer

3 Graphical User Interfaces

3.1 ActionBar

3.2 NavigationView als Sidebar

3.3 Buttons / Buttondesigning

3.4 Spinner

3.5 TextViews

3.6 EventListener

3.7 SurfaceView für Animation

4 Canvas Animationen

4.1 DrawingThread / SoundAnimationThread

4.2 Die Methode „draw“

4.3 Die Methode „update“

4.4 Soundanimation durch animierte Canvas Komponenten

5 AR-Glass-Printer Implementierungen

Im folgenden werden implemntierungsspezifische Details über das Unterstützten verschiedener AR-Geräte näher erläutert. Die mit einem AR-Gerät kommunizierenden Module sind hier als AR-Printer bezeichnet. Jeder AR-Printer ist für die Dauer einer Verbindung mit dem ihm zugeteilten Gerät umfassend verantwortlich für die Kommunikation zwischen App und Gerät. Dies beinhaltet:

1. Den Verbindungsaufbau
2. Die Verarbeitung von von der Applikation gesendeten Anfragen zur Textausgabe auf dem AR-Gerät
3. Die Verarbeitung von vom AR-Gerät gesendeten Statusnachrichten
   Dies beinhaltet auch das Handling von Error-Codes.
4. Den Verbindungsabbau

Nachdem über das User Interface ein verfügbarer AR-Printer gewählt wurde, erzeugt eine Factory Klasse die jeweilige Printer-Instanz, welche im Konstruktor versucht eine Verbindung zum AR-Gerät aufzubauen. Ist der Verbindungsaufbau nicht möglich, wird der User über die fehlende Verbindung benachrichtigt und der Start-Record-Button deaktiviert, bis eine Verbindung zum Gerät hergestellt wurde oder vom User ein anderer AR-Printer gewählt wird.

5.1 Die Klasse AbstractPrinter

Die Klasse AbstractPrinter implementiert das Interface IPrinter und abstrahiert gemeinsames Verhalten aller AR-Printer. Die Implementierung eines spezifischen AR-Printers erfolgt über die Ableitung dieser Klasse und nicht über das Interface direkt. Dies dient der Vermeidung von Fehlern und der Vereinheitlichung der Arbeitsweise von unterschiedlichen Implementierungen.

Implementierte Methoden Während der Verbindungsauf- beziehungsweise -abbau geräte- und somit implementierungsspezifisch sind, kann das Verhalten der Methoden startPrinting(), addToMessageBuffer(String message), und stopPrinting() Geräteübergreifend implementiert werden.
Die Methode addToMessageBuffer(String message) erfüllt die Aufgabe, den übergebenen String zum Message Buffer hinzuzufügen. Der Message Buffer ist durch eine LinkedBlockingQueue realisiert und stellt somit eine Schlange dar, bei der Entnahmen immer am vorderen Ende geschehen und Elemente an ihrem Ende angefügt werden können. Sie arbeitet nach dem First-In-First-Out-Prinzip.
Die Methode startPrinting() erzeugt einen neuen Thread, welcher innerhalb einer eigenen Methode doPrinterJob() eine Schleife solange durchläuft, bis das boolsche Attribut isProcessing auf false gesetzt wurde, was bei Beendigung des Konvertierungsvorgangs der Fall ist. Innerhalb der Schleife wird überprüft, ob der Message Buffer neue Elemente zur Entnahme enthält. Ist dies der Fall, wird erfolgt ein Aufruf der abstrakten Methode printMessage(String message), welche die Nachricht im gerätespezifischen Protokol an das verbundene AR-Gerät sendet. Diese Methode muss für jeden spezifischen AR-Printer implementier eigens implementiert werden. Ein boolscher Rückgabewert gibt Auskunft über den Erfolg des Sendens der Nachricht. Liefert die Überprüfung der isProcessing-Variable false zurück, so deutet dies auf das Ende des Vorgangs hin. Um Ausgaben nicht apprupt zu beenden, wird neben des isProcessing-Attributes auch noch der Inhalt des Message Buffers geprüft. Is isProcessing zwar false, aber der Message Buffer enthält noch Elemente, die zur Ausgabe in Auftrag gegeben wurden, fährt der Printing-Thread fort, bis alle Ausgaben getätigt wurde.
Die Methode stopPrinting() setzt das boolsche Attribut isProcessing auf false. Dies führt zum Beenden der Schleife innerhalb des Printing-Threads und somit zur Beendigung des Threads an sich.

5.2 Support der GlassUp AR Brille

Die GlassUp Brille GlassUp ist ein italienisches Unternehmen, welches eine 65 Gramm **faq** schwere Augmented Reality Brille herstellt. Sie ist in 5.2 abgebildet. Die Brille befindet sich zwar noch in der Entwicklung, kann aber alles bieten, was zur Funktionalität unserer Applikation notwendig ist:
Nach eigener Aussage legt die Firma Wert darauf, eine unauffällige Brille zu produzieren, welche weniger einen Multimedialen Gegenstand als einen generellen Alltagshelfer darstellt **home**
So besitzt diese Brille keine Kamera, wie beispielsweise die von Google angebotene

l1 

Abbildung 5.1: Die GlassUp Augmented Reality Brille**home**

Alternative**google glasses** Eine Kamera gefährdet den Einsatz der Augmented Reality Brille insoweit, dass nach Paragraph 90 des Telekommunikationsgesetzes der Besitz von Gegenständen, "die ihrer Form nach einen anderen Gegenstand vortäuschen oder die mit Gegenständen des täglichen Gebrauchs verkleidet sind und auf Grund dieser Umstände [...] dazu bestimmt sind, das [...] Bild eines anderen von diesem unbemerkt aufzunehmen." Auch ist die GlassUp Brille mit einem Preis von 299 Euro **faq** noch erschwinglich. Die frei erhältliche SDK enthält außerdem einen Emulator, wodurch ein Brillenkauf für Entwicklungszwecke nicht von Anfang an zwangsläufig notwendig ist **SDK**
Die Brille wird durch die Bluetooth-Technik mit einem Mobiltelephon verbunden und Projeziert Texte und einfache Grafiken in der Farbe grün in den Mittelpunkt des Sichtfelds des Anwenders. Sie verfügt über ein Touchpad mit drei Eingabeknöpfen, durch die Smartphone-Anwendungen alternativ gesteuert werden können.

```
1 [
  caption
  ={
  Die

  Methode

  addMessag
  (
  String
```

Message
)

der

Klasse

GlassUpPr
}\
label
{
lst
:GlassUpPrinter: addMessageToLineList(String message)},captionpos=t]

```
/**
* method takes a message and splits it in lines of 16 chars
* to be printed to the AR device. The lines are added to this.
    linesToPrint
* @param messageBuffer
*/
private boolean addMessageToLineList(String messageBuffer){
int counter = 0;

if(messageBuffer == null){
return false;
}

while(counter < messageBuffer.length()) {

if (messageBuffer.length() - (counter + 17) <= 0) {
//rest of buffer is smaller than one line, -> prepare buffer and send
//then break
this.linesToPrint.add(messageBuffer.substring(counter));
break;
}
//after 17 signs there is a space --> perfect line
if (messageBuffer.charAt(counter + 17) == ' ') {
this.linesToPrint.add(messageBuffer.substring(counter, counter + 18));
counter += 18;
} else {
//check next ' ' before 17
boolean foundSpace = false;

for (int i = counter + 17; i > counter; i--) {
//space found?
if (messageBuffer.charAt(i) == ' ') {
this.linesToPrint.add(messageBuffer.substring(counter, i+1));
counter = i + 1;
```

```
35 foundSpace = true;
36 break;
37 }
38 }
39
40 //check if a space was found in the line
41 if (!foundSpace) {
42 //if no space in whole line just break on letter 17
43 this.linesToPrint.add(messageBuffer.substring(counter, counter+17));
44 counter += 17;
45 }
46 }
47 }
48 return true;
49 }
```

5.3 Der TextField Printer

6 Speech-To-Text-Konvertierung

6.1 Der Android Speech Recognition Client

6.2 Speech-To-Text Konvertierung durch die Google Cloud API

Die Google Streaming Speech Cloud API definiert in ihrer Spezifikation sowohl die Art der Authentifizierung, als auch die verschiedenen Zugriffsmöglichkeiten. Für Java empfiehlt sich eine Authentifizierung über die Oauth2 Methode und die Kommunikation über Google Remote Procedure Calls (grpc).

Der Begriff Remote Procedures kommt aus dem Bereich verteilter Systeme, bei denen ein Programm auf einem Client-Rechner auf einem Server laufende Programmteile ausführen will.
Um dies zu realisieren, verfügt der Client über einen sogenannten method stub. Dieser Stub ist eine Dummymethode mit gleicher Signatur wie die produktive auf dem Server laufende Methode. Die Dummymethode ruft über einen bestehenden Channel zum Server dann die produktive Methode mit den übergebenen Parametern auf und wartet dann auf den Rückgabewert der Servermethode. Hat sie den Wert erhalten, gibt sie Ihn zurück an das aufrufende Unterprogramm. Der eine Remote Procedure ausführende Client kann beziehungsweise muss sich somit nicht selbst um die Client-Server-Kommunikation kümmern, wie es beim REST-Modell der Fall ist, und bekommt im Bestfall nicht einmal mit, dass die aufgerufene Methode extern ausgeführt wurde.

Google bietet für das RPC Konzept eine library ein, welche als Dependency zum Projekt hinzugefügt werden muss. Für die Speech API wird die library com.google.cloud.speech.v1beta1, die RPC Komponenten befinden sich im Package com.google.cloud.speech.v1beta1.SpeechGrpc. Der SpeecGrpc Stub kann über die static Methode SpeechGrpc.newStub(ManagedChannel channel) erzeugt werden. Diese Methode gibt eine Instanz der Klasse SpeechGrpc.SpeechStub zurück.

6.3 Der TextFieldRecorder / Konvertierer

Literatur

[Buc13] Jürgen Buck. *Radiologie Träger des Fortschritts: Festschrift für Friedrich H.W. Heuck*. Google-Books-ID: mjOmBgAAQBAJ. Springer-Verlag, 7. März 2013. 280 S. ISBN: 978-3-642-80128-0.

[KR12] Greg Kipper und Joseph Rampolla. *Augmented Reality: An Emerging Technologies Guide to AR*. Google-Books-ID: OyGiW2OYI8AC. Elsevier, 31. Dez. 2012. 177 S. ISBN: 978-1-59749-734-3.

Anhang

(Beispielhafter Anhang)

A. Assignment

B. List of CD Contents

C. CD

B. Auflistung der Begleitmaterial-Archivdatei

Die Archivdatei wurde zusammen mit der Online-Version dieser Ausarbeitung auf die Lernplattform hochgeladen.

```
⊢ Literature/
|    ⊢ Citavi-Project(incl pdfs)/   ⇒ Citavi (bibliography software) project with
|    |                               almost all found sources relating to this report.
|    |                               The PDFs linked to bibliography items therein
|    |                               are in the sub-directory 'CitaviFiles'
|    |    – bibliography.bib        ⇒ Exported Bibliography file with all sources
|    |    – Studienarbeit.ctv4      ⇒ Citavi Project file
|    |    ⊢ CitaviCovers/           ⇒ Images of bibliography cover pages
|    |    ⊢ CitaviFiles/            ⇒ Cited and most other found PDF resources
|    ⊢ eBooks/
|    ⊢ JournalArticles/
|    ⊢ Standards/
|    ⊢ Websites/
|
⊢ Presentation/
|    –presentation.pptx
|    –presentation.pdf
|
⊢ Report/
     – Aufgabenstellung.pdf
     – Studienarbeit2.pdf
     ⊢ Latex-Files/ ⇒ editable LaTeX files and other included files for this report
          ⊢ ads/                  ⇒ Front- and Backmatter
          ⊢ content/              ⇒ Main part
          ⊢ images/               ⇒ All used images
          ⊢ lang/                 ⇒ Language files for LaTeX template
```