Institut für Visualisierung und Interaktive Systeme
Universität Stuttgart
Pfaffenwaldring 5a
70569 Stuttgart
Germany

Bachelorarbeit Nr. 221

Entwicklung und Evaluation einer Anwendung zur automatischen Speicherung und Verwaltung gelesener Texte auf Desktop PCs

Gerd Matheis

Studiengang:	Softwaretechnik
Prüfer:	Prof. Dr. Niels Henze
Betreuer:	Dipl.-Medieninf. Tilman Dingler
begonnen am:	16. April 2015
beendet am:	16. Oktober 2015
CR-Klassifikation:	H.3.7, H.4.1, I.7.5

Kurzfassung

Die Vielzahl und die Verfügbarkeit von Informationen wächst im Zeitalter des Internets und von Smartphones so schnell, dass es unmöglich erscheint Schritt zu halten. Facebook, Twitter und YouTube sind dabei die prominentesten Plattformen für die Verbreitung von Informationen. So werden wir regelrecht von Informationen überflutet, die wir so schnell gar nicht mehr richtig aufnehmen können.

Das Problem der Informationsflut belastet uns längst nicht mehr nur in unserer Freizeit. Auch am Arbeitsplatz müssen wir immer mehr Informationen aufnehmen und verarbeiten. Oftmals muss dort auch häufig zwischen verschiedenen Tätigkeiten gewechselt werden. Die Unmenge an Information in Kombination mit dem ständigen Wechsel zwischen den Tätigkeiten belastet unser Gehirn sehr. Dies kann sich beispielsweise durch eine schlechte Konzentrationsfähigkeit und eine verkürzte Aufmerksamkeitsspanne äußern. Das Resultat ist, dass wir uns längst nicht mehr alles merken können, was wir über den (Arbeits-) Tag alles geleistet und bearbeitet haben. Zumindest nicht mehr in allen Details.

Wer bestimmte Informationen immer wieder wiederholt kann sie sich langfristig gut einprägen. Dabei muss es nicht zwangsläufig ein Karteikartensystem sein, denn auch technisch unterstützte Systeme zur Retrospektive können dabei helfen, sich Informationen leichter und auch länger zu merken (Isaacs, et al., 2013). Ein System, welches die Informationen, die wir über den Tag am Bildschirm aufnehmen automatisch mitverfolgt und archiviert, kann eine solche Retrospektive ermöglichen und so den Benutzer bei seiner täglichen Arbeit sinnvoll unterstützen, da er alle Informationen zentral aus einem System abrufen kann.

Aus diesem Hintergrund heraus wird in dieser Bachelorarbeit im Rahmen des RECALL-Projektes[1] ergründet, ob und wie sich der Einsatz einer Tracking-Software zur Retrospektive in Kombination mit einem Eye-Tracker im Alltag verwirklichen lässt und wie effektiv diese Methode ist.

Neben der Effektivität der verschiedenen getesteten Erkennungsmethoden zur Bewertung der Wichtigkeit der Inhalte wird sich in der Diskussion unter anderem damit auseinander gesetzt, ob die Probanden einen Nutzen in der Software sehen oder ob die Sicherheitsbedenken aufgrund des starken Eingriffes in die Privatsphäre überwiegen. Außerdem soll ein Überblick darüber gegeben werden, welche Features die Studienteilnehmen für am Wichtigsten halten.

[1] http://recall-fet.eu

Inhalt

1 Einleitung

Im Rahmen des Recall-Projektes2 werden an der Universität Stuttgart in Kooperation mit den Universitäten von Lancaster, Della Svizzera Italiano und Essex Methoden erforscht um das Erinnerungsvermögen sinnvoll zu unterstützen und zu verbessern. Dabei wird ein besonderes Augenmerk auf die jüngsten technischen Entwicklungen gelegt, da diese ein automatisches und beständiges Aufzeichnen verschiedenster Aspekte unseres Alltagslebens ermöglichen. (Recall)

1.1 Herausforderung

Die jüngsten technischen Entwicklungen wie Smartphones oder (soziale) Plattformen im Internet geben uns die Möglichkeit zu jederzeit beliebige Informationen abzurufen und uns in Echtzeit mit anderen Menschen auszutauschen. Das Internet wächst immer schneller und damit auch die Menge der Informationen. Das liegt einerseits an den vielen verfügbaren Plattformen, aber auch daran, dass es immer leichter wird, Webseiten und Blogs zu erstellen und mit Inhalt zu füllen. Ein Beispiel für das Wachstum der verfügbaren Informationen ist die Video-Plattform YouTube. Dort werden pro Minute etwa 300 Stunden Videomaterial hochgeladen (Youtube, 2015). Außerdem tauschen sich Millionen Nutzer über Facebook, Twitter und über eine Vielzahl an verschiedensten Foren über alle möglichen Themen aus und täglich kommen immer mehr Plattformen zu dieser langen Liste hinzu. Folglich verbringen wir auch immer mehr Zeit im Internet.

Um die aktuellsten Geschehnisse mitzubekommen gibt es oft keine Alternative mehr: Termine werden in der Benutzergruppe des nächstbesten Handy-Chats vereinbart oder direkt über Facebook, Twitter und Co veröffentlicht. Wer da nicht dabei ist hat oft keine Chance mitzubekommen, wer sich wann wo trifft. Wer sich dem nicht fügen möchte wird ziemlich bald als Außenseiter dastehen - abgeschnitten von jeglicher Gruppenkommunikation.

Soziale Medien ermöglichen es uns aber auch, Freundschaften über den kompletten Globus hinweg zu pflegen und jeden Tag Leute aus den unterschiedlichsten Ländern übers Internet kennen lernen zu können.

Diese ganzen Möglichkeiten, die uns das Internet, die Smartphones und die ganzen Plattformen bieten haben auch ihre negativen Seiten. In einer Statistik von 2009 (Statista GmbH, 2009) gaben 15 Prozent der Befragten an, täglich 3 bis 5 Stunden Facebook zu nutzen. Dabei geht es teilweise soweit, dass Nutzer regelrechte Angst davor haben, etwas zu verpassen, wenn sie nicht ständig alle Neuigkeiten abrufen können, was im Volksmund auch als „FOMO[3] bezeichnet wird (Levy, 2014).

[2] http://www.recall-fet.eu
[3] Fear of Missing Out ist die Angst etwas zu verpassen. (vgl. https://de.wikipedia.org/wiki/Fomo)

Diese steigende Informationsflut überschwemmt einen jedoch nicht nur während der Freizeit. Laut eines Forschungsprojektes der Bundesanstalt für Arbeitsschutz und Arbeitsmedizin (Moser, Preising, Göritz, & Paul, 2002), die bereits 2002 veröffentlich wurde, steigt auch der Druck auf Arbeitnehmer durch das Internet stetig. Ein Artikel des Focus Online bestätigt diese These und zeigt, dass sich die Lage weiter verschlimmert hat: „[Diese] wachsende Informationsflut und ständige Unterbrechungen machen nicht nur den Arbeitsalltag zu einer Herausforderung für unser Gehirn.“, (FOCUS, 2012). Dabei den Überblick zu behalten ist ziemlich schwer. Die Menge an Informationen überflutet einen förmlich, sodass es nahezu keine Möglichkeit gibt sich alle Informationen zu merken. Die Folgen dieser steigenden Dauerbelastung lassen sich gut erahnen.

Arbeitsministerin Andrea Nahles äußerte sich gegenüber der Rheinischen Post mit der schwer zu widerlegende These: „Es gibt unbestritten einen Zusammenhang zwischen Dauererreichbarkeit und der Zunahme von psychischen Erkrankungen“ (Rheinische Post, 2014).

1.2 Vision

Aufgrund dieser immer weiter steigenden Belastung durch Dauererreichbarkeit und durch die Informationsflut müssen dringend Gegenmaßnahmen ergriffen werden. Hier können Arbeitgeber und Arbeitnehmer selbst dafür sorgen, dass die Belastung reduziert wird. Obwohl neue Technologien für die Informationsflut verantwortlich scheinen, bergen sie aber auch großes Potential, die Belastung zu reduzieren und die Menschen bei ihrer Arbeit zu unterstützen. So stellt sich die Aufgabe dieses Potential auch zu nutzen.

Da die Informationsaufnahme in der Regel durch das Lesen von Texten erfolgt wäre eine Lösung, dass eine Software im Hintergrund die Leseaktivitäten des Tages verfolgt, von den wichtigsten gelesenen Texten *Screenshots*[4] anfertigt und diese in einer Tagesübersicht bereitstellt. Damit kann jederzeit nachvollzogen werden, was wann und wie lange bearbeitet wurde.

Es wurde bereits gezeigt, dass der Einsatz solcher Systeme dem Nutzer dabei helfen kann, sich Informationen besser und langfristiger einprägen zu können (Isaacs, et al., 2013). Wird darüber hinaus reflektiert was an einem Tag erarbeitet wurde, stärkt das nicht nur das Gedächtnis, sondern bietet dem Nutzer auch die positive Rückmeldung, etwas geleistet zu haben. Diese Bestätigung im eigenen Tun triggert das Belohnungszentrum im Gehirn und sorgt damit für ein Glücksgefühl und mehr Wohlgefühl.

[4] Als Screenshot wird ein Bild vom aktuellen Bildschirminhalt oder einem Teil davon bezeichnet. (vgl. https://de.wikipedia.org/wiki/Screenshot)

Mittels *OCR*[5] und anderer Methoden können auch zusätzliche Metadaten aus den Bildern extrahiert werden. So ist es beispielsweise denkbar, den Text eines Internetartikels mittels *OCR* zu extrahieren und direkt mit abzuspeichern. Ebenso kann dadurch die Quelle (*URL*[6] / Dateiname) extrahiert und mit abgespeichert werden, um ein späteres Wiederfinden des Original-Artikels zu ermöglichen. Die Möglichkeit, gelesene Texte wiederzufinden und auf sie zuzugreifen, lädt dazu ein, den Text erneut zu lesen und so zu reflektieren.

Damit diese Datensammlung aber nicht zu einer weiteren Informationsüberflutung führt, muss die Software erkennen können, welche Screenshots überhaupt für den Benutzer relevant sind. Hierbei kann ein Eye-Tracker[7] helfen, der die Augenbewegungen erkennt. Anhand dieser lässt sich einfach erkennen, auf welche Inhalte der Benutzer seine Aufmerksamkeit richtet (Kunze, Andreas, Utsumi, Shiga, & Kise, 2013)[8]. Ebenfalls können die Augenbewegungen Aufschluss über die Relevanz des Inhalts geben (Buschner, Dengel, & van Elst, 2008).

Alle diese Daten können in einer persönlichen Datenbank für den jeweiligen Benutzer archiviert werden. So ist es nicht nur möglich, am Ende des Tages eine Übersicht über die Inhalte des jeweiligen Tages zu erhalten, sondern es kann darüber hinaus auch auf die Informationen der letzten Tage und Wochen zugegriffen werden. Das stellt insbesondere dann einen Vorteil dar, wenn nach einem Internet-Artikel gesucht wird, von dem nur noch ein paar Stichworte bekannt sind.

Da die Menge der verfügbaren Informationen im Internet ständig wächst, kann es schnell vorkommen, dass sich ein Artikel auf herkömmliche Art nicht wieder finden lässt, während sich durch den Einsatz eines solchen Textarchives in Kombination mit einer Volltextsuche der schnell wieder auf den entsprechenden Artikel zugegriffen werden kann(vgl. (Gemmell, Bell, & Lueder, 2006)).

Ähnlich zu den sozialen Plattformen wäre es auch denkbar, dass Benutzer einzelne Einträge mit Kollegen und Freunden teilen. So können sich mehrere Nutzer gegenseitig bei Ihrer Arbeit mit Informationen unterstützen. Auch könnten Diskussionen im Netz durch das Referenzieren und Teilen der Quellen qualitativ aufgewertet werden, da jeder Diskussionsteilnehmer sehen kann, worauf die entsprechenden Argumente basieren.

[5] „Optical Character Recognition“. (vgl. https://de.wikipedia.org/wiki/Texterkennung)
[6] Internetadresse, quasi der Pfad zu einer Webseite
[7] Gerät zur Aufzeichnung von Blickbewegungen. (vgl. Kapitel 2.1.3 Eye-Tracker und Eye-Tracking & https://de.wikipedia.org/wiki/Eye-Tracking)
[8] Vgl. Kapitel 2.1.2 Leseverhalten

1.3 Mögliche Risiken

Ein derartiges System ist prinzipiell nichts anderes als eine vollständige Überwachung des Benutzers, weswegen die Erhebung dieser Daten sehr heikel ist und stark in die Privatsphäre des Benutzers eingreift. So könnten entsprechende Daten auch dazu genutzt werden um die Arbeitsleistung des Benutzers sehr präzise nachzuvollziehen, was im schlechtesten Fall auch zu einer Abmahnung oder Kündigung führen kann.

Würden diese Daten entsprechend ausgewertet, ließe sich mit Leichtigkeit ein detailliertes Profil der entsprechenden Person anfertigen oder Firmengeheimisse ausspähen. Somit wären entsprechende Datenbanken wohl ein begehrtes Ziel für Spionagedienste oder Hacker, die diese Daten dann weiterverkaufen. Schon heute versuchen Firmen möglichst viele Daten über ihre Kunden zu sammeln (Fleschner & Matting, 2013), (Tatje, 2013), um ein Profil von ihnen zu erstellen und so ihr Konsumverhalten einschätzen und auch ein Stück weit manipulieren zu können (Morgenroth, 2014). Deswegen ist es wichtig, dass ein solches Archiv gegen den Zugriff von Dritten durch geeignete Verfahren wie Verschlüsselung möglichst gut geschützt ist.

Doch auch die Urheber der Texte können durch dieses System geschädigt werden. Wenn Nutzer die Einträge teilen können und diese Einträge die kompletten Artikel oder auch nur Teile eines Artikels enthalten, könnten Urheberrechte durch das Teilen verletzt werden. Das Teilen von geschützten Inhalten im kleinen Rahmen lässt sich jetzt schon kaum verhindern. Doch durch eine derartige Software ist es leichter möglich, entsprechende Inhalte einer breiten Masse zur Verfügung zu stellen. Selbst wenn die Software keine explizite Funktion zum Teilen enthält, ist es technisch noch immer möglich Artikel manuell zu teilen. Externe Programmierer könnten auch entsprechende Programme entwickeln, mit denen die Inhalte doch geteilt werden können. Somit ist es technisch nur schwer möglich, das geistige Eigentum Dritter ausreichend zu schützen.

2 Grundlagen und verwandte Arbeiten

2.1 Grundlagen

Im Folgenden werden die nötigen Begriffsdefinitionen gegeben und die wichtigsten Grundlagen erläutert, welche für das weitere Verständnis relevant sind. Außerdem werden die, für die Entwicklung genutzten Technologien vorgestellt und die wichtigsten Fachbegriffe erläutert.

2.1.1 Augenbewegungen

Das menschliche Auge ist in der Regel ständig in Bewegung und erfasst die visuellen Reize, die uns umgeben. Nach der Definition von Keith Rayner (Rayner, 2009) lassen sich Augenbewegungen auf zwei einfache Grundelemente zurückführen. Zum einen auf die tatsächliche Bewegung der Augen, welche als *Sakkade* bezeichnet wird und zum anderen auf die, als *Fixation* bezeichnete Zeit, in der die Augen auf annähernd einer Stelle verweilen. Dabei können wir Informationen in der Regel nur während *Fixationen* aufnehmen, da sich die Augen während einer *Sakkade* meist so schnell bewegen, dass wir nur ein verzerrtes Bild wahrnehmen können. Das heißt allerdings nicht, dass unser Gehirn während einer *Sakkade* untätig ist, denn dieses kann in dieser Zeit die bereits aufgenommenen Informationen weiter verarbeiten.

2.1.2 Leseverhalten

Wird ein Text im Stillen gelesen, gibt es ein ganz bestimmtes und markantes Muster von *Fixationen* und *Sakkaden* in den Augenbewegungen, welches unbewusst und völlig automatisch angewendet wird. Dabei fixieren die Augen den Text für etwa 100 bis 500 Millisekunden um dann eine *Sakkade* von links nach rechts zu vollziehen (Biedert, Buschner, & Dengel, 2009), (Emam & Youssef, 2012). Etwa 180 bis 250 Millisekunden dieser Fixationszeit sind auf die sogenannte *Sakkaden*-Verzögerung zurückzuführen. Dies ist die Zeit, die benötigt wird, um nach dem Entziffern eines Wortes oder Wortabschnittes die nächste *Sakkade* zu planen. Die Länge der *Sakkade* beträgt durchschnittlich sechs bis neun Buchstaben und hängt somit von der Schriftgröße, nicht aber von der Entfernung des Lesers zum Text ab und dauert ungefähr 20 – 60 Millisekunden. Allerdings sind auch ungefähr 10 – 15 % der *Sakkaden* Rücksprünge zu einer bereits gelesenen Textstelle. Das heißt, die Augen bewegen sich entlang der Zeile zurück oder sogar zur vorhergehenden Zeile. Besonders charakteristisch ist jedoch die *Sakkade*, die bei einem Zeilenumbruch vollzogen wird, da diese eine Bewegung von rechts nach links über die gesamte Textbreite darstellt.

Die Lesegeschwindigkeit hängt allerdings noch von anderen Faktoren ab. Wird beispielsweise ein inhaltlich anspruchsvoller Text gelesen, kann es sein, dass mehr Rücksprünge benötigt werden, um den Text inhaltlich erfassen zu können. Auch das Schriftbild spielt eine wesentliche Rolle, da es gegebenenfalls länger braucht, um die einzelnen Buchstaben und Wörter zu entziffern.

2.1.3 Eye-Tracker und Eye-Tracking

Beim Eye-Tracking werden die Bewegungen der Augen durch Kameras aufgenommen. Dabei bestehen die Augenbewegungen prinzipiell aus *Fixationen* und aus *Sakkaden*. Anhand der Bilder, welche die Kameras von den Augen machen, lässt sich ermitteln, wohin die jeweilige Person gesehen hat.

Eye-Tracking kann mit unterschiedlichen Geräten durchgeführt werden. Es gibt unter anderem Bildschirme mit integrierten Eye-Trackern oder spezielle Brillen, welche die Augenbewegungen aufzeichnen können („head mounted eye tracker“).

Anhand dieser Daten wird nachvollzogen, wohin eine Person sieht und wie lange der Blick dort verweilt. So lassen sich unter anderem Einblicke darin erlangen, wie ein Nutzer Informationen aufnimmt oder ob ein bestimmtes Layout (eines Programmes oder einer Webseite) die Aufmerksamkeit des Nutzers besser lenken kann als andere.

Zusätzlich lassen sich mit diesen Daten bestimmte Benutzeraktionen erkennen. Zum Beispiel, wann der Benutzer liest (vgl. Kapitel 2.1.2, Seite 12) oder ob der Benutzer am Rechner sitzt. Darum haben sich Eye-Tracker in den vergangenen Jahren für viele Studien im Bereich der Visualisierung zu einem elementaren Werkzeug für die Erforschung neuentwickelter Visualisierungen und Interaktionen entwickelt.

Mit Hilfe der Informationen, die ein Eye-Tracker bereitstellt, können aber auch Programme gezielter auf das Nutzerverhalten reagieren. Es ist so beispielsweise, störende Meldungen und Popups zu unterbinden, solange erkannt wird, dass der Nutzer konzentriert arbeitet (einen Text liest, programmiert etc.), damit dieser in seinem Arbeitsfluss nicht unterbrochen wird.

Im Rahmen dieser Bachelorarbeit werden portable Eye-Tracker der Firma Tobii Technology AB[9] genutzt. Insgesamt stehen vier EyeX-Eye-Tracker zur Verfügung, wobei zwei aus der EyeX-C-Serie und zwei aus der EyeX-2-Serie stammen. Diese Eye-Tracker werden mittels USB 3.0 mit dem Rechner verbunden. Über das Tobii-EyeX-Framework können die Punkte, welche die Augen auf dem Bildschirm fixieren, abgerufen werden. Außerdem werden durch das Framework einige Funktionen zur Kalibrierung und für die Einstellungen des Eye-Trackers zur Verfügung gestellt.

[9] http://www.tobii.com

2.1.4 Lese-Erkennung

Bei der Leseerkennung („Reading-Detection“) geht es darum, anhand der Punkte, die von einem Eye-Tracker übermittelt werden, zu erkennen, ob der Benutzer beispielsweise liest. Diese Erkennung ist theoretisch ohne technische Hilfsmittel möglich, sofern die Augenbewegungen beim Lesen genau beobachtet werden (vgl. Kapitel 2.1.2, Seite 12). Beim Lesen lässt sich ein Z-Muster erkennen, da unsere Augen von links nach rechts der Zeile folgen und am Ende der Zeile eine schnelle Bewegung (*Sakkade*) zur nächsten Zeile machen. Komplexer wird die Erkennung dadurch, dass beim Lesen auch immer wieder Rücksprünge ausgeführt werden, um bereits gelesene Zeilen erneut zu lesen, weil beispielsweise der letzten Satz inhaltlich nicht verstanden wurde. Ein weiteres Problem besteht darin, dass bei der Lese-Erkennung zu Beginn unklar ist, ob der Benutzer liest oder seine Augenbewegung in diesem Moment zufällig einer Lesebewegung ähnelt.

Aus diesem Grund benötigt jeder Algorithmus zur Lese-Erkennung immer etwas Zeit, bis er mit einigermaßen hoher Sicherheit davon ausgehen kann, dass der Benutzer im Moment wirklich liest oder nicht.

Abbildung 2.1: Tobii EyeX angebracht an einem externen Monitor (Tobii Technology AB, 2014)

2.1.5 C# und das .NET-Framework

Die Sprache C# (gesprochen: „c sharp“) wurde von Microsoft im Rahmen des *.NET-Projektes* (gesprochen: „dot net“) zusammen mit dem *.NET-Framework* entwickelt und erstmals im Jahre 2002 veröffentlicht.

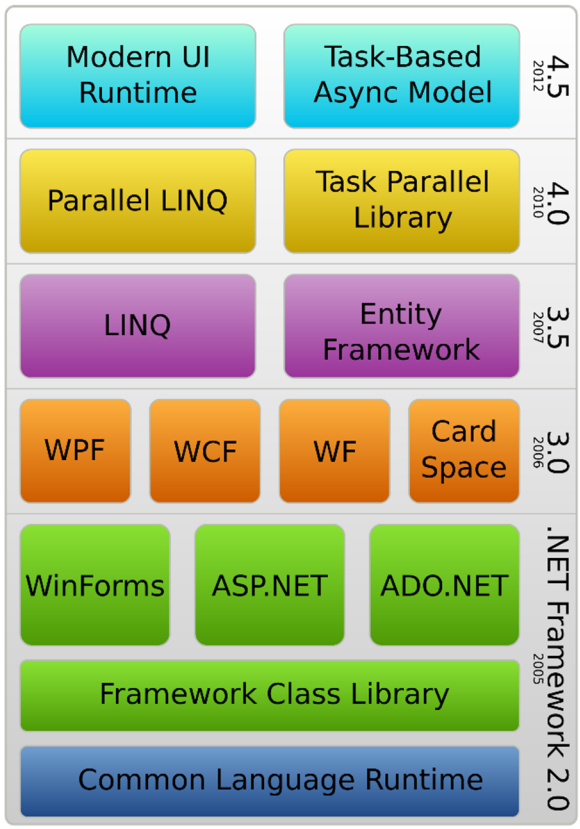


Abbildung 2.2: Die .NET-Framework-Hierarchie (Soumyasch, 2007)

C# ist eine objektorientierte Sprache, deren Syntax sich an der von C/C++ bzw. Java orientiert. Das besondere an C# im Vergleich zu C/C++ ist, dass der Code vollständig *managed* ist und die in C++ als unsicher geltenden Konzepte in C# nur in als „unsicheren Code“ gekennzeichneten Code-Segmenten erlaubt sind.

Managed Code bedeutet in diesem Fall, dass der Code durch eine Runtime ausgeführt wird, die sich unter anderem um Typsicherheit und um *Garbage Collection* kümmert, sodass nur noch in seltenen Fällen auf unsicheren Code[10] zurückgegriffen werden muss (Microsoft, What Is Managed Code?, 2015). Da sich manche Aufgaben jedoch nicht anders lösen lassen, muss die Runtime intern dennoch auf unsichere Konzepte zurückgreifen. Hierbei wird allerdings darauf geachtet, dass diese Code-Segmente fehlerfrei implementiert sind.

Bei C# stellt diese Runtime das .NET-Framework dar, welches auch als eine strukturierte Sammlung von allgemeinen Funktionen betrachtet werden kann.

Januar 2008 stellte Microsoft bereits Teile des Quelltextes des .NET-Frameworks öffentlich unter der „Microsoft Reference License“ zur Verfügung (Schwichtenberg, Quellcode einiger .NET-Bibliotheken jetzt verfügbar, 2008). Am 12. November 2014 wurde eine Teilmenge des Reference Source Quelltextes unter der MIT-Lizenz auf GitHub veröffentlicht. (Schwichtenberg, Microsoft: .NET wird komplett Open Source, 2014)

[10] Siehe https://msdn.microsoft.com/de-de/library/t2yzs44b.aspx

Alle .NET-Sprachen, zu denen auch C# gehört, werden zwar kompiliert, jedoch nicht direkt in *Maschinensprache*[11] sondern in „Common Intermediate Language“-Code übersetzt, wobei die *CIL*[12] aus der *MSIL*[13] hervorgeht. Diesen *IL*[14]-Code übersetzt zur Laufzeit ein *JIT*[15]-Compiler in Maschinensprache. (Microsoft, MSDN - Kompilieren von Anwendungen mit .NET Native, 2015)

C# liegt derzeit (September 2015) in der Version 6.0 vor und das .NET-Framework in Version 4.6.

2.1.6 MVVM-Pattern

Das MVVM-Pattern wurde am 8. Oktober 2005 von John Gossman, einem Microsoft MVP[16], veröffentlicht (Gossman, 2005). Es stellt dabei eine spezielle Form des Presentation Model Patterns dar, welches am 19. Juli 2004 von Martin Fowler veröffentlicht wurde (Fowler, 2004).

Dabei zielte das MVVM-Pattern ursprünglich auf das Erstellen von Anwendungen mittels *WPF* (vgl. Kapitel 2.1.8, Seite 18) ab, findet aber mittlerweile auch bei anderen Projekten Verwendung, da sich mit dem MVVM-Pattern Anwendungen gut gliedern lassen.

Das MVVM-Pattern unterteilt sich in drei Komponenten, aus denen sich der Name des Patterns ableitet:

- *Model*
- *View*
- *ViewModel*

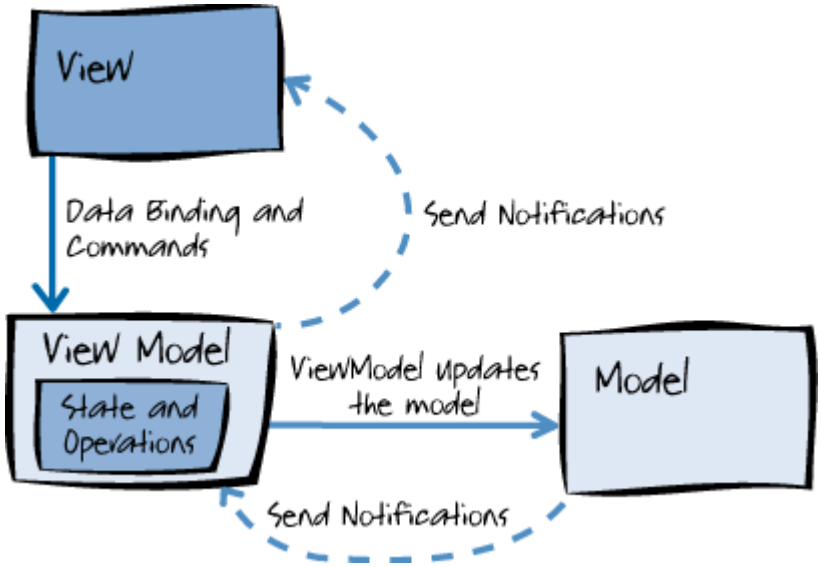


Abbildung 2.3: Das MVVM-Pattern. (Microsoft, MSDN - Das MVVM Pattern, 2012)

[11] Maschinensprache ist Code in einem, für den Computer ausführbaren Format. (vgl. https://de.wikipedia.org/wiki/Maschinensprache)
[12] Common Intermediate Language. (vgl. https://de.wikipedia.org/wiki/Common_Intermediate_Language)
[13] Microsoft Intermediate Language. (vgl. https://de.wikipedia.org/wiki/Common_Intermediate_Language)
[14] Intermediate Language ist eine Zwischensprache für Code, der aus portabilitätsgründen nur vorkompiliert wurde und noch nicht in Maschinensprache vorliegt.
[15] JIT – „Just in Time“
[16] Microsoft Most Valuable Professional ist die höchste Auszeichnung von Microsoft für engagierte Experten, die herausragende Artikel über Microsoft-Technologien veröffentlicht haben. (vgl. https://de.wikipedia.org/wiki/Microsoft_MVP)

Dabei setzt das MVVM-Pattern auf größtmögliche Trennung der grafischen Oberfläche (*View*) und der Anwendungs-Logik. Dies hat den Vorteil, dass Anwendungen theoretisch komplett unabhängig von ihrer *View* sind und die *View* deswegen mit vertretbarem Aufwand ausgetauscht werden kann. Dadurch wird beispielsweise ermöglicht, eine Windows-Anwendung so abzuändern, dass sie danach als Web-Anwendung zur Verfügung steht, ohne dabei die Programmlogik anpassen zu müssen. Die Aufgabe der *View* ist es, wie in Abbildung 2.3 zeigt, die anzuzeigenden Daten aus zugehörigen *ViewModels* zu holen. Dabei sorgt *DataBinding* dafür, dass die Daten der *View* mit den Daten des *ViewModels* synchronisiert werden. Zudem kann die View *Commands* auslösen, welche durch das *ViewModel* zur Verfügung gestellt werden. Diese *Commands* lösen dann weitere Programmabläufe aus. So kann ein *Command* beispielsweise durch einen Klick auf einen Button ausgelöst werden.

Das *ViewModel* hingegen hat die Aufgabe einer *View* die notwendigen Daten zur Verfügung zu stellen. Zudem stellt das *ViewModel* der *View* sogenannte *Commands* bereit, die dafür sorgen, dass weitere Programmlogik ausgeführt werden kann. Ändern sich Werte im *ViewModel*, so benachrichtigt das *ViewModel* die *View* mit Hilfe sogenannter *Notifications* darüber, dass sich Werte geändert haben, damit sich die View wieder mit dem *ViewModel* synchronisiert. Auf gleicher Ebene wie die *ViewModels* stehen *BusinessServices* zur Verfügung (die nicht in der Abbildung aufgeführt sind). Diese *BusinessServices* beinhalten die Anwendungs-Logik. Die *Models* stellen nur das Datenmodell für die entsprechende Anwendung zur Verfügung und enthalten selbst keine Anwendungslogik.

Soll nun eine Datei geöffnet, eingelesen und der Inhalt der Datei in einem Textfeld ausgegeben werden, ist der Ablauf folgender:

Zunächst löst der Nutzer über einen Klick auf einen Button in der *View* den Befehl, die Datei zu laden aus. Dieser Befehl wird weitergegeben an das zugehörige *ViewModel*, welches den Code enthält, der den richtigen *BusinessService* ansteuert. Über einen Funktionsaufruf vom ViewModel an den *BusinessService* wird der Code im *BusinessService* ausgeführt, der dafür sorgt, dass die Datei eingelesen wird. Als Rückgabeparameter der Funktion übergibt der *BusinessService* dem *ViewModel* den Dateiinhalt. Das *ViewModel* speichert den Inhalt der Datei in einem Feld und benachrichtigt die *View* darüber, dass sich das entsprechende Feld geändert hat. Das *DataBinding* der *View* sorgt dafür, dass die Daten des Feldes geladen und in der *View* angezeigt werden.

2.1.7 Projekt

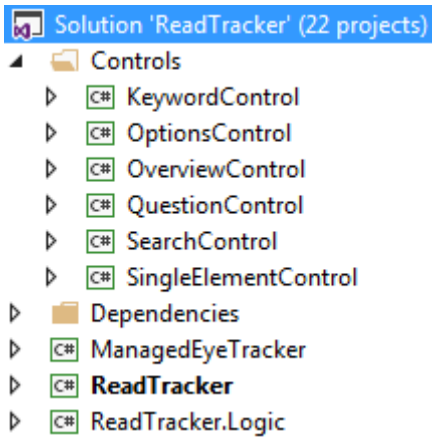


Abbildung 2.4: Projekte des Prototyps

In *VisualStudio*[17] gibt es die Möglichkeit eine Software in verschiedene Projekte zu untergliedern (vgl. Abbildung 2.4). Jedes Projekt wird in eine eigene DLL kompiliert, wobei die Hauptanwendung (hier das Projekt „ReadTracker“) in eine EXE-Datei kompiliert wird. Die Projekte sind zunächst voneinander unabhängig, können aber aufeinander referenzieren, wobei Ringverweise unzulässig sind. Der Vorteil ist, dass verschiedene Programmteile von einander getrennt werden können. Insbesondere bei eigenen *Controls* ist dies von Vorteil: Wird jedes *Control* in ein eigenes Projekt gepackt, sind diese voneinander unabhängig kompilierbar. Tritt beim Kompilieren eines *Controls* ein Fehler auf, funktionieren zumindest die anderen *Controls* noch, was dafür sorgt, dass die Vorschau der Anwendung im Hauptprojekt für diese noch funktioniert.

Die Aufteilung der verschiedenen Programmteile sorgt ebenso für zusätzliche Übersicht. So hilft die Trennung außerdem dabei, Programmierfehler zu verhindern: Werden *View* und *ViewModel* in verschiedenen Projekten realisieret, ist sichergestellt, dass die beiden Ebenen voneinander getrennt sind und nicht vermischt werden können.

2.1.8 WPF

Die Windows Presentation Foundation (kurz: *WPF*, vgl. Abbildung A.1) ist ein Framework für grafische Benutzeroberflächen (kurz: *GUI*)[18], welches von Microsoft entwickelt und 2006 erstmals veröffentlicht wurde (Nathan, 2010). Die *WPF* baut dabei auf *DirectX* auf und ist somit bestens gerüstet für den Einsatz von visuellen Effekten und zur Wiedergabe von Medien wie Sound und Videos. Das Herzstück der *WPF* ist eine auflösungsunabhängige vektorbasierende *rendering engine*[19], die so konstruiert ist, dass sich die Vorteile moderner [Grafikkarten] nutzen lassen (frei übersetzt nach: (Microsoft, MSDN - Introduction to WPF, 2015)). Die grafischen Oberflächen der *WPF* werden dabei durch *XAML*[20] beschrieben, welche sich am XML-Standard orientiert. Aus dieser *XAML*-Beschreibung der Oberfläche wird zur Laufzeit ein Baum von Elementen, der *Visual Tree* aufgebaut, auf dem die *WPF* die entsprechenden Operationen durchführt.

[17] Entwicklungsumgebung von Microsoft für .NET-Sprachen.
[18] GUI – „graphical user interface“ ist eine grafische Oberfläche zur Bedienung einer Software. (vgl. https://de.wikipedia.org/wiki/Grafische_Benutzeroberfläche)
[19] Render Engine: Programmteil, der berechnet, welche Elemente wie angezeigt werden.
[20] e**X**tensible **A**pplication **M**arkup **L**anguage

Eines der wichtigsten Konzepte der *WPF* ist das *DataBinding*. Dabei handelt es sich um eine Definition in der *XAML*, die angibt, wie die Daten aus dem *ViewModel*-Layer auf der Oberfläche dargestellt werden sollen. Durch das *DataBinding* sorgt die *WPF* dafür, dass die Daten in der *View* und die Daten im zugehörigen *ViewModel* synchronisiert werden. Dabei werden sowohl bidirektionale als auch beide unidirektionale Bindungen unterstützt. Im vorigen *GUI*-Framework *WinForms* von Microsoft musste der Programmierer teilweise selbst für diese Synchronisation sorgen.

Weiterhin unterstützt die *WPF* 3D-Grafiken und Animationen, welche bei *WinForms* ebenfalls manuell und ggf. mit Elementen von Drittanbietern eingefügt werden müssen.

Zeitgleich mit der *WPF* wurde von Microsoft das MVVM-Pattern (vgl. Kapitel 2.1.6, Seite 16) eingeführt, welches eng mit der *WPF* verzahnt ist. Darum werden *WPF*-Anwendungen in der Regel nach dem MVVM-Pattern entwickelt.

Ziel ist es, durch den Einsatz der *WPF* und des MVVM-Patterns die grafische Oberfläche so weit von der Anwendungslogik zu trennen, dass beide Teile unabhängig voneinander abgeändert werden können. Dies bringt vor allem den Vorteil, dass Programmdesigner unabhängig von den Programmentwicklern an der Software arbeiten können. Diese Trennung durch die *WPF* gibt vor allem deshalb Sinn, weil sich erstmals alle Elemente auf einfache Art und Weise durch Templates und Styles komplett umgestalten und zu einem einheitlichen Anwendungsdesign umarbeiten lassen.

2.2 Verwandte Arbeiten

Bisher gibt es wenige Systeme, die mittels Eye-Tracking die, für den Benutzer wichtigen Texte und Dokumente erkennen und diese archivieren können. Deutlich mehr Projekte und Forschungsarbeiten beschäftigen sich allgemeiner mit dem Einsatz von Eye-Trackern im alltäglichen Leben. Im Folgenden soll ein kurzer Überblick über die verwandten Arbeiten gegeben werden.

2.2.1 Leseerkennung auf Android-Smartphones

Victor Riempp beschäftigt sich in seiner Diplomarbeit (Riempp, 2015) mit einem ähnlichen System für Android-Geräte. Er hat dabei eine Android-App entwickelt, die er als *Reading Tracker* bezeichnet.

Der *Reading Tracker* sei auf allen Android-Geräten mit der Version 4.4 „KitKat“ (Wikipedia, 2015) lauffähig. Dabei greift der *Reading Tracker* über einen sogenannten *Accessibility Service* auf die Textinhalte zu, um diese an den zugehörigen Web-Dienst *Reading Archive* zu senden. Das Problem an dieser Lösung ist allerdings, dass der komplette Text abgegriffen wird, der im Moment des Abgreifens im aktiven Fenster verfügbar ist. Dieses Problem lässt sich auch nicht ohne weiteres beheben, da sich mit dieser Vorgehensweise nur schwer herausfinden lässt, welche Textpassagen für den Anwender wichtig sind. Ein weiteres Problem ist, dass nur Texte aus Fenstern abgegriffen werden können, welche jenen *Accessibility Service* unterstützen. Dieser ist bisher noch in der Minderheit der Programme verfügbar, weswegen das *Reading Archive* nur unvollständig geführt und das damit verbundene Potential nicht ausgeschöpft werden kann.

Das *Reading Archive* besteht aus einer *MySQL*[21]-Datenbank sowie einer Webseite, die als grafische Oberfläche für den Benutzer fungiert. Über diese Webseite kann der Benutzer, nach einem erfolgreichen Anmeldevorgang, auf alle gesammelten Texte zugreifen. Zudem werden dem Nutzer einige Statistiken über sein persönliches Leseverhalten gezeigt wie beispielsweise die Anzahl gelesener Wörter innerhalb der letzten 24 Stunden inklusive einer Angabe, wie vielen DIN-A4-Seiten dies durchschnittlich entspricht.

Zuletzt zeigt Victor Riempp, dass sich anhand der Daten verschiedene Visualisierungen erstellen lassen, die dem Benutzer interessante Einblicke in sein Leseverhalten erlauben. Als Beispiel ist eine *Word Cloud*[22] in die Webseite integriert, die dem Benutzer die meistgelesenen Wörter der letzten 24 Stunden visualisiert.

[21] https://www.mysql.de
[22] Vgl. https://de.wikipedia.org/wiki/Schlagwortwolke

2.2.2 Text 2.0

Text 2.0 ist ein Framework zur Entwicklung von Webanwendungen, die durch den Einsatz eines Eye-Trackers um verschiedene Funktionen bereichert werden. Als ein möglicher Anwendungsfall wird dabei der sogenannte *Augmented Text* genannt, bei welchem mittels Eye-Tracking ermittelt wird, an welcher Stelle im Text sich der Leser aktuell befindet. Der Text kann dann bei Erreichen eines bestimmten Abschnittes um Bilder, Animationen oder um Tonausgaben bereichert werden.

Mit Hilfe des *Augmented Reading* können unklare Wörter mittels einer Einblendung erklärt, in andere Sprachen übersetzt oder um weiterführende Informationen ergänzt werden.

Das Ziel der Entwickler von Text 2.0 ist es, zu zeigen, wie sich das Lesen von Text auf digitalen Geräten weiterentwickeln kann und welche Möglichkeiten sich durch Eye-Tracking eröffnen. Dabei zeigen Geräte wie das *eyeBook* oder das Browser-Plugin Text 2.0, dass *Augmented Text* bereits heute für die meisten Nutzer dieser Technologien gut funktioniert.

Desweiteren wird aufgezeigt, dass es noch ein langer Weg ist, bis sich *Augmented Reading* und *Augmented Text* endgültig etablieren können. So reiche die Genauigkeit der aktuell verfügbaren *Augmented Reading*-Algorithmen nicht aus, um ein zuverlässiges *Augmented Reading* zu ermöglichen. (Biedert, Buschner, & Dengel, 2009)

2.2.3 Echoes From the Past

In dem Artikel zur ACM SIGCHI Conference 2013 (Isaacs, et al., 2013) zeigen die Autoren anhand einer Nutzerstudie einer Handy-App über den Zeitraum von einen Monat, dass sich durch technologiegestützte Reflektion messbare Verbesserungen in Bezug auf das Wohlbefinden erzielen lassen.

Für die Studie haben die Forscher die Android-Applikation „Echo“ entwickelt, welche die täglichen Aktivitäten aufzeichnet und dadurch eine Reflektion der Aktivitäten ermöglicht. Dabei legt der Benutzer über den Tag Einträge an, welche die Ereignisse seines Tages wiederspiegeln. Beim Erstellen eines Eintrages soll der Nutzer einen kurzen Betreff eingeben und sein aktuelles Wohlbefinden anhand einer Skala von Eins bis Neun bewerten. Zudem hat er die Möglichkeit dem Eintrag eine weitergehende Beschreibung, Fotos, Videos oder eine Audio-Aufnahme anzufügen.

Dabei lädt Echo den Nutzer aktiv dazu ein, sich frühere Ereignisse erneut ins Gedächtnis zu rufen: Jeden Tag werden bis zu drei Ereignisse aus verschiedenen Zeitabschnitten angezeigt, wobei ältere Einträge häufiger als jüngere Einträge angezeigt werden. Für jeden Eintrag werden dem Nutzer der ursprüngliche Eintrag sowie alle zugehörigen Wiederholungen angezeigt. Danach soll der Nutzer sein Wohlbefinden über das Event erneut bewerten und kann weitere Zusatzdaten anfügen. Es ist ebenfalls möglich, dass Benutzer ein beliebiges Event auswählen und reflektieren können.

Das Ergebnis der Studie ist, dass die Nutzer, die ihrer Erlebnisse aufgezeichnet und reflektiert haben ihr Wohlbefinden steigern konnten, da sie negative Ereignisse schneller verarbeiten und positive Lehren daraus ziehen konnten. Selbst jene Nutzer, welche in der Studie nur ihre Ereignisse aufgezeichnet, diese aber nicht wiederholt haben, haben dadurch profitiert, dass sie ihre positiven Emotionen beim Festhalten noch einmal genießen und ihre negativen Gefühle analysieren und verarbeiten konnten statt diese zu unterdrücken.

2.2.4 MyLifeBits

MyLifeBits ist ein Forschungsprojekt von Mircosoft Research, bei dem es darum geht, alles in einem persönlichen digitalen Archiv aufzuzeichnen, was ein Mensch in seinem Leben sieht, hört, sagt oder schreibt (Bell & Gemmell, 2007). Die Idee hinter einem solchen System stammt von Vannevar Bush, der 1945 seine Idee des „Memex“ in seinem Artikel „As we may think“ vorgestellt hat. Diese Idee ist die Grundlage für das Projekt MyLifeBits (Gemmell, Bell, & Lueder, 2006).

Die Versuchsperson dieses Forschungsprojektes, Gordon Bell, sammelt dabei seit Beginn des Projektes im Jahre 2001 alle möglichen persönlichen Daten in diesem digitalen Archiv: Er zeichnet alles was er sieht oder hört sowie seine Interaktionen mit Maschinen und seine Kommunikation mit anderen Menschen darin auf. Außerdem speichert er alle Internetseiten, die er besucht, alle seine E-Mails, Bücher, Videos und sonstigen Daten, die aufgezeichnet werden können. In Zukunft ist es denkbar Informationen aufzuzeichnen, die nicht bewusst wahrgenommen werden können: Mit tragbaren Sensoren ließen sich der Sauerstoffgehalt des Blutes und die CO_2-Konzentration der Atemluft aufzeichnen.

Das Archiv kann dabei unter anderem mit einer Volltextsuche durchsucht werden. Außerdem lassen sich gespeicherte Information um weitere Informationen in Form von Annotationen ergänzen.

3 Konzept

Wie bereits bei dem Projekt „Echo“ (Isaacs, et al., 2013) gezeigt wurde, können technisch gestützte Systeme dazu beitragen, das Wohlbefinden der Anwender zu steigern. Zudem sorgt die wiederholte Reflektion von Information dafür, dass diese längerfristig im Gedächtnis verankert werden.

Aus diesem Grund soll ein Prototyp für ein System zur Archivierung von gelesenen Texten entstehen, welches die wichtigsten Informationen des Tages automatisch abgreift und archiviert. Außerdem soll dem Anwender eine Retrospektive des jeweiligen Tages angezeigt werden, sodass dieser seinen Tagesablauf noch einmal nachvollziehen kann.

Da der Prototyp auf herkömmlichen Laptops und Desktop-PCs laufen soll und Windows das verbreitetste Betriebssystem ist wird der Prototyp für Windows-Systeme entwickelt. C# basiert auf dem .NET-Framework, welches bereits viel Funktionalität mitbringt und eignet sich somit gut für die Entwicklung von Windows-Anwendungen. Für die Oberfläche der Anwendung wird die *WPF* (vgl. Kapitel 2.1.8, Seite 18) eingesetzt, da diese Technologie vergleichsweise neu ist und somit voraussichtlich noch einige Zeit aktuell ist. Diese Entscheidungen sollen dazu beitragen, dass der Prototyp weiter verwendet werden kann, um ein vollwertiges System zur Speicherung und Verwaltung von Texten zu entwickeln. Durch die grundlegende Entscheidung, die Oberfläche mit der *WPF* zu realisieren, wird gleichzeitig festgelegt, dass die Anwendungsarchitektur auf dem MVVM-Pattern (vgl. Kapitel 2.1.6, Seite 16) basiert.

Für die endgültige Anwendung empfiehlt es sich dringend, die Daten in einer Datenbank zu organisieren. Für den Prototypen, der auf den Laptops und PCs von zwölf Probanden laufen soll, wird jedoch von der Integration einer Datenbank abgesehen. Dies hat mehrere Gründe: Zum einen bedeutet die Anbindung an eine Datenbank einen spürbaren Mehraufwand in der Entwicklung. Zum anderen müssten die Probanden weitere Software installieren. Für die Studie soll der Prototyp jedoch so schlank wie möglich sein, sodass die Probanden nicht durch die Installation von zusätzlicher Software (z.B. einer *SQL*-Datenbank) belastet werden.

3.1 Entwicklung der Textverwaltungs-Software

Das Wichtigste ist der Algorithmus zur Lese-Erkennung, da die Studie auf diesem aufbaut und dieser zuverlässig arbeiten muss. Der Algorithmus kann jedoch nur getestet werden, wenn die Anwendung Daten vom entsprechenden Eye-Tracker erhält. Darum muss als erstes die Anbindung an den entsprechenden Eye-Tracker erfolgen und eine Testanwendung entwickelt werden, um den Algorithmus testen und optimieren zu können.

Anschließend wird die Anwendung iterativ ausgebaut und um diverse Funktionen erweitert. Zunächst wird eine exemplarische Oberfläche entstehen, welche gleichzeitig die Möglichkeit bietet um Funktionen erweitert zu werden. Die Anwendung soll den Benutzer so wenig wie möglich von seinem Tagesgeschäft ablenken. Darum ist vorgesehen, das Hauptfenster beim Start der Anwendung nicht direkt anzuzeigen, sondern dieses dem Anwender lediglich über ein Icon in der *Taskleiste*[23] zur Verfügung zu stellen.

Danach werden die Tagesübersicht sowie der Fragebogen entwickelt, da diese Elemente für die Nutzerstudie benötigt werden. Alle weiteren Features haben eine nachrangige Priorität und werden nur entwickelt, wenn genügend Zeit dafür verbleibt.

3.2 Benutzerstudie / Evaluation

Sobald die Entwicklung des Prototyps abgeschlossen ist, wird die Studie gestartet. Dabei wird zunächst eine Pilotstudie mit drei Probanden durchgeführt um den Ablauf der Studie zu erproben und um letzte Fehler aus dem Ablauf oder in der Anwendung zu beheben. Im Anschluss daran wird die Benutzerstudie mit zwölf Teilnehmern durchgeführt.

Die Benutzerstudie besteht aus zwei Phasen. In der ersten Phase soll in einer kurzen Laborstudie die Software installiert und erklärt. Anschließend werden einige kurze Tests durchgeführt, um die Erkennungsrate des Algorithmus zur Lese-Erkennung zu ermitteln. In den folgenden Tagen führt der Proband die Feldstudie durch, bei der die verschiedenen Programm-Modi (Bedingungen, vgl. Kapitel 5.2, Seite 36) getestet werden. Diese Benutzerstudie wird mit einer Retrospektive der Feldstudie und einem Interview des Probanden abgeschlossen. Da für die Feldstudie drei Bedingungen zu testen sind, und diese in unterschiedlicher Reihung von allen Probanden durchgeführt werden sollen, müssen an der Studie zwölf Probanden teilnehmen, um jede Reihenfolge zumindest zweimal durchzuführen.

Im Anschluss an die Studie werden die Daten ausgewertet und in der Bachelor-Arbeit festgehalten. Zuletzt werden die Daten in einer Diskussion interpretiert.

[23] Taskleiste: Liste der aktiven Programme, die größtenteils im Hintergrund laufen. vgl. https://de.wikipedia.org/wiki/Taskleiste

4 Implementierung

Die Anwendung wird planmäßig nach dem MVVM-Pattern implementiert. Dabei wird das MVVM-Pattern abgewandelt, da das Pattern manche Funktionen nicht ermöglicht.

Ein modaler Dialog, wie beispielsweise der Dialog zum Speichern einer Datei, aus der Programmlogik heraus, kann mit dem nativen MVVM-Pattern nicht geöffnet werden, ohne einen Austausch der *View*-Ebene erheblich zu erschweren.

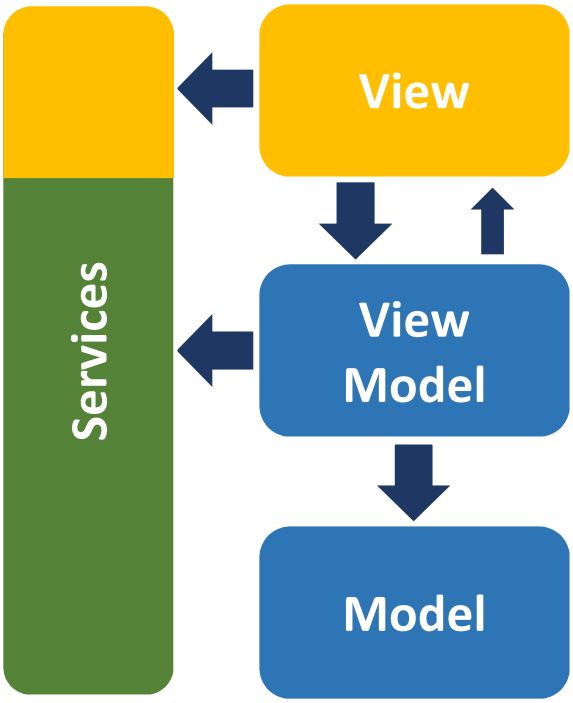


Abbildung 4.1: Abwandlung des MVVM-Patterns

Da diese Austauschbarkeit eines der wichtigsten Punkte des MVVM-Patterns ist, wurde es, wie in Abbildung 4.1 zu sehen ist, um sogenannte *Services* erweitert. Diese *Services* stellen dem *ViewModel* die benötigten Funktionen zur Verfügung welche auf der *View*-Ebene ausgeführt werden. Hierzu ist es notwendig, den *Services* mitzuteilen, mit welcher *View* die entsprechenden Funktionen ausgeführt werden. Darum müssen die *Views* beim Start der Anwendung bei den *Services* registriert werden. Aus diesem Grund ist in Abbildung 4.1 ein Teil der *Services* in der Farbe der *View*-Ebene dargestellt: Dies symbolisiert die *View*-Elemente, die für die Funktionen benötigt werden und bei einer Änderung der *View*-Ebene ebenfalls angepasst werden müssen. Die Austauschbarkeit der *View*-Ebene ist dadurch zwar komplexer geworden, bleibt aber grundsätzlich erhalten. Die *Services* sind im *ServiceManager*, einer *Singleton*-Klasse zusammengefasst (vgl. Kapitel 4.1, Seite 26).

Im Normalfall wird beim Start eines Programms das Hauptfenster angezeigt. In diesem Fall wird nur ein Icon der *Taskeiste* hinzugefügt, damit der Nutzer nicht bei seiner Arbeit gestört wird. Dies macht es erforderlich, dass für die Anwendung zusätzlich ein *Bootstrapper* geschrieben wird, der den Programmablauf steuert.

4.1 ServiceManager

Die Funktionen, die den *ViewModels* zur Verfügung stehen sollen werden auf *ViewModel*-Ebene durch Interfaces definiert. Der *ServiceManager*, eine *Singleton*-Implementierung, sorgt dafür, dass pro *Interface* nur eine *View* registriert wird. Diese muss zudem das entsprechende *Interface* implementieren. Sind diese Kriterien erfüllt wird in einem internen *Dictionary* gespeichert, welches *Interface* mit welcher *View* verknüpft ist. Benötigt ein *ViewModel* Zugriff auf eine der Funktionen, so holt es sich die Klasse, die das entsprechende Interface implementiert, beim *ServiceManager* und ruft die benötigte Funktion auf. Diese wird auf der *View*-Ebene ausgeführt und die Ergebnisse werden an das *ViewModel* zurückgegeben.

4.2 Genutzte Frameworks

Im Folgenden sind die Frameworks aufgeführt, welche in der Anwendung genutzt werden.

4.2.1 log4net

Die Apache log4net Bibliothek ist ein Werkzeug, welches den Programmierer dabei unterstützen soll, Log-Nachrichten in verschiedensten Ausgabeformaten auszugeben (frei übersetzt nach (Apache Software Foundation, 2015)). Die Bibliothek ist lizenziert unter der Apache Software License Version 2.0[24] und stellt verschiedene Funktionen für das Logging zur Verfügung. Log4net wird unter anderem vom *StudyLogger* (vgl. Kapitel 4.3.3.5, Seite 32) und von der Logger-Infrastruktur des GM.Frameworks (vgl. Kapitel 4.2.3, Seite 27) genutzt.

4.2.2 Hardcodet.NotofyIcon.Wpf

Da die WPF standardmäßig keine *TrayIcon*-Anwendungen unterstützt, wird hierfür auf das Framework Hardcodet.NotifyIcon.Wpf von Philipp Sumi in der Version 1.0.5 zurückgegriffen, welches diese Lücke füllt. Das Framework ist unter der Code Project Open License[25] lizenziert und als NuGet Package verfügbar. Weitere Informationen und Programmierbeispiele finden sich auf der Projekt-Webseite unter http://www.hardcodet.net/wpf-notifyicon (Stand: 05.10.2015)

[24] http://www.apache.org/licenses/LICENSE-2.0
[25] http://www.codeproject.com/info/cpol10.aspx

4.2.3 GM.Framework

Das GM.Framework ist ein eigenes Framework, welches außerhalb der Bachelorarbeit entwickelt wurde. Es enthält viele verschiedene Funktionen, die häufig wiederkehrende Aufgaben eines Programmierers vereinfacht.

Der Prototyp nutzt aus diesem Framework einen eigenen Serialiserer, der mehr Möglichkeiten bietet als der Standard-XML-Serialisierer des .NET-Frameworks. Zudem nutzt er die Logger-Infrastruktur, die auf log4net basiert. Weiterhin werden einige *WPF*-spezifische Funktionen genutzt wie beispielsweise der *ServicesManager* oder das *Wizard*-Framework. Zusätzlich nutzt der Prototyp einige Hilfsmethoden aus diesem Framework. Unter anderem den Zugriff auf die Windows-API *User32.dll*.

4.3 Die einzelnen Projekte

Die Software besteht aus insgesamt 22 Projekten (vgl. Abbildung 2.4, Seite 18). Die eigentliche Anwendungslogik ist dabei in neun Projekte unterteilt, wobei sechs Projekte für die einzelnen *Controls* benötigt werden. Der Abhängigkeitsgraph in Abbildung 4.2 zeigt, wie die einzelnen Projekte miteinander verknüpft sind. Das Projekt *ReadTracker* ist dabei das Hauptprojekt. Die für die View nötigen *Controls* sind in separaten Projekten realisiert und in der Abbildung lediglich als Gruppe „Controls“ zu sehen.

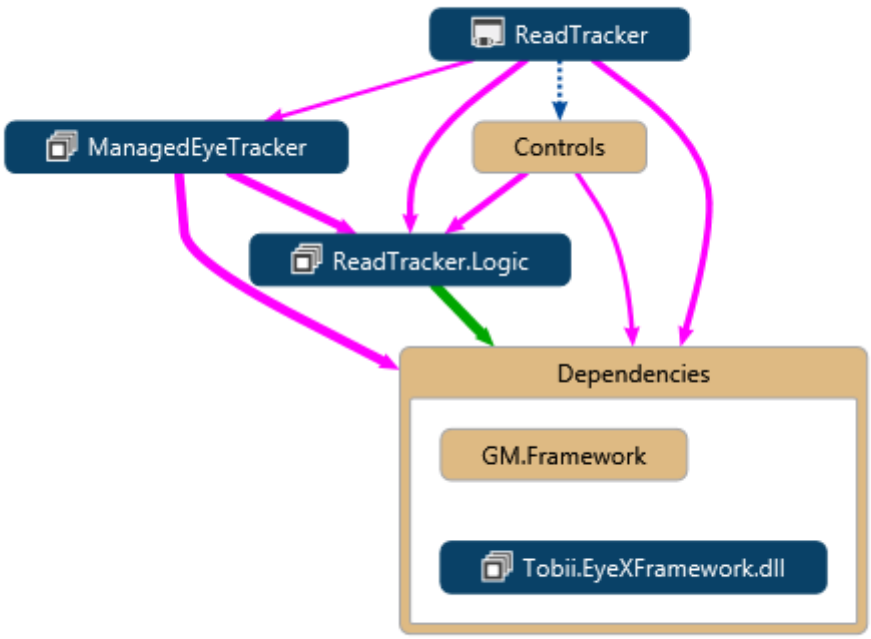


Abbildung 4.2: Übersicht der Projektabhängigkeiten

4.3.1 Projekt ReadTracker

Das *ReadTracker*-Projekt ist das Hauptprojekt. Wie Abbildung 4.4 zeigt enthält das Projekt den *Bootstrapper*, die Views und die für die Views benötigten *ValueConverter*. Der *Bootstrapper* wird dabei von der Klasse *App*, dem Einstiegspunkt einer *WPF*-Anwendung, aufgerufen. Er kümmert sich um den Programmablauf und erstellt unter anderem das Icon in der *Taskbar* oder sorgt dafür, dass das Hauptfenster angezeigt wird.

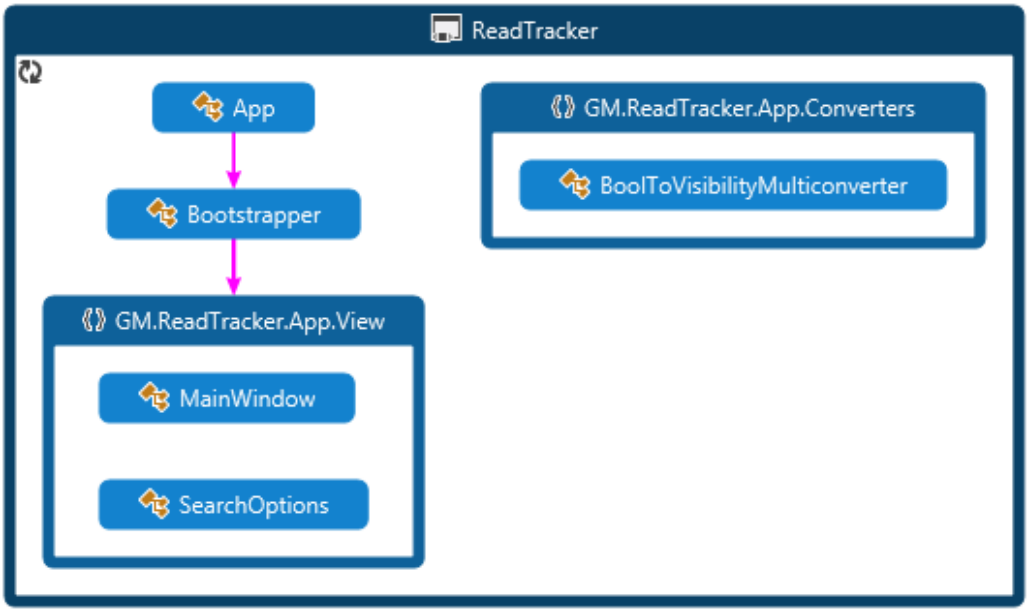


Abbildung 4.3: Abhängigkeitsgraph des ReadTracker-Projektes

4.3.2 Projekt ManagedEyeTracker

Das Projekt *ManagedEyeTracker* (vgl. Abbildung 4.4) enthält die Anbindung zum Tobii-Eye-Tracker. Diese wird absichtlich in einem eigenen Projekt entwickelt, um sie von der restlichen Anwendung bestmöglichst zu entkoppeln. Ziel dabei ist es, dass eine Anbindung an andere Eye-Tracker mit möglichst geringem Aufwand auch nachträglich integriert werden kann.

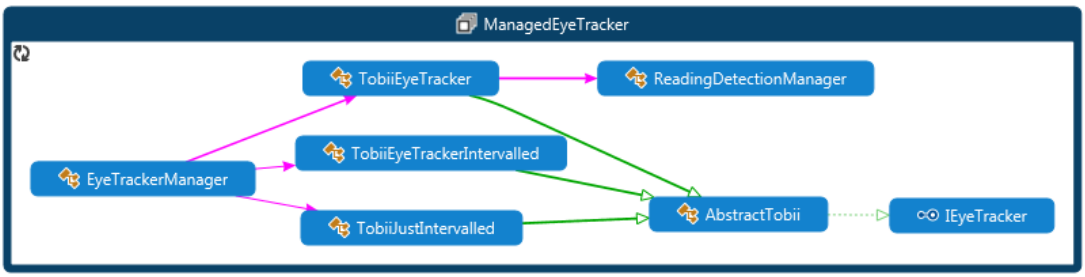


Abbildung 4.4: Abhängigkeitsgraph des ManagedEyeTracker-Projektes

Wie Abbildung 4.4 zeigt, müssen alle Eye-Tracker vom Interface *IEyeTracker* ableiten. In diesem Fall kann für die drei verschiedenen Eye-Tracker *TobiiJustIntervalled* (*Intervall*-Modus), *TobiiEyeTrackerIntervalled* (*Fokus*-Modus) und *TobiiEyeTracker* (*Lese*-Modus) eine abstrakte Basisklasse angelegt werden. Diese Basisklasse sorgt unter anderem dafür, dass sich alle drei Modi für den Studienteilnehmer optisch gleich verhalten. (Anzeige von Konfigurationen etc.). Der *EyeTrackerManager* sorgt dafür, dass für jeden Tag der Studie die entsprechende Eye-Tracker-Klasse und somit der entsprechende Modus ausgeführt werden. Er selbst implementiert das *IEyeTracker*-Interface und stellt somit eine Start- und Stopp- Methode bereit (vgl. Abbildung 4.5).

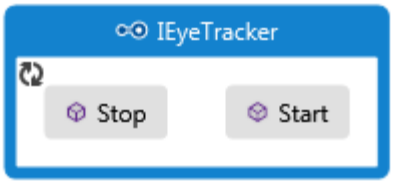


Abbildung 4.5: Das Interface IEyeTracker

4.3.2.1 ReadingDetectionManager

Der *ReadingDetectionManager* ist die Klasse, welche den Algorithmus für die Lese-Erkennung enthält (vgl. Abbildung 4.6).

Zunächst wird versucht, den Algorithmus aus dem Paper „A robust algorithm for reading detection“. (Campbell & Maglio, 2001) nachzuimplementieren. Es stellt sich schnell heraus, dass die Punkte, die der Eye-Tracker liefert, dafür zunächst aufbereitet werden müssen, um die Messschwankungen auszugleichen. Dazu müssen die Punkte, die einander nahe liegen, geclustert werden. Statt die Daten zu clustern, um sie dann zu verarbeiten ist es möglicherweise sinnvoller, das Clustern der Daten mit der Lese-Erkennung zu verbinden. Dadurch entsteht der Ansatz, die Lese-Erkennung mittels der Regressgeradensteigung umzusetzen. Dabei werden einige Punkte gesammelt und aus diesen Punkten eine Regressionsgerade errechnet. Sofern diese horizontal verläuft, eine bestimmte Mindestlänge aufweist und dabei eine festgelegte Steigung (unabhängig ob positiv oder negativ) nicht übersteigt, kann daraus erschlossen werden, dass sich die Augen an einer Line orientiert haben, wie es unter anderem beim Lesen der Fall ist. Bei der Regressionsgerade wird nicht geachtet, ob diese von links nach rechts oder umgekehrt verläuft, wodurch Rücksprünge in die Erkennung miteinbezogen werden.

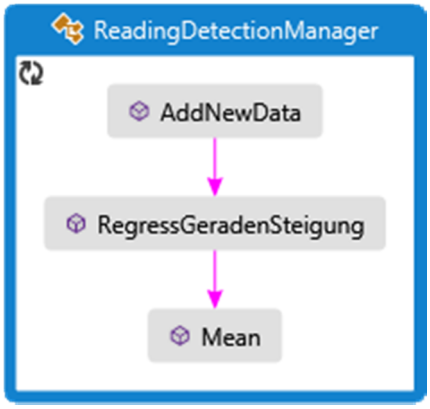


Abbildung 4.6: Der ReadingDetectionManager

Die Punkte sind in diesem Fall in einer *Queue*[26] mit Länge 25 organisiert. Das heißt, dass jeder Punkt sich auf seine unmittelbaren Vorgänger und Nachfolger auswirkt. Wann immer ein neuer Punkt über die Methode *AddNewData* hinzugefügt wird, wird gleichzeitig der älteste Punkt aus der *Queue* entfernt. Dies sorgt dafür, dass durchgängig geprüft wird, ob eine Lesebewegung stattfindet. Wenn immer 25 Punkte zusammengefasst werden entstehen Momentaufnahmen, die das Potential enthalten, die Erkennung zu verfälschen, wenngleich dies äußerst unwahrscheinlich ist.

Sobald die Regressionsgerade eine bestimmte Steigung unterschreitet und mindestens eine gewisse Länge aufweist, wird dieses als *Hit* gewertet. Das bedeutet, der Algorithmus merkt sich, dass die Regressionsgerade möglicherweise eine Lesebewegung anzeigen könnte, indem er den Wert der *Hit*-Variable inkrementiert. Weicht die Regressionsgerade vom Leseschema ab und wird nicht als *Sakkade* eingestuft, wird die *Hit*-Variable auf null gesetzt und die Erkennung wieder von neuem gestartet.

Als Zeilensprung wird die Regressionsgerade dann erkannt, wenn sie sich von rechts oben nach links unten bzw. von links oben nach rechts unten bewegt. Dies wird dadurch geprüft, dass die Steigung der Regressionsgerade (ob nun positiv oder negativ) in einem bestimmten Wertbereich liegen und die Regressionsgerade eine deutliche Länge aufweisen muss, um als Zeilensprung gewertet zu werden.

Wird ein Zeilensprung erkannt und hat der Algorithmus genügend *Hit*-Punkte gesammelt kann mit hoher Wahrscheinlichkeit davon ausgegangen werden, dass der Benutzer aktuell liest. In diesem Fall gibt der Algorithmus beim Aufruf der Methode *AddNewData* den Wert *true* zurück.

Dieser Algorithmus lässt entweder eine Optimierung hinsichtlich der Erkennungsgeschwindigkeit oder eine Optimierung hinsichtlich der Erkennungssicherheit zu. Wird die Zahl der benötigten Zeilensprünge und *Hits* reduziert, wird der Algorithmus schneller ein Leseverhalten feststellen. Werden diese Werte erhöht, wird der Algorithmus hinsichtlich der Erkennung von Lesebewegungen zuverlässiger und produziert seltener *false positives*[27].

[26] Queue meint in C# die generische Klasse Queue, die als FIFO-Queue realisiert ist.
[27] Von einem „false positive“ wird bei z.B. einem Testverfahren dann gesprochen, wenn der Test positiv verläuft, obwohl er eigentlich negativ verlaufen sollte.

4.3.3 Projekt ReadTracker.Logic

Im Projekt *ReadTracker.Logic* befindet sich der Großteil der Programmlogik. Aus diesem Grund wird es von den anderen Projekten referenziert und bildet dadurch die Basis der Anwendung. Alle *ViewModels* sind hier enthalten und strikt von der *View* getrennt. Ebenso sind hier alle Grafiken enthalten, die in der Anwendung verwendet werden. In Abbildung 4.7 ist zu sehen, dass das Projekt auch eine Reihe von *Managern* enthält. Dabei steht jeder *Manager* für eine bestimmte Aufgabe um die er sich kümmert. Ein *Manager* ist in der Regel eine *Singleton*-Implementierung oder eine statische Klasse, sodass die Daten und Methoden, die er bereitstellt, in anderen Programmteilen zur Verfügung stehen. Die Aufgaben der einzelnen *Manager* werden in den folgenden Kapiteln näher erläutert. Das Projekt enthält darüber hinaus auch das *IWinServices*-Interface. Dieses wird vom *ServiceManager* (vgl. Kapitel 4.1, Seite 26) genutzt um den *ViewModels* zu ermöglichen modale Dialoge und *MessageBox*en anzuzeigen.

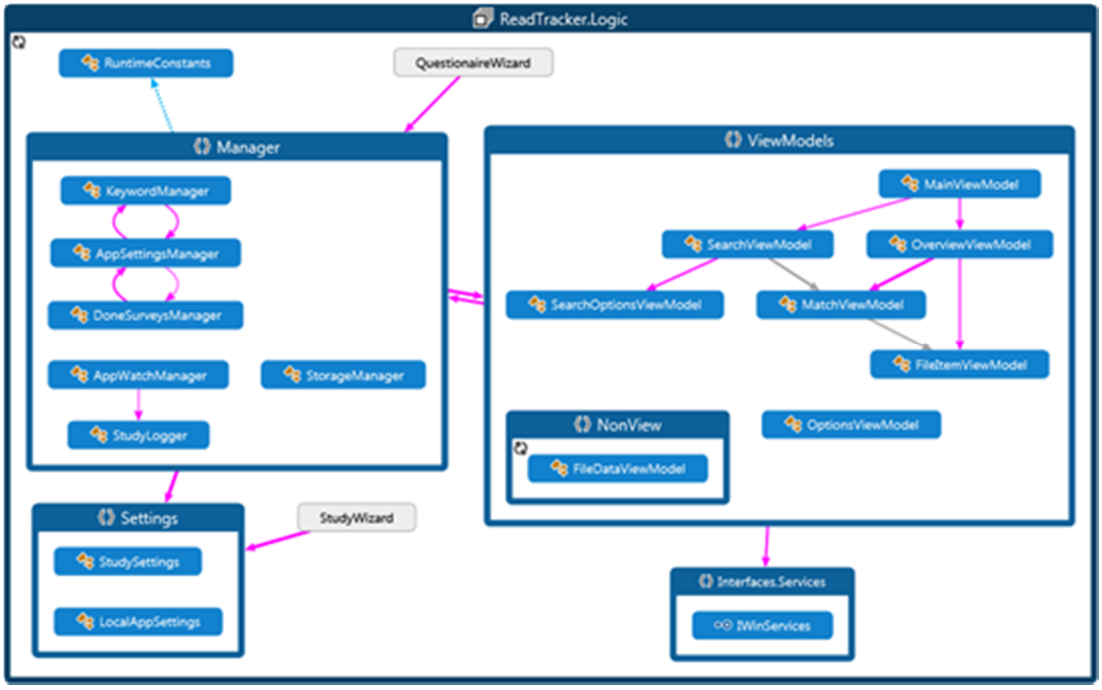


Abbildung 4.7: Abhängigkeitsgraph des ReadTracker.Logic-Projektes

4.3.3.1 AppSettingsManager

Der *AppSettingsManager* ist eine *Singleton*-Klasse, die sich um das Laden und Speichern der Anwendungs-Konfigurationen kümmert. Dabei werden die Daten erst geladen, wenn der erste Zugriff auf die Daten die *Singleton*-Instanz erzeugt. Für den Fall, dass beim Laden ein Fehler auftritt steht ein Event zur Verfügung, welches die entsprechende Fehlermeldung übergibt.

4.3.3.2 AppWatchManager

Der *AppWatchManager* ist eine *Singleton*-Klasse, die in einem eigenen *Thread* überwacht, ob sich die aktive Anwendung ändert. Diese Funktion wird nur für die Studie benötigt.

Der *Manager* stellt eine Start- und eine Stop-Methode zur Verfügung, über die er gesteuert wird. Beim Starten des *Managers* erzeugt dieser einen neuen *Task*, der im angegebenen Zeitabstand prüft, ob sich die aktive Anwendung geändert hat. Dazu wird die Prozess-ID des aktiven Prozesses abgerufen und es wird überprüft, ob diese mit der Prozess-ID bei der letzten Prüfung übereinstimmt. Falls diese ID nicht übereinstimmt schreibt der Manager selbstständig einen entsprechenden Eintrag in das Study-Log. Der interne Task läuft so lange, bis er durch die Stop-Methode abgebrochen wird.

4.3.3.3 DoneSurveysManager

Der *DoneSurveysManager* ist eine *Singleton*-Klasse und stellt die Information bereit, ob der Studien-Teilnehmer am aktuellen Tag bereits an der Umfrage teilgenommen hat. Diese Funktion wird nur für die Studie benötigt.

Die *Singleton*-Instanz des *DoneSurveysManager* wird erzeugt, sobald von diesem zum ersten Mal Informationen angefordert werden. Dabei holt sich der *DoneSurveysManager* vom *AppSettingsManager* die entsprechenden gespeicherten Daten, verarbeitet diese und stellt sie zur Verfügung.

4.3.3.4 StorageManager

Der *StorageManager* ist ebenfalls eine statische Klasse, welche die Methoden zum Laden, Speichern und Ändern der Einträge des Textarchives zur Verfügung stellt.

4.3.3.5 StudyLogger

Der *StudyLogger* ist eine *Singleton*-Klasse, die einen Log4Net-Logger erzeugt und mit diesem die verschiedenen Ereignisse der Benutzerstudie in eine Textdatei loggt. Diese Funktion wird nur für die Studie benötigt.

4.4 Die Anwendung

Sobald die Anwendung gestartet wird sorgt der *Bootstrapper* dafür, dass in die *Taskleiste* ein Icon eingetragen wird, über welches die Anwendung geöffnet und beendet werden kann (vgl. Abbildung 4.8).

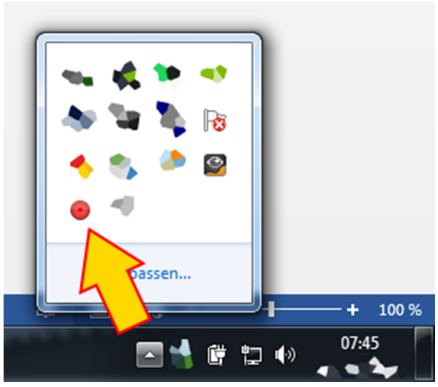


Abbildung 4.8: Anwendung in der Taskleiste

Anschließend steuert die Anwendung die Kalibrirungsfunktionen des Tobii-Eye-Trackers an. Dabei muss dieser zunächst am Bildschirm ausgerichtet werden (vgl. Abbildung 4.9). Dies ist nur dann notwendig, wenn der Tobii-Eye-Tracker nicht fest am Bildschirm befestigt wird, wie in Abbildung 2.1 (). Danach wird der Eye-Tracker neu kalibiriert (vgl. Abbildung 4.10)

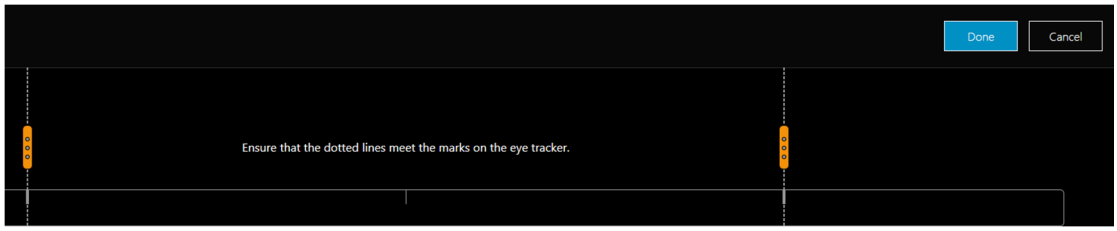


Abbildung 4.9: Tobii am Bildschirm ausrichten

Im Anschluss läuft die Anwendung im Hintergrund um den Benutzer bei seiner Arbeit nicht zu stören.

Über einen Rechtsklick auf das Symbol in der *Taskleiste* lässt sich die Anwendung öffnen oder schließen. Sobald die das Hauptfenster geöffnet ist, wird der Tobii-Eye-Tracker abgeschalten, um zu verhindern, dass die Tracking-Anwendung getrackt wird.

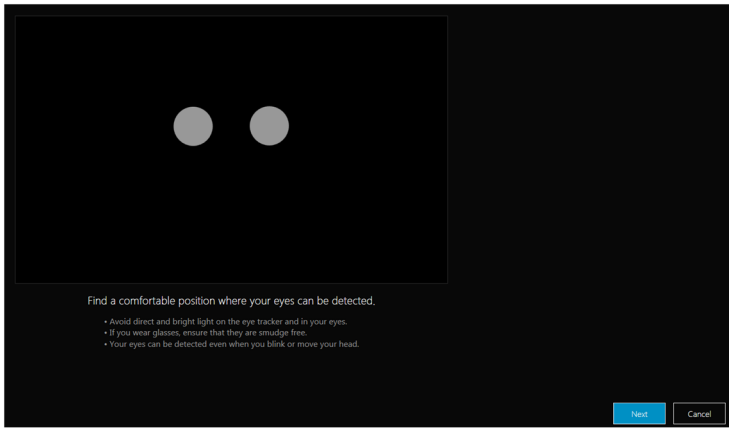


Abbildung 4.10: Tobii-Kalibrierung

Sobald die Tracking-Anwendung (vgl. Abbildung 4.11, Seite 34) geschlossen wird schaltet diese den Tobii-Eye-Tracker automatisch wieder ein und setzt das Tracking fort.

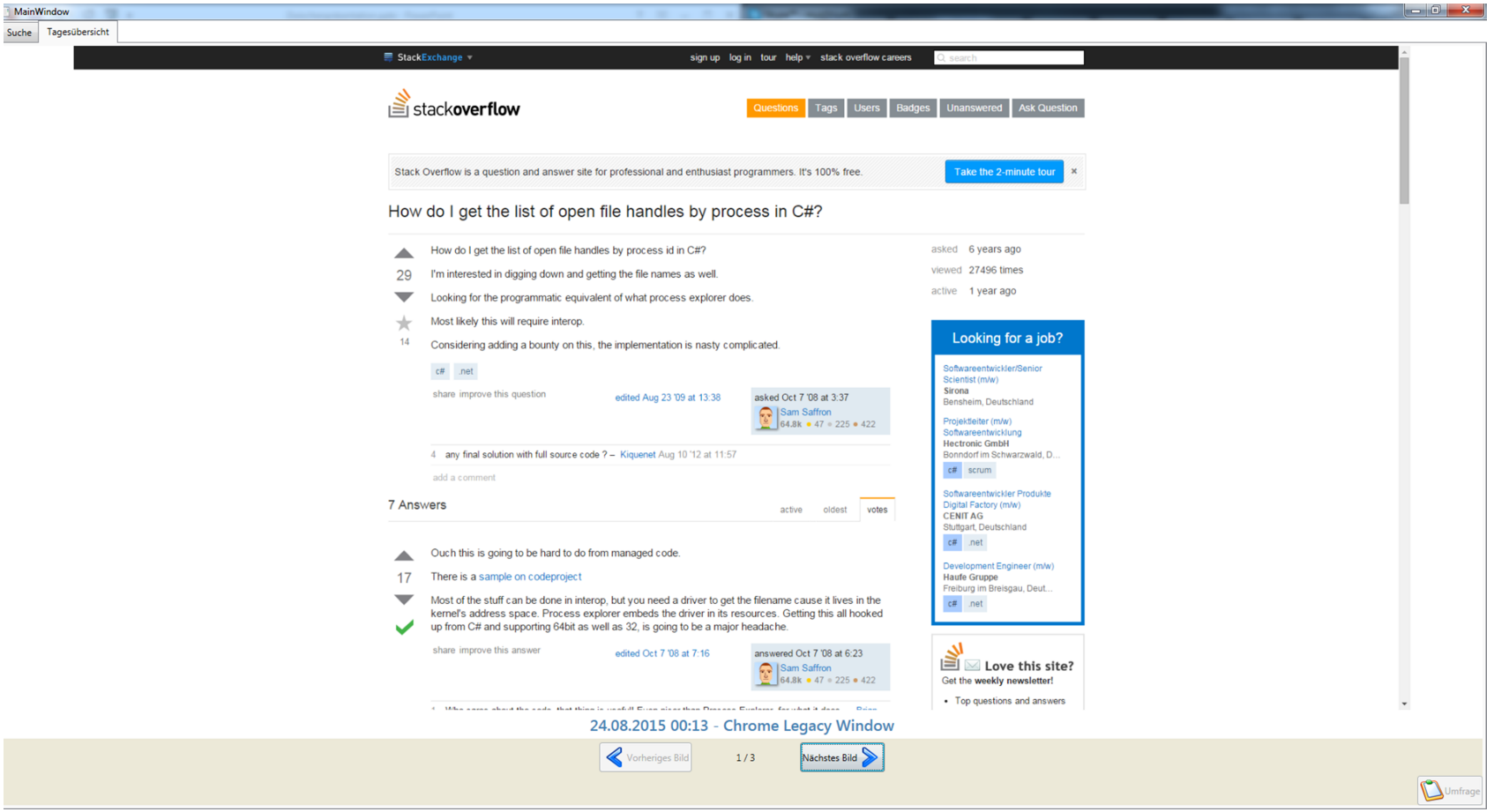


Abbildung 4.11: Die Rückschau des Prototyps

5 Nutzerstudie

An der Nutzerstudie sind zwölf Teilnehmer beteiligt. Über einen Zeitraum von drei Wochen sollen jede Woche vier Teilnehmer an der Studie teilnehmen, da insgesamt vier Eye-Tracker (Tobii EyeX, vgl. Kapitel 2.1.3, Seite 13) zur Verfügung stehen. Dabei ist die Nutzerstudie in eine Laborstudie und eine anschließende Feldstudie aufgeteilt. Die Feldstudie dauert jeweils drei Tage, da hierbei pro Tag eine der drei Studienbedingungen (vgl. Kapitel 5.2, Seite 36) getestet werden soll. Nach der Feldstudie wird noch einmal mit jedem Teilnehmer separat ein kurzes Abschlussgespräch geführt. Darin werden einige abschließende Fragen besprochen und der Teilnehmer hat die Gelegenheit, allgemeine Kritik und Anregungen zu äußern. Als abschließendes Fazit soll der Teilnehmer sich noch zu der Frage äußern, ob er sich die Nutzung des Systems zur Textverwaltung in Zukunft vorstellen kann und welchen Mehrwert er sich daraus verspricht.

5.1 Teilnehmer

Die Versuchspersonen nehmen alle freiwillig an der Studie teil und werden über den Verlauf der Studie, sowie über datenschutzrechtliche Belange in Kenntnis gesetzt.

Von insgesamt 12 Teilnehmern sind 4 weiblich und 8 männlich. Dabei unterteilt sich die Gruppe in 8 Studenten, 3 Entwickler und ein Rentner.

Das Alter der Probanden erstreckt sich von 20 bis 65, mit einem durchschnittlichen Alter von etwa 28 Jahren, bei einer Standardabweichung von 11,6 Jahren.

7 Teilnehmer geben an, dass sie eine Brille bzw. Kontaktlinsen tragen. In jedem Fall wird darauf geachtet, dass die entsprechende Option („keine Sehhilfe“, „Brille“, „Kontaktlinsen“) bei der Kalibrierung der Eye-Trackers angegeben und so durch die Eye-Tracker-Software berücksichtigt wird.

9 Teilnehmer arbeiten während der Studie an einem externen Monitor, wobei teilweise auch mehrere Bildschirme genutzt werden.

Die Teilnehmer werden befragt, wie lange sie unter der Woche im Schnitt pro Tag am Bildschirm verbringen. Hierbei geben 6 Teilnehmer an, dass sie 8 Stunden und mehr am Bildschirm verbringen. 6 bis 8, 2 bis 4 sowie 1 bis 2 Stunden werden von jeweils 1 Person und 4 bis 6 Stunden von 3 Personen als durchschnittliche Zeit angegeben.

3 Teilnehmer geben an, dass sie bereits in irgendeiner Form ein System zur Archivierung gelesener Texte nutzen und 6 Teilnehmer geben an, ein System zur Verfolgung ihrer Arbeitsfortschritte zu verwenden. 1 Person davon nutzt beides, während 4 Personen gar kein System nutzen. Dabei geben viele Probanden für das System zur Archivierung ihrer gelesenen Texte ihre E-Mail-Software an. Ein System, welches alle gelesenen Texte verwaltet nutzt zum Zeitpunkt der Befragung keiner der Probanden.

5.2 Programm-Modi (Bedingungen)

Insgesamt werden in der Studie drei Bedingungen getestet, die in unterschiedlichen Reihenfolgen durchgeführt werden. Da sich bei drei Bedingungen insgesamt sechs verschiedene Reihenfolgen ergeben, wird jede von zwei Probanden getestet.

Den Teilnehmern ist während der Studie bekannt, dass das Erzeugen der *Screenshots* jeden Tag durch andere Kriterien und Bedingungen ausgelöst wird. Allerdings weiß keiner der Probanden, welcher Modus für welche Kriterien und Bedingungen steht und in welcher Reihenfolge die verschiedenen Modi verwendet werden. Obwohl ein Modus vollständig auf die Daten verzichtet, welche durch den Eye-Tracker bereit stehen, wird dieser dennoch, analog zu den beiden anderen Modi, angeschaltet. Dies soll dem Nutzer suggerieren, dass der Eye-Tracker weiterhin genutzt wird, um mögliche Störfaktoren ausschließen zu können. Somit war der Ablauf an jedem Tag für den Teilnehmer komplett identisch.

Modus 1 („Intervall“): Es wird alle zwei Minuten ein *Screenshot* der aktiven Anwendung gemacht.

Modus 2 („Fokus“): Sobald der Nutzer mit den Augen eine andere Anwendung fokussiert, wird von dieser Anwendung ein *Screenshot* gemacht. Sollten die Augen zwei Minuten in der gleichen Anwendung verweilen wird ein weiterer *Screenshot* der Anwendung gemacht.

Modus 3 („Lesen“): Dieser Modus macht immer dann ein *Screenshot*, wenn der Lese-Erkennungs-Algorithmus eine Lesetätigkeit detektiert, jedoch höchstens alle zwei Minuten, wenn es sich um dieselbe Anwendung handelt.

5.3 Aussagen der täglichen Umfrage

Der Fragebogen, welcher von den Teilnehmern nach jedem (Arbeits-) Tag beantwortet werden soll besteht aus insgesamt 16 Aussagen die mittels Likert-Skala von 1 („Stimme gar nicht zu“) bis 7 („Stimme voll zu“) bewertbar sind, sowie einem Freitextfeld am Ende für Anregungen und Feedback. Dabei lauten die Aussagen der Umfrage wie folgt:

1. Ich habe heute sehr effektiv gearbeitet.
2. Ich empfand den Umfang der Rückschau als angemessen.
3. Die für die Rückschau erforderliche Zeit war angemessen.
4. Die Rückschau empfand ich als nützlich.
5. Die Rückschau hat meine heutigen Aktivitäten gut wiedergegeben.
6. Die Rückschau enthielt die wichtigsten Aktivitäten von heute.
7. Anhand der Screenshots konnte ich meine heutigen Aktivitäten gut nachvollziehen.
8. Die gezeigten Screenshots empfand ich als relevant.
9. Diese Rückschau würde ich gerne auch für künftiges Nachschlagen archivieren.
10. Die Rückschau hat mir geholfen, mir meine heutigen Aktivitäten ins Gedächtnis zu rufen.
11. Ich hätte gern auch Daten aus vorherigen Tagen in der Rückschau gehabt.
12. Die Rückschau beinhaltete für mich bedenkliche Inhalte.
13. Die Rückschau hat Daten beinhaltet, die ich lieber nicht archivieren möchte.
14. Ich habe die Such- oder Filterfunktion ausgiebig genutzt.
15. Ich fand die Suchfunktion hilfreich.
16. Die Suchfunktion hat nützliche Resultate produziert.

5.4 Durchführung

Die folgenden Unterkapitel beschreiben die beiden Phasen der Nutzerstudie im Detail.

5.4.1 Laborstudie

In der Laborstudie wird das Einverständnis zur Teilnahme eingeholt und ein kurzer Eingangsfragebogen ausgefüllt. Im Eingangsfragebogen werden als demographische Informationen das Alter, das Geschlecht und der Beruf bzw. die Haupttätigkeit abgefragt.

Außerdem wird vermerkt, wie viel Zeit der Teilnehmer durchschnittlich an einem normalen Arbeitstag am Bildschirm verbringt. Weiterhin wird festgestellt, ob der Teilnehmer bereits ein System zur Verwaltung von gelesenen Texten und ein System zur Verwaltung der täglichen (Arbeits-) Fortschritte nutzt.

Im Anschluss dazu wird die Software zur Verwaltung der gelesenen Texte auf dem System des Teilnehmers installiert und die Handhabung von Soft- und Hardware ausführlich erklärt.

Zuletzt wird eine Performanz-Analyse durchgeführt, bei welcher der Teilnehmer zunächst einen Paragraphen normal lesen, danach einen Artikel überfliegen und zuletzt einen dritten Artikel nach einem Schlüsselwort durchsuchen muss. Als Ergebnis dieser Performanz-Analyse soll festgestellt werden, wie gut die Erkennungsgenauigkeit der Lese-Erkennung bzw. wie hoch die Fehlerrate ist.

5.4.2 Feldstudie

Abbildung 5.1: Liste der Aktiven (Hintergrund-)Prozesse in der Taskleiste unter Windows 8

In der Feldstudie wird an jedem der drei Versuchstage eine andere Bedingung getestet. Dabei wird am Ablauf, wie ihn der Teilnehmer wahrnimmt, an keinem der drei Tage etwas verändert um etwaige Störfaktoren ausschließen zu können. Der Proband soll zu Beginn seines (Arbeits-) Tages am Rechner das Tracking-Programm starten, welches zunächst im Hintergrund läuft und über ein Icon in der *Taskleiste* geöffnet werden kann. Anschließend soll der Teilnehmer seinen Tag wie gewohnt verbringen. Dabei erzeugt das Tracking-Programm eine Tagesübersicht, wobei die jeweilige Bedingung vorgibt, nach welchen Kriterien diese angelegt werden soll. Am Ende des Tages soll der Proband sich die komplette Tagesübersicht ansehen und anschließend einen Fragebogen ausfüllen, in welchem er die Übersicht und die Suchfunktion des Programmes bewertet. Um zu verhindern, dass diese abschließende Umfrage vergessen wird, blendet das Programm ab 17:00 Uhr stündlich eine Erinnerung als *PopUp* ein. Zudem wird vor Beendigung des Tracking-Programmes, falls die Umfrage noch nicht gemacht wurde, abgefragt, ob der Teilnehmer diese Umfrage vor der Beendigung des Programmes noch durchführen möchte.

5.4.3 Abschlussgespräch

Nachdem ein Teilnehmer die Studie drei Tage durchgeführt hat, wird ihm noch einmal gezeigt, welche Daten letztendlich zur Auswertung der Studie benötigt werden. Dabei wird jedem Teilnehmer noch einmal erklärt, was die einzelnen Einträge bedeuten, sodass er noch einmal entscheiden kann, ob diese Daten im Rahmen dieser Bachelorarbeit genutzt werden dürfen oder ob er im letzten Moment die Studie doch noch abbrechen möchte.

Außerdem wird in einem abschließenden Interview gefragt, wie der Teilnehmer mit der Handhabung der Software zurechtkam und ob das Programm ihn während seiner Arbeitsphasen gestört hat. Weiterhin wird abgefragt, ob der Teilnehmer sich durch den Eye-Tracker oder das Programm in irgendeiner Weise beobachtet oder unwohl gefühlt hat. Zuletzt soll der Teilnehmer noch darüber nachdenken, ob er eine solche Software in Zukunft nutzen möchte und welchen Nutzen er sich von der Nutzung verspricht. Dabei wird auch besprochen, welche Features die Anwendung noch bieten müsste, um dem Teilnehmer den gewünschten Mehrwert bieten zu können.

6 Ergebnisse

In diesem Kapitel werden die Ergebnisse der täglichen Umfrage präsentiert und für jedes Diagramm die wichtigsten Auffälligkeiten beschrieben. Für jede Aussage wurde ein Diagramm angelegt. Dabei sind auf der horizontalen Achse die einzelnen Probanden (S01 bis S12) und zuletzt der Mittelwert (MW) der Antworten aufgeführt. Für jeden Probanden gibt es drei Säulen, welche den verschiedenen Modi (vgl. Kapitel 5.2, Seite 36), die von den Probanden getestet wurden, entsprechen. Die Modi sind in einer einheitlichen Reihenfolge angegeben, die nur in zwei Fällen der Durchführungsreihenfolge entspricht. Die erste Säule (grün) stellt dabei den *Intervall*-Modus, die zweite Säule (blau) den *Fokus*-Modus und die dritte Säule (gelb) den *Lese*-Modus dar.

6.1 Effektiv gearbeitet

Wie Diagramm 6.1 zeigt ist die Effektivität der einzelnen Teilnehmer über den Testzeitraum annähernd konstant, was sich auch im Mittelwert wiederspiegelt. Da die eigene Produktivität ein sehr subjektiver Wert ist, lassen sich die Werte nur schlecht mit denen der anderen Teilnehmer vergleichen. Auffällig ist, dass Proband S10 bei der dritten Bedingung einen deutlichen Produktivitätseinbruch hat, wohingegen Proband S06 die Produktivität über die verschiedenen Modi hinweg steigert.

In Abhängigkeit zum verwendeten Modus ergibt die Bewertung der eigenen Effektivität der Probanden zwischen den Modi keinen statistisch signifikanten Unterschied ($\chi^2(2) = 1{,}895 \mid p = 0{,}388$).

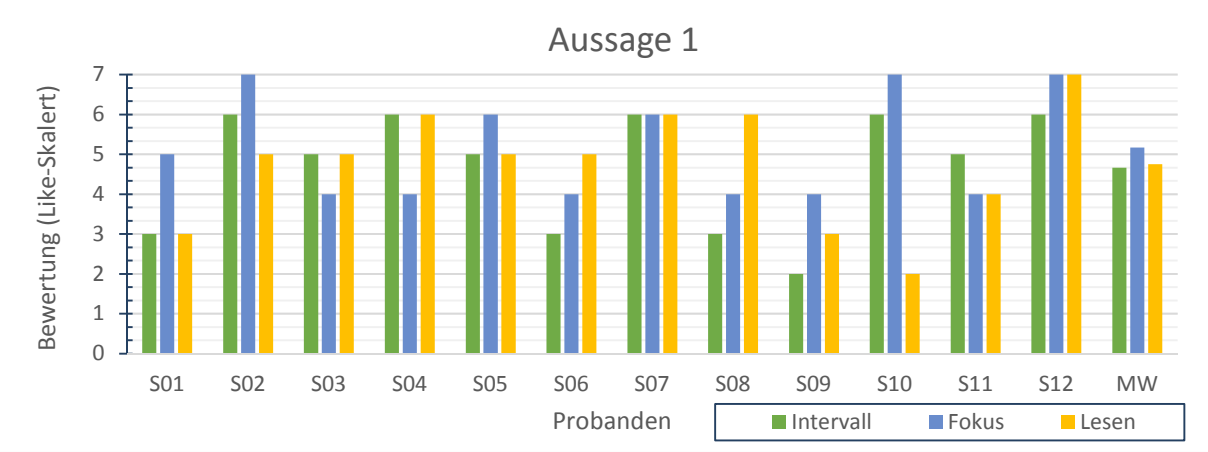


Diagramm 6.1: Likert-Skala-Bewertung der Probanden und Mittelwert zur Aussage: "Ich habe heute sehr effektiv gearbeitet."

6.2 Angemessener Umfang

In Diagramm 6.2 ist dargestellt, wie der Umfang der Rückschau bewertet ist. Auffällig sind dabei die Bewertungen der Teilnehmer S01, S03, S05, S06 und S11.

Teilnehmer S01 empfindet den Umfang der Rückschau beim *Intervall*-Modus weniger angemessen als den des *Fokus*-Modus. Die Teilnehmer S03, S05, S06 und S11 hingegen bewerten den *Lese*-Modus einen Punkt schlechter als den *Intervall*-Modus.

In Abhängigkeit zum verwendeten Modus ergibt die Bewertung der Probanden, ob der Umfang der Rückschau jeweils für angemessen gehalten wird, einen statistisch signifikanten Unterschied: $\chi^2(2) = 13{,}911 \mid p = 0{,}001$. Als Post-hoc-Analyse wurde ein Wilcoxon-Vorzeichen-Rang-Test durchgeführt und die Ergebnisse mit einer Bonferroni-Korrektur bereinigt. Dies ergibt ein Signifikanzniveau von $p < 0{,}017$.

Es gibt keine statistisch signifikanten Unterschiede zwischen dem *Intervall*-Modus und dem *Fokus*-Modus ($Z = -2{,}155 \mid p = 0{,}31$) als auch zwischen dem *Intervall*-Modus und dem *Lese*-Modus ($Z = -1{,}703 \mid p = 0{,}089$). Allerdings gibt es einen statistisch signifikanten Unterschied zwischen dem *Fokus*- und dem *Lese*-Modus ($Z = -3{,}078 \mid p = 0{,}002$). Der Umfang der Rückschau im *Lese*-Modus wird dabei als angemessener empfunden.

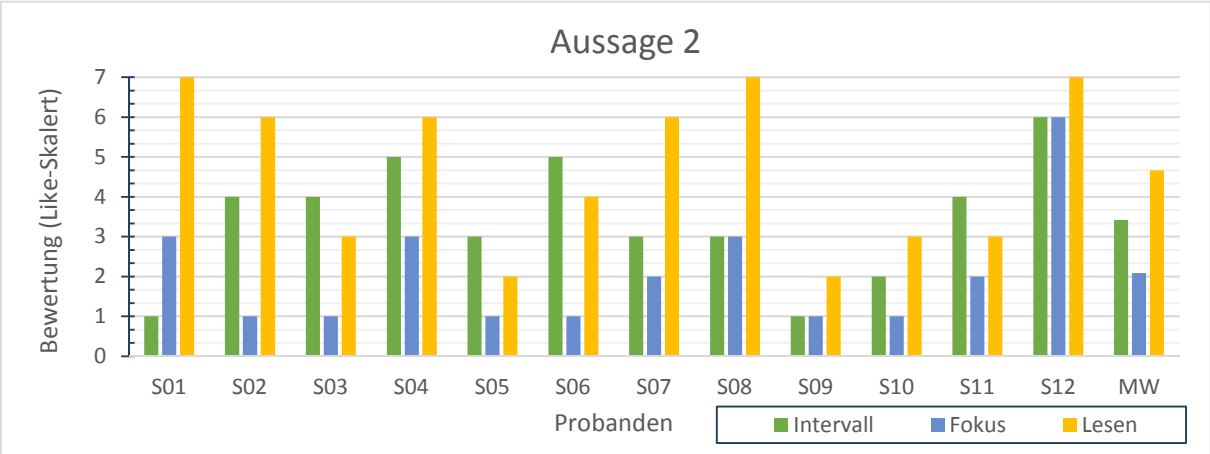


Diagramm 6.2: Likert-Skala-Bewertung der Probanden und Mittelwert zur Aussage: "Ich empfand den Umfang der Rückschau als angemessen."

6.3 Angemessener Zeitaufwand

Bei der dritten Aussage schneidet ebenfalls der *Lese*-Modus im Durchschnitt am besten ab, wie Diagramm 6.3 zeigt. Bis auf Teilnehmer S11 sind sich hier auch alle einig, dass der *Fokus*-Modus bezüglich der benötigten Zeit am wenigsten angemessen und der *Lese*-Modus am angemessensten ist.

In Abhängigkeit zum verwendeten Modus ergibt die Bewertung der Probanden, ob die, für die Rückschau erforderliche Zeit jeweils für angemessen gehalten wird, einen statistisch signifikanten Unterschied: $\chi^2(2) = 15{,}350 \mid p < 0{,}001$. Als Post-hoc-Analyse wurde ein Wilcoxon-Vorzeichen-Rang-Test durchgeführt und die Ergebnisse mit einer Bonferroni-Korrektur bereinigt. Dies ergibt ein Signifikanzniveau von $p < 0{,}017$.

Es gibt keine statistisch signifikanten Unterschiede zwischen dem *Intervall*-Modus und dem *Fokus*-Modus ($Z = -1{,}672 \mid p = 0{,}95$) als auch zwischen dem *Intervall*-Modus und dem *Lese*-Modus ($Z = -2{,}355 \mid p = 0{,}019$). Allerdings gibt es einen statistisch signifikanten Unterschied zwischen dem *Fokus*- und dem *Lese*-Modus ($Z = -2{,}82 \mid p = 0{,}005$). Die für die Rückschau erforderliche Zeit im *Lese*-Modus wird dabei als angemessener empfunden.

Besonders interessant sind in diesem Diagramm die Bewertungen der Teilnehmer S01, S05, S08 und S09. Bei Ihnen zeigt die Bewertung der Modi erhebliche Unterschiede. So empfindet Teilnehmer S01 die benötigte Zeit für den *Intervall*-Modus absolut unangemessen, während die Teilnehmer S05 und S09 sich einig sind, dass nur der *Lese*-Modus vom Zeitaufwand her angemessen ist. Die Bewertung von Teilnehmer S08 stimmt prinzipiell mit der Bewertung der meisten anderen überein, enthält aber die größten Unterschiede zwischen den einzelnen Modi.

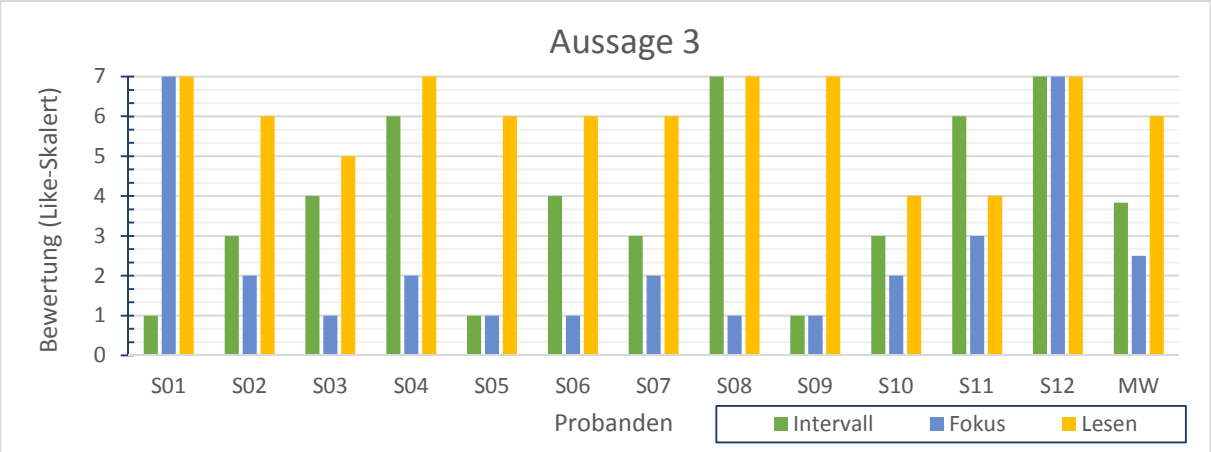


Diagramm 6.3: Likert-Skala-Bewertung der Probanden und Mittelwert zur Aussage: "Die für die Rückschau erforderliche Zeit war angemessen."

6.4 Nützliche Rückschau

Die Daten von Aussage 4 lassen sich nicht so ohne weiteres auswerten, da die Bewertungen stark auseinander gehen. Zudem ist die Nützlichkeit der Rückschau sehr subjektiv und lässt sich daher nur schwer zwischen den einzelnen Teilnehmern vergleichen. So gehen auch die Meinungen, ob und welche Rückschau nützlich ist sehr weit auseinander.

Im Gesamten zeigt der Mittelwert aber den Trend, dass der *Lesen*-Modus die nützlichste Rückschau liefert wohingegen der *Fokus*-Modus tendenziell eher am schlechtesten abschließt.

In Abhängigkeit zum verwendeten Modus ergibt die Bewertung der Probanden zur Nützlichkeit der Rückschau zwischen den Modi keinen statistisch signifikanten Unterschied ($\chi^2(2) = 5{,}150 \mid p = 0{,}076$).

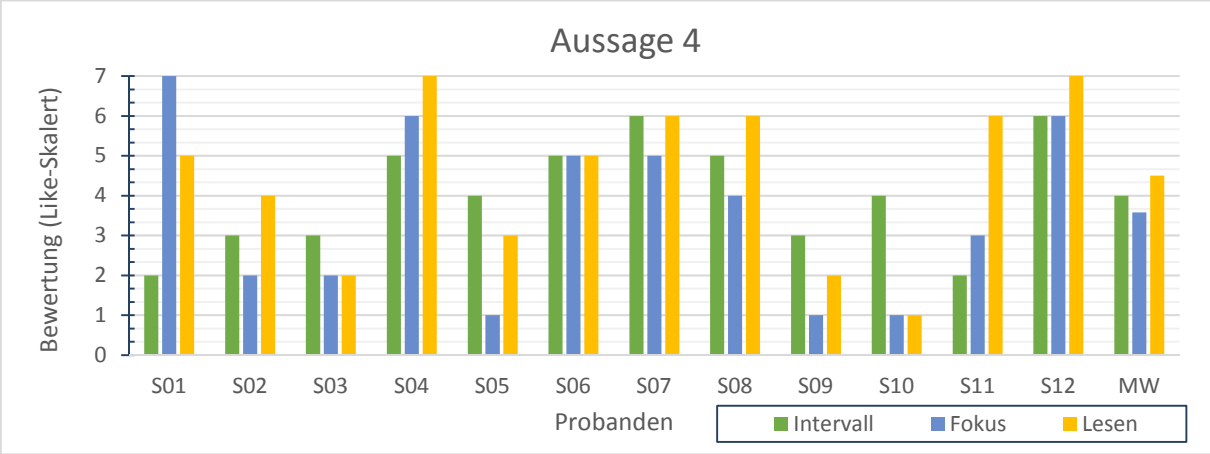


Diagramm 6.4: Likert-Skala-Bewertung der Probanden und Mittelwert zur Aussage: "Die Rückschau empfand ich als nützlich."

6.5 Aktivitäten gut wiedergegeben

Bei Assage 5 fällt auf, dass die Bewertungen nur selten weniger als 4 Punkte betragen. Während dies beim *Intervall*- und *Fokus*-Modus nur jeweils einmal vorkommt (S11 & S09) wird der *Lese*-Modus insgesamt viermal (S03, S05, S06, S09) mit weniger als 4 Punkten bewertet. Weiterhin fällt auf, dass die Bewertung zwischen den einzelnen Modi bei 50% der Teilnehmer nicht oder nur um einen Punkt variiert.

In Abhängigkeit zum verwendeten Modus ergibt die Bewertung der Probanden, wie gut die Wiedergabe der Aktivitäten durch die Rückschau war, keinen statistisch signifikanten Unterschied ($\chi^2(2) = 0{,}4 \mid p = 0{,}891$).

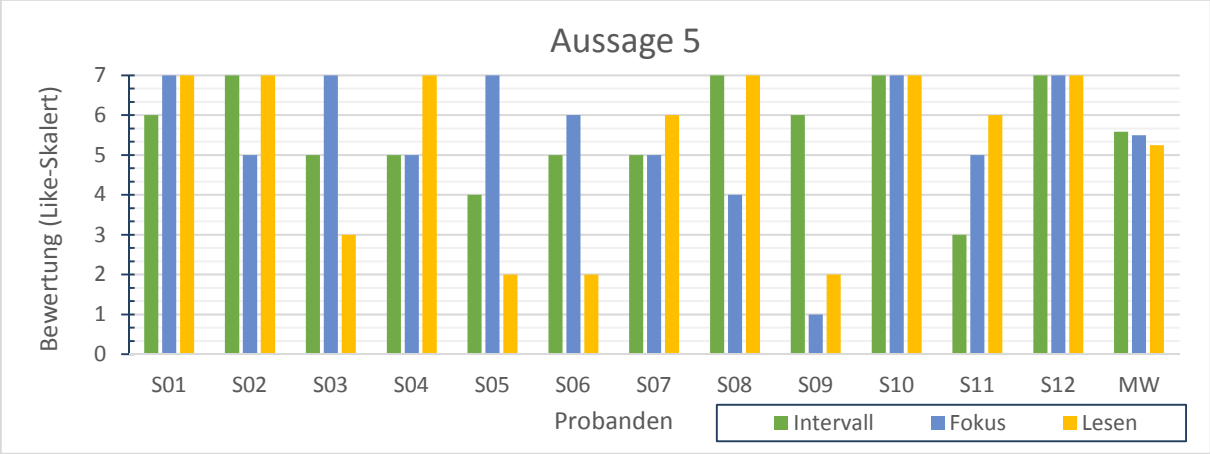


Diagramm 6.5: Likert-Skala-Bewertung der Probanden und Mittelwert zur Aussage: "Die Rückschau hat meine heutigen Aktivitäten gut wiedergegeben."

6.6 Wichtigste Aktivitäten enthalten

In diesem Fall sind sich die Nutzer sehr uneinig. Während fünf Nutzer (S01, S04, S07, S10 und S12) kaum einen Unterschied zwischen den einzelnen Modi erkennen können, schwanken die Bewertungen bei den anderen Teilnehmern teilweise erheblich. Dies spiegelt sich so auch im Mittelwert wieder, bei dem alle drei Modi in etwa gleich abschneiden.

In Abhängigkeit zum verwendeten Modus ergibt die Bewertung der Probanden, ob die Rückschau die wichtigsten Aktivitäten enthält, zwischen den Modi keinen statistisch signifikanten Unterschied ($\chi^2(2) = 0{,}062 \mid p = 0{,}969$).

Auffällig ist, dass lediglich der *Lese*-Modus von den Teilnehmern teilweise mit nur einem Punkt bewertet ist. Außerdem fällt auf, dass sowohl Teilnehmer S05 als auch Teilnehmer S09 höchstens vier Punkte vergeben, mit dem Ergebnis der Rückschau also teilweise nicht zufrieden sind. Dabei schneidet der *Lese*-Modus bei den Beiden noch am besten ab.

Weiterhin fällt auf, dass die Teilnehmer S02, S08 und S11 ebenfalls den *Lese*-Modus, teilweise jedoch mit erheblichem Unterschied zu den anderen Modi, am besten bewerten.

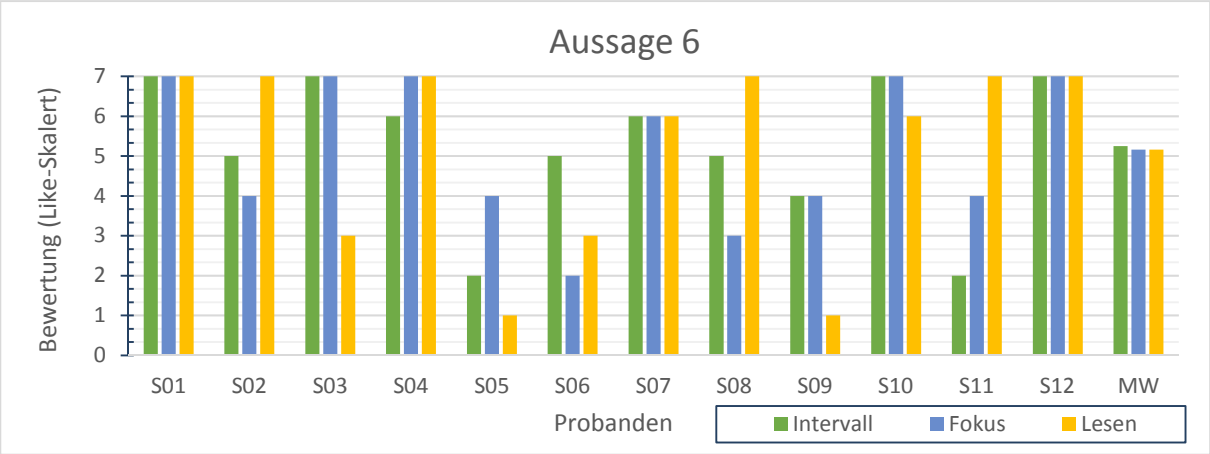


Diagramm 6.6: Likert-Skala-Bewertung der Probanden und Mittelwert zur Aussage: "Die Rückschau enthielt die wichtigsten Aktivitäten von heute."

6.7 Aktivitäten nachvollziehbar

In Abhängigkeit zum verwendeten Modus ergibt die Bewertung der Probanden, ob anhand der Rückschau die Tagesaktivitäten nachvollzogen werden kann, zwischen den Modi keinen statistisch signifikanten Unterschied ($\chi^2(2) = 1{,}056 \mid p = 0{,}59$).

Auch hier gehen die Meinungen teilweise stark auseinander, bei welchem Modus die *Screenshots* dem Benutzer dabei am besten helfen, die täglichen Aktivitäten nachzuvollziehen. Interessanterweise ist die Bewertung des *Lese*-Modus, der eigentlich die wenigsten Bilder liefert, im Schnitt sogar auf Platz zwei.

Es fällt außerdem auf, dass alle Teilnehmer (bis auf S09 und S11) bei mindestens einem Modus der Aussage voll zustimmen. Dabei hat aber Teilnehmer S11 der Aussage zumindest beim *Lese*-Modus stark zugestimmt, wohingegen Teilnehmer S09 offensichtlich mit keinem der drei Modi seine Aktivitäten gut nachvollziehen kann.

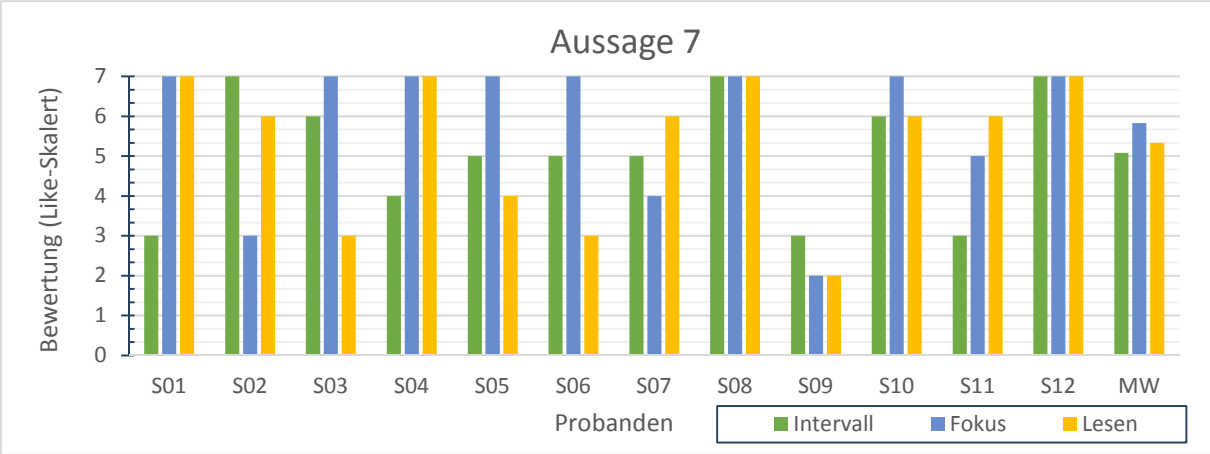


Diagramm 6.7: Likert-Skala-Bewertung der Probanden und Mittelwert zur Aussage: "Anhand der Screenshots konnte ich meine heutigen Aktivitäten gut nachvollziehen."

6.8 Relevanz der Screenshots

In Diagramm 6.8 fällt auf, dass der *Fokus*-Modus recht häufig schlecht bewertet ist. Dies schlägt sich auch im Mittelwert nieder. Zudem fällt auf, dass der *Lese*-Modus fast immer am besten bewertet ist. Der *Intervall*-Modus ist auf Platz zwei, obwohl dieser sehr naiv arbeitet.

In Abhängigkeit zum verwendeten Modus ergibt die Bewertung der Probanden, ob die gezeigten *Screenshots* als relevant empfunden werden, einen statistisch signifikanten Unterschied: $\chi^2(2) = 8{,}227 \mid p < 0{,}016$. Als Post-hoc-Analyse wurde ein Wilcoxon-Vorzeichen-Rang-Test durchgeführt und die Ergebnisse mit einer Bonferroni-Korrektur bereinigt. Dies ergibt ein Signifikanzniveau von $p < 0{,}017$.

Es gibt keine statistisch signifikanten Unterschiede zwischen dem *Intervall*-Modus und dem *Fokus*-Modus ($Z = -1{,}373 \mid p = 0{,}17$) als auch zwischen dem *Intervall*-Modus und dem *Lese*-Modus ($Z = -1{,}839 \mid p = 0{,}66$). Allerdings gibt es einen statistisch signifikanten Unterschied zwischen dem *Fokus*- und dem *Lese*-Modus ($Z = -2{,}862 \mid p = 0{,}004$). Die, im *Lese*-Modus gemachten Screenshots werden dabei als relevanter empfunden.

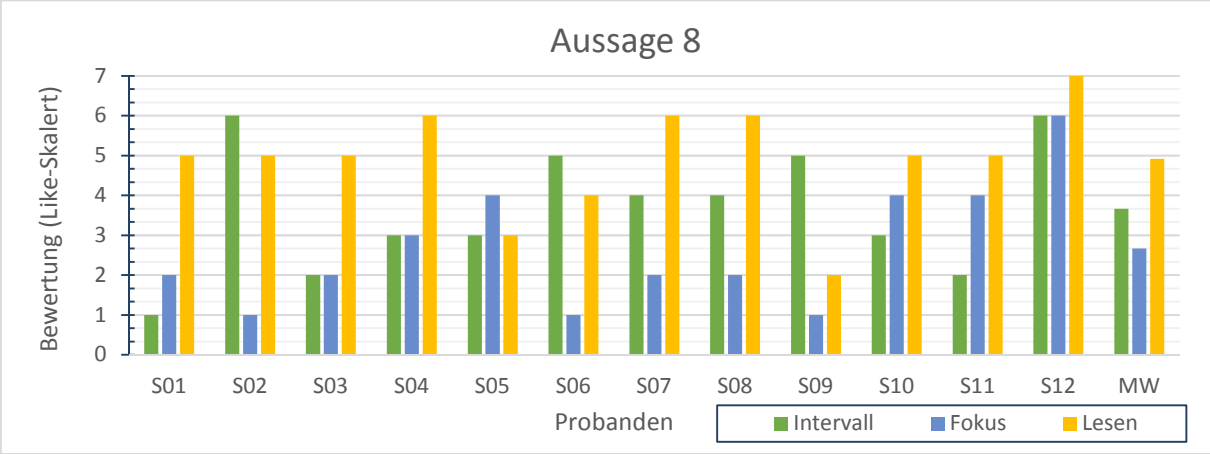


Diagramm 6.8: Likert-Skala-Bewertung der Probanden und Mittelwert zur Aussage: "Die gezeigten Screenshots empfand ich als relevant."

6.9 Rückschau archivieren

Die Rückschau wollen viele Teilnehmer nie oder nur selten archivieren. Lediglich Teilnehmer S12 gibt für jeden Tag an, die Rückschau gerne archivieren zu wollen. Dennoch wollen die Teilnehmer im Schnitt die Rückschau, die durch den *Lese*-Modus erzeugt wurde am ehesten archivieren. Doch der Mittelwert bestätigt auch den Trend, dass die Teilnehmer die Rückschau eher selten archivieren möchten, da im Mittel alle Werte unterhalb von vier Punkten liegen.

In Abhängigkeit zum verwendeten Modus ergibt die Bewertung der Probanden, ob die Rückschau archiviert werden soll, zwischen den Modi keinen statistisch signifikanten Unterschied ($\chi^2(2) = 2{,}579 \mid p = 0{,}275$).

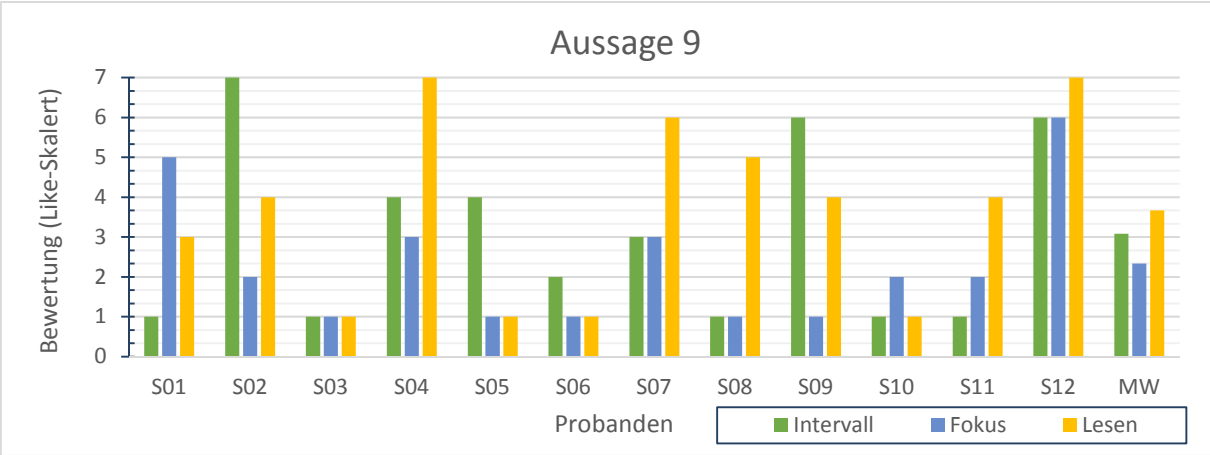


Diagramm 6.9: Likert-Skala-Bewertung der Probanden und Mittelwert zur Aussage: "Diese Rückschau würde ich gern auch für künftiges Nachschlagen archivieren."

6.10 Retrospektive

Die Modi sind größtenteils sehr positiv bewertet und es gibt kaum Bewertungen unter 4 Punkten. Im Mittel sind alle Modi mit mindestens 5 Punkten bewertet. Insgesamt scheint es zwischen den Modi keinen Unterschied zu geben, wenn es darum geht, sich seine Aktivitäten ins Gedächtnis zu rufen, wie der Mittelwert zeigt.

In Abhängigkeit zum verwendeten Modus ergibt die Bewertung der Probanden, ob die Rückschau bei der Retrospektive hilft, zwischen den Modi keinen statistisch signifikanten Unterschied ($\chi^2(2) = 0{,}424 \mid p = 0{,}809$).

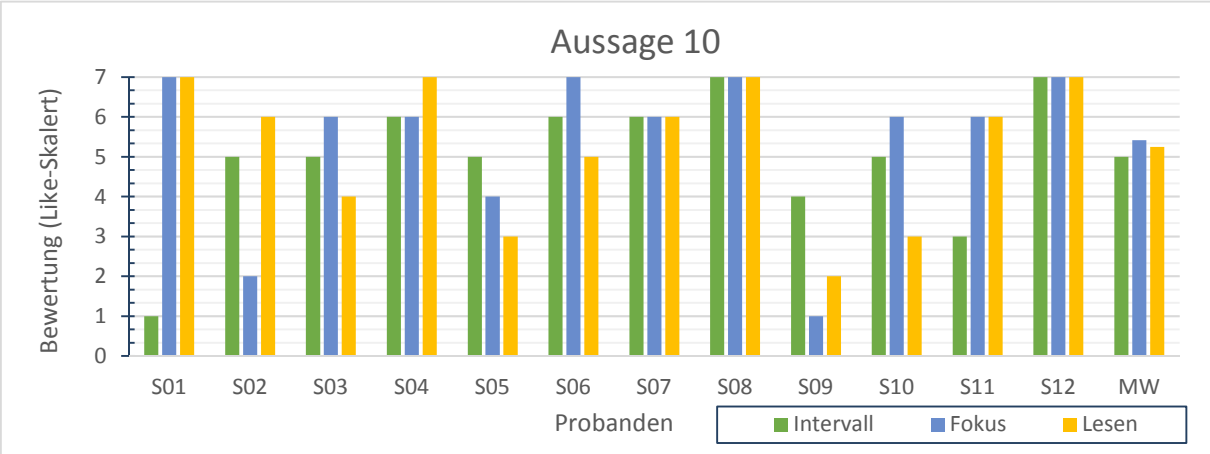


Diagramm 6.10: Likert-Skala-Bewertung der Probanden und Mittelwert zur Aussage: "Die Rückschau hat mir geholfen, mir meine heutigen Aktivitäten ins Gedächtnis zu rufen."

6.11 Ältere Daten erwünscht

Im Regelfall scheinen die Teilnehmer, dem Diagramm nach, keine Daten aus vorherigen Tagen in der Rückschau zu benötigen. Doch auch hier gehen die Meinungen teilweise auseinander.

In Abhängigkeit zum verwendeten Modus ergibt die Bewertung der Probanden, ob ältere Daten in der Rückschau erwünscht sind, zwischen den Modi keinen statistisch signifikanten Unterschied ($\chi^2(2) = 4{,}789 \mid p = 0{,}091$).

Auffällig ist aber, dass die Teilnehmer am ehesten beim *Lese*-Modus gerne Daten aus den vorherigen Tagen in der Rückschau sehen möchten. Vor allem beim Mittelwert fällt dies auf, wo sich die Bewertung des *Lese*-Modus von den beiden anderen Modi deutlich absetzt.

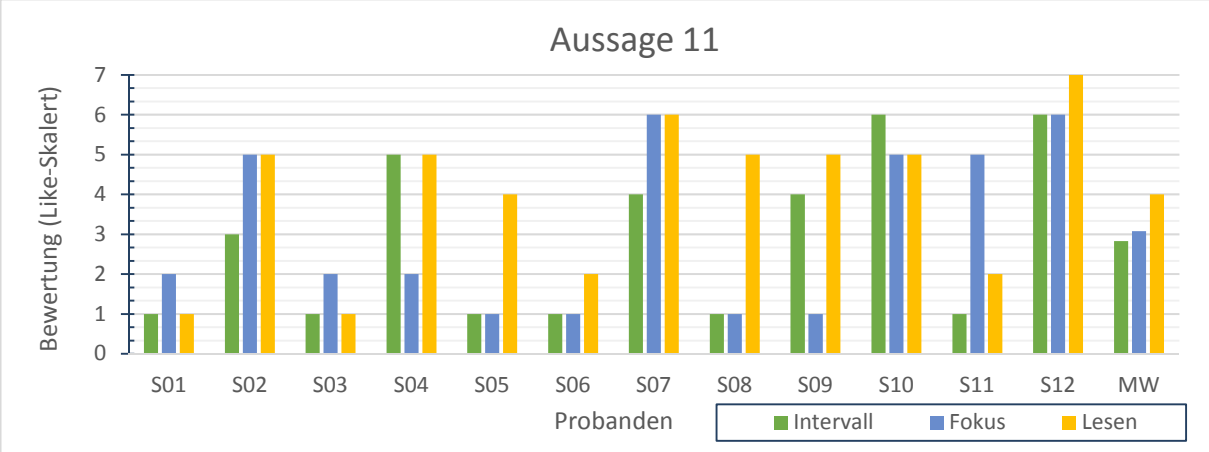


Diagramm 6.11: Likert-Skala-Bewertung der Probanden und Mittelwert zur Aussage: "Ich hätte gern auch Daten aus vorherigen Tagen in der Rückschau gehabt."

6.12 Bedenkliche Inhalte

Wie der Mittelwert zeigt, birgt die Rückschau, unabhängig vom genutzten Modus, bedenkliche Inhalte. Vier Teilnehmern (S04, S10, S11 und S12) haben bezüglich der Inhalte keinerlei Bedenken. Die Teilnehmer S02 und S07 sind dabei kritischer wohingegen der Rest der Befragten größtenteils angibt, dass die Rückschau bedenkliche Inhalte enthielt.

In Abhängigkeit zum verwendeten Modus ergibt die Bewertung der Probanden, ob die Rückschau bedenkliche Inhalte enthält, zwischen den Modi keinen statistisch signifikanten Unterschied ($\chi^2(2) = 0{,}24 \mid p = 0{,}887$).

Interessant sind hier die Bewertungen der Teilnehmer S05 und S08. Während bei Teilnehmer S05 die Bedenklichkeit der Inhalte beim *Lese*-Modus stark zunimmt, nimmt sie bei Teilnehmer S08 im gleichen Modus stark ab.

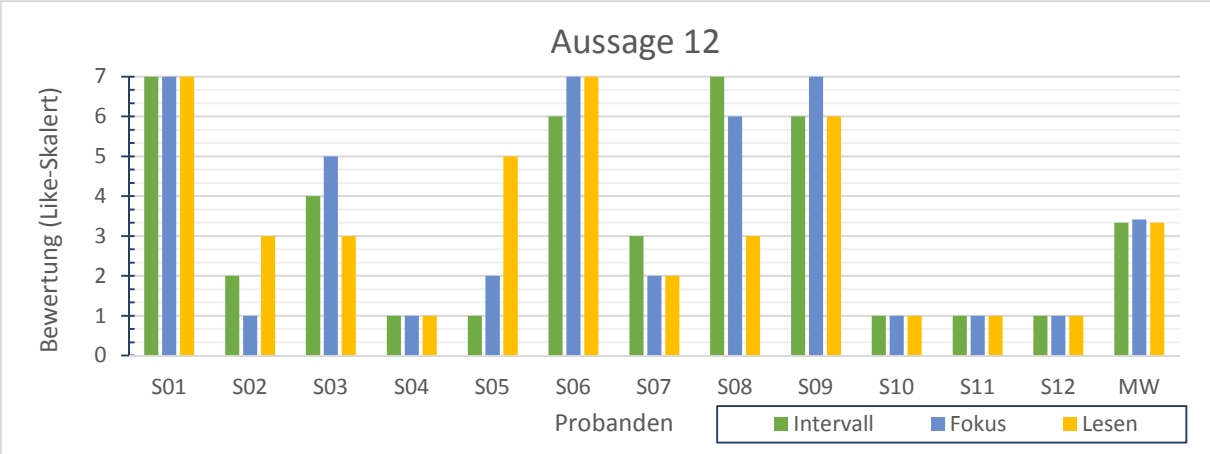


Diagramm 6.12: Likert-Skala-Bewertung der Probanden und Mittelwert zur Aussage: "Die Rückschau beinhaltete für mich bedenkliche Inhalte."

6.13 Inhalte nicht archivieren

Auffällig ist, dass im Mittelwert der *Intervall*-Modus und der *Fokus*-Modus gegenüber dem *Lese*-Modus leicht erhöht sind. Außerdem fällt auf, dass die Bewertungen in Diagramm 6.13 ähnlich auseinander gehen wie im vorhergehenden Diagramm 6.12. Dabei sind die Bewertungen der einzelnen Teilnehmer (bis auf Teilnehmer S03 und S07) entweder vollständig positiv oder negativ. Dabei bleibt die Bewertung von Teilnehmer S07 über alle Modi hinweg negativ, während Teilnehmer S03 nur beim *Fokus*-Modus der Aussage voll zustimmt und die anderen beiden Modi neutral bewertet.

In Abhängigkeit zum verwendeten Modus ergibt die Bewertung der Probanden, ob die Rückschau Daten enthielt die nicht gespeichert werden sollten, zwischen den Modi keinen statistisch signifikanten Unterschied ($\chi^2(2) = 2{,}333 \mid p = 0{,}311$).

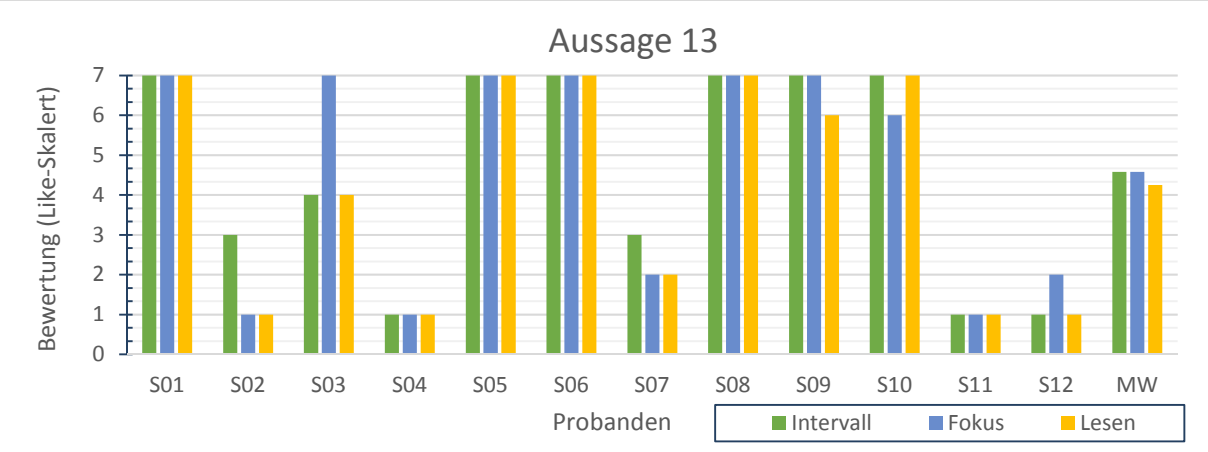


Diagramm 6.13: Likert-Skala-Bewertung der Probanden und Mittelwert zur Aussage: "Die Rückschau hat Daten beinhaltet, die ich lieber nicht archivieren möchte."

6.14 Suchfunktion genutzt

Lediglich Teilnehmer S12 gibt an, die Suchfunktion durchgängig genutzt zu haben. Ansonsten ist angegeben, dass die Suchfunktion nur in einzelnen Fällen genutzt wird.

In Abhängigkeit zum verwendeten Modus ergibt die Bewertung der Probanden, ob sie die Suchfunktion ausgiebig genutzt haben, zwischen den Modi keinen statistisch signifikanten Unterschied ($\chi^2(2) = 0{,}105 \mid p = 0{,}949$).

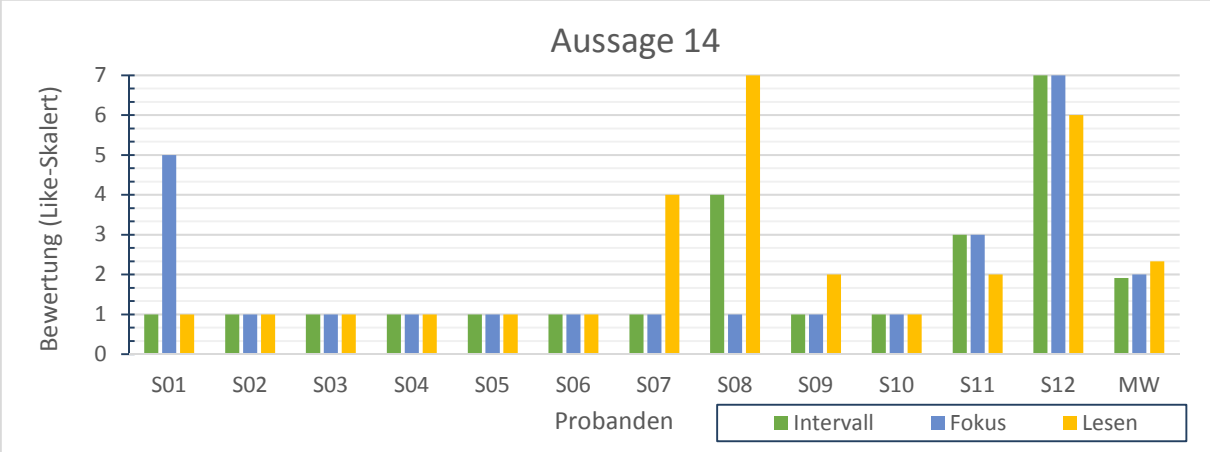
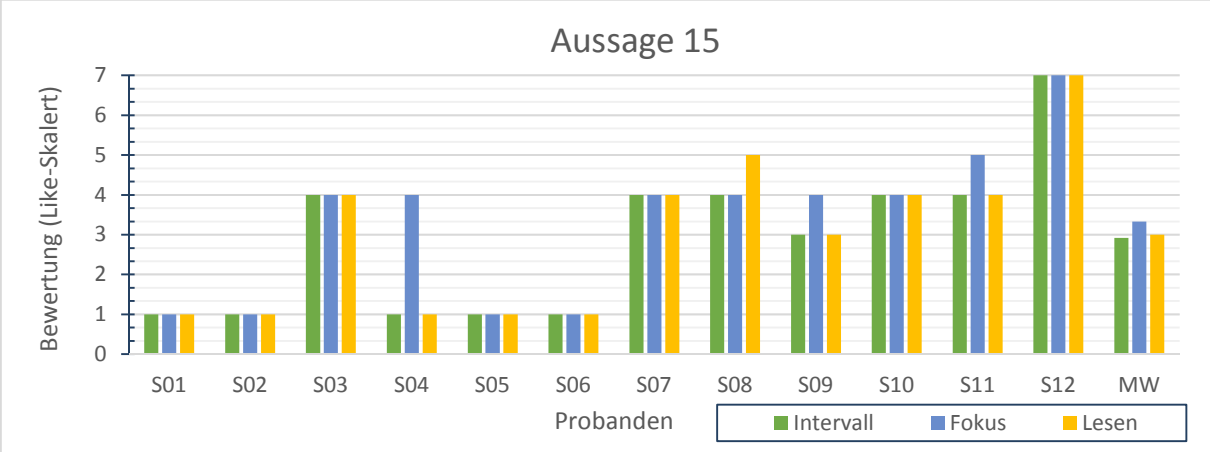


Diagramm 6.14: Likert-Skala-Bewertung der Probanden und Mittelwert zur Aussage: "Ich habe die Such- oder Filter-funktion ausgiebig genutzt."

6.15 Hilfreiche Suchfunktion

Nur wenige Teilnehmer geben an, dass die Suchfunktion hilfreich ist. Lediglich Teilnehmer S12 bewertet die Suchfunktion als hilfreich. Häufig wird die Suchfunktion jedoch neutral bewertet, weswegen der Mittelwert eine negative Tendenz hat.

In Abhängigkeit zum verwendeten Modus ergibt die Bewertung der Probanden, ob sie die Suchfunktion hilfreich fanden, zwischen den Modi keinen statistisch signifikanten Unterschied ($\chi^2(2) = 3{,}5 \mid p = 0{,}174$).



6.16 Resultate der Suchfunktion

Da viele Nutzer die Suchfunktion nicht nutzen, wie Diagramm 6.14 zeigt, sind hier viele neutrale Bewertungen enthalten. Die Restlichen Bewertungen sind größtenteils negativ, wobei die Resultate der Suchfunktion teilweise auch positiv bewertet sind.

In Abhängigkeit zum verwendeten Modus ergibt die Bewertung der Probanden, ob die Suchfunktion nützliche Resultate produziert hat, zwischen den Modi keinen statistisch signifikanten Unterschied ($\chi^2(2) = 2{,}0 \mid p = 0{,}368$).

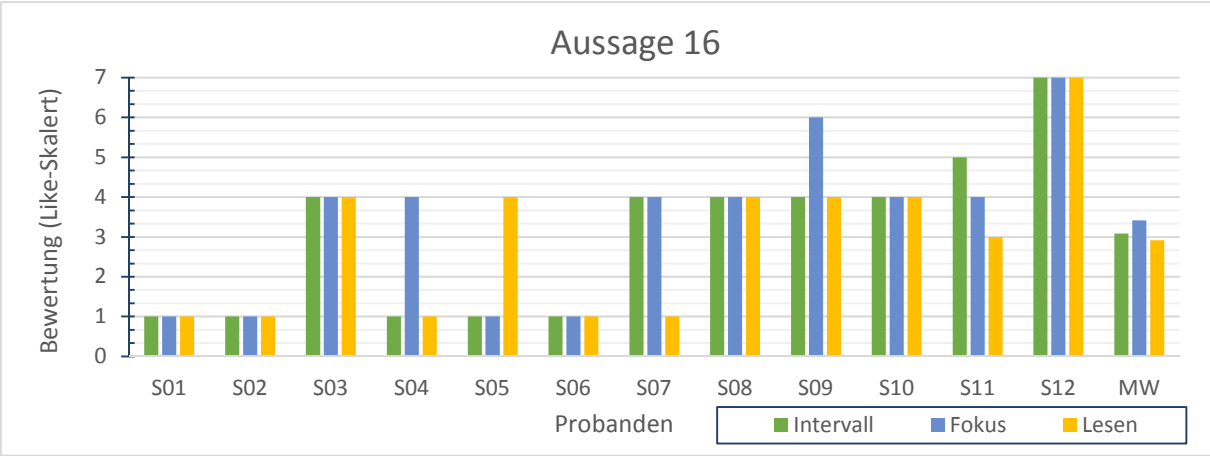


Diagramm 6.16: Likert-Skala-Bewertung der Probanden und Mittelwert zur Aussage: "Die Suchfunktion hat nützliche Resultate produziert."

6.17 Performanz Analyse

Aus Tabelle 6.1 geht hervor, dass der Algorithmus zur Lesedetektion insgesamt acht Mal das Lesen erkannt hat. Dabei hat er bei vier Probanden das Überfliegen und bei einem Probanden das Suchen nach einem Schlüsselwort als Lesebewegung detektiert. Zudem hat der Algorithmus bei den Teilnehmern S03, S04, S09 und S10 bei der Aufgabe eine Textpassage zu lesen, keine Lesebewegung detektiert.

Aufgabe	S01	S02	S03	S04	S05	S06	S07	S08	S09	S10	S11	S12
Lesen	✓	✓	✗	✗	✓	✓	✓	✓	✗	✗	✓	✓
Überfliegen	✗	✗	✓	✓	✗	✓	✗	✗	✓	✗	✗	✗
Suchen	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗

Tabelle 6.1: Genauigkeit der Lesedetektion

7 Diskussion

Im Folgenden werden die Ergebnisse aus Kapitel 6 näher betrachtet und miteinander verglichen, um daraus Rückschlüsse zu ziehen.

7.1 Effektiv gearbeitet

Wie Diagramm 6.1 zeigt bleibt die Effektivität der Probanden über die verschiedenen Modi hinweg stabil. Lediglich bei Teilnehmer S10 fällt ein starker Einbruch beim *Lese*-Modus auf. Da dieser Einbruch die Ausnahme darstellt und alle anderen Probanden eine annähernd ähnliche Produktivität angeben, scheint der Modus für die Effektivität keine Rolle zu spielen.

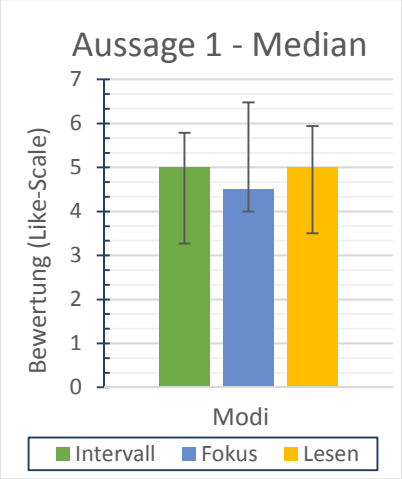
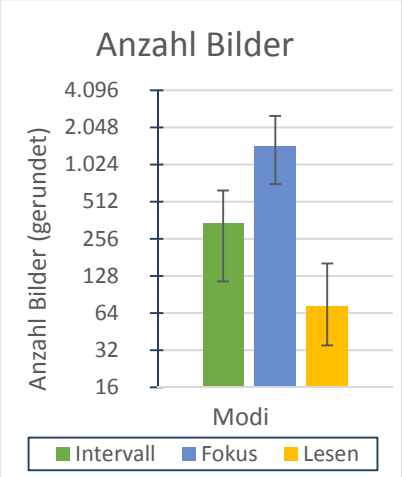


Diagramm 7.1: Median-Werte der Effektivitätsbewertung

7.2 Angemessener Umfang

Diagramm 7.2 zeigt, dass die Anzahl der Bilder zwischen den verschiedenen Modi stark variiert. Während im *Fokus*-Modus durchschnittlich bei 1448 Bildern liegt sind es beim *Lese*-Modus nur 73 Bilder, was etwa 5% der Menge des *Fokus*-Modus entspricht. Diagramm 7.3 zeigt, dass der *Lese*-Modus im Schnitt deutlich besser bewertet ist, als der *Fokus*-Modus.

Der Vergleich von Diagramm 7.2 und Diagramm 7.3 zeigt, dass die Bewertung des Umfangs mit der Zahl der Bilder korreliert: Je weniger Bilder in der Rückschau enthalten sind, desto besser ist die Rückschau bewertet.



Anzahl an Bildern pro Modus

Bei den Bewertungen fällt zudem auf, dass Teilnehmer S01 den Umfang der Rückschau beim *Intervall*-Modus weniger angemessen als den des *Fokus*-Modus empfindet. Dies lässt sich dadurch erklären, dass sich in diesem Fall die Zahl der Bilder im *Fokus*-Modus im Vergleich zum *Intervall*-Modus verringert hat (vgl. Diagramm 7.4), obwohl der *Fokus*-Modus üblicherweise am meisten Bilder produziert. Dieser Teilnehmer hat allerdings sieben Stunden mit dem *Intervall*-Modus und lediglich 1,5 Stunden mit dem *Fokus*-Modus gearbeitet. Zudem werden beim *Fokus*-Modus deutlich weniger Anwendungen über den Tag genutzt (14 statt 51), was die geringe Anzahl an Bildern in der Rückschau erklärt.

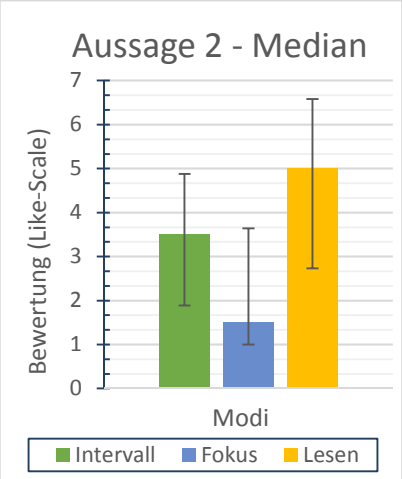


Diagramm 7.3: Median-Werte der Umfangsbewertung

Zudem bewerten die Teilnehmer S03, S05, S06 und S11 den *Lese*-Modus um einen Punkt schlechter als den *Intervall*-Modus. Dies begründen die Teilnehmer entweder im Freitext-Feld der Umfrage oder im abschließenden Interview so, dass der *Lese*-Modus zwar gut ist, jedoch insgesamt zu wenig *Screenshots* macht.

Somit gilt zunächst, dass weniger Bilder als angemessener empfunden werden, diese Aussage aber nur bis zu einer bestimmten Anzahl an Bildern korrekt ist. Ab dieser Anzahl ist der Umfang der Bilder für eine Rückschau offenbar zu gering.

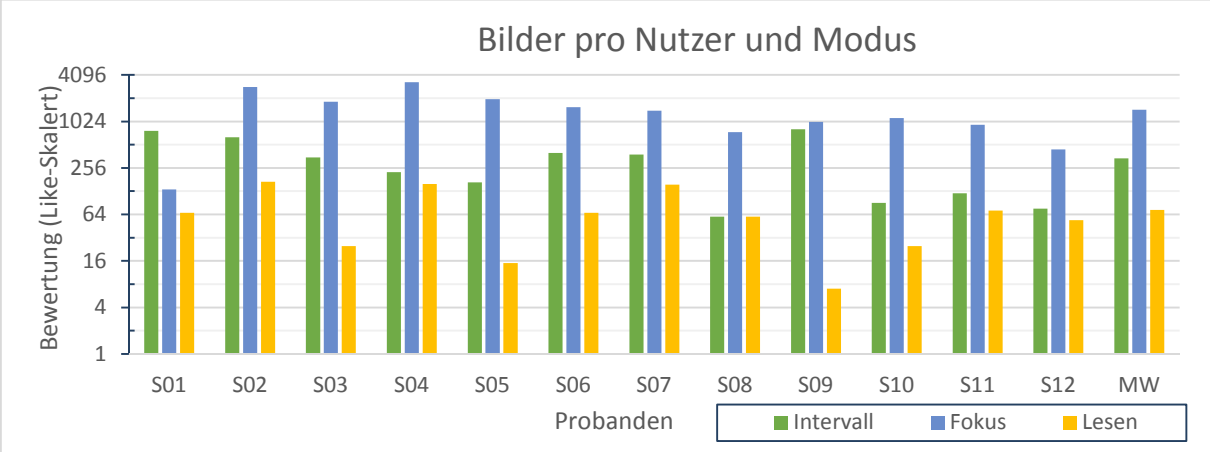


Diagramm 7.4: Bilder pro Nutzer und Modus

7.3 Angemessener Zeitaufwand

Beim Vergleich des Mittelwertes von Diagramm 6.2 mit dem von Diagramm 6.3 fällt auf, dass diese sich ähnlich verhalten. Auch ein Vergleich der Mediane (Diagramm 7.3, Seite 56 und Diagramm 7.6) zeigt, dass die Bewertungen sich ähneln. Dies legt nahe, dass eine Beziehung zwischen der Einschätzung der Angemessenheit des Umfangs und der Einschätzung der Angemessenheit der benötigten Zeit besteht. Dabei schneidet der *Lesen*-Modus in beiden Fällen am Besten ab.

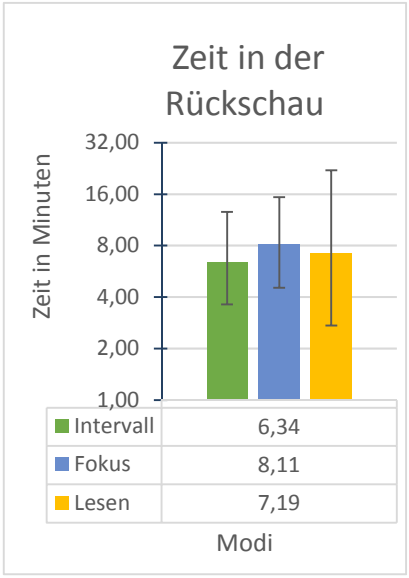


Diagramm 7.5: Durchschnittliche Zeit für die Rückschau pro Modus

Interessant dabei ist, dass die Teilnehmer im Schnitt ähnlich viel Zeit für die Rückschau des *Fokus*-Modus wie für den *Lese*-Modus aufwenden, wie Diagramm 7.5 verdeutlicht. Am wenigsten Zeit bringen die Teilnehmer für den *Intervall*-Modus auf, der bei den Bewertungen den zweiten Platz einnimmt.

Diagramm 7.5 zeigt, dass die Teilnehmer für die Rückschau, unabhängig vom Modus und der Zahl der *Screenshots*, die dieser abgespeichert hat, gleich viel Zeit benötigen. Diagramm 7.7 zeigt, dass die Teilnehmer in der Rückschau des *Lese*-Modus erheblich mehr Zeit pro Bild aufwenden.

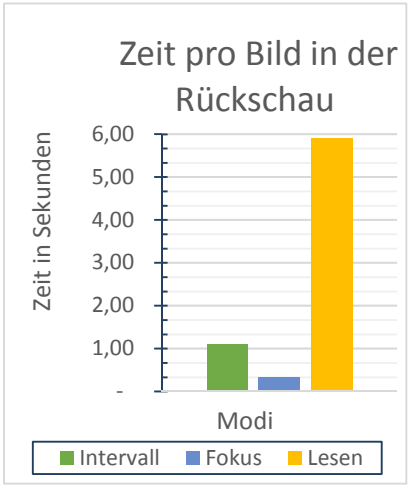


Diagramm 7.7: Durchschnittliche Zeit pro Bild und Modus

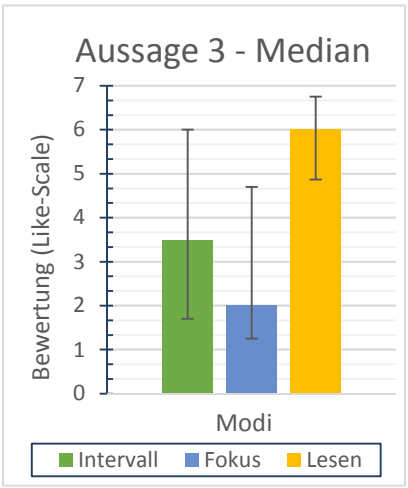


Diagramm 7.6: Medianwerte der Bewertung des Zeitaufwandes

7.4 Nützliche Rückschau

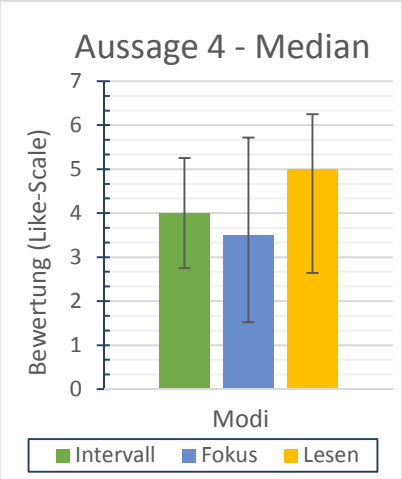
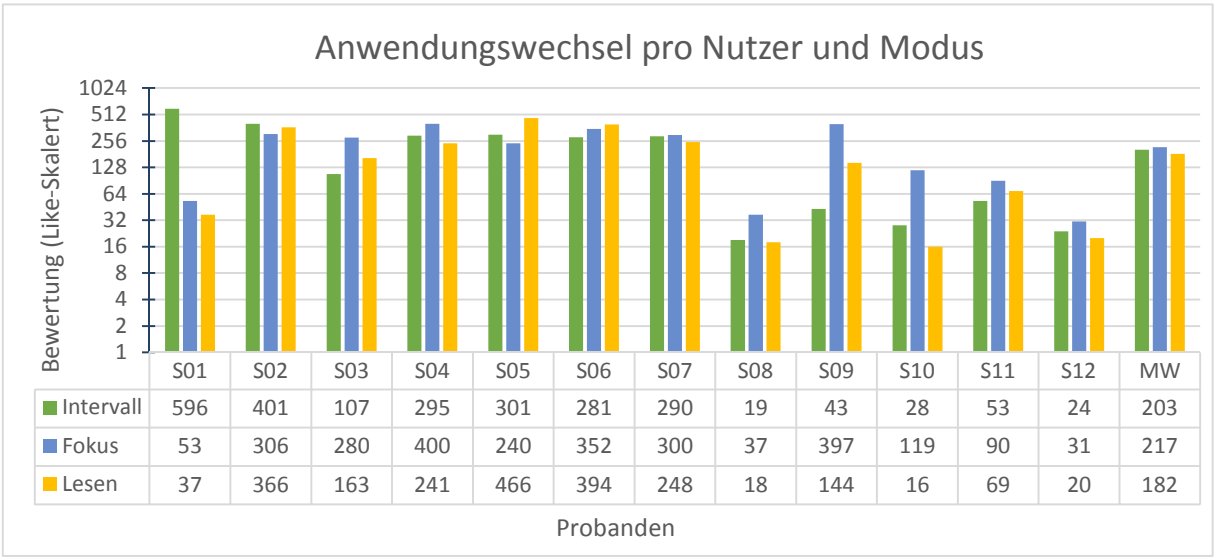


Diagramm 7.8: Medianwerte der Bewertung der Nützlichkeit

Die Werte in Diagramm 6.4 variieren stark. Hinzu kommt noch, dass die Nützlichkeit ein stark subjektiver Wert ist. Dieser lässt sich nur schwer zwischen den Einzelnen Teilnehmern vergleichen, da die Teilnehmer bei der Bewertung unterschiedliche eigene Maßstäbe ansetzen.

Insgesamt zeichnet sich jedoch sowohl beim Mittelwert (vgl. Diagramm 6.4, Seite 43) und beim Median in Diagramm 7.8 der schwache Trend ab, dass die Nützlichkeit der Rückschau ähnlich verhält wie die Bewertungen des Umfangs und des Zeitaufwandes.

Da der *Intervall*-Modus teilweise und der *Fokus*-Modus vollständig darauf basiert, wann welche Anwendung aktiv ist lässt sich möglicherweise eine Verbindung zwischen der Nützlichkeit der Rückschau und der Zahl der Anwendungswechsel, dargestellt in Diagramm 7.9, finden. Dieses zeigt zwischen den einzelnen Modi insgesamt kaum einen Unterschied.



	S01	S02	S03	S04	S05	S06	S07	S08	S09	S10	S11	S12	MW
Intervall	596	401	107	295	301	281	290	19	43	28	53	24	203
Fokus	53	306	280	400	240	352	300	37	397	119	90	31	217
Lesen	37	366	163	241	466	394	248	18	144	16	69	20	182

Diagramm 7.9: Anwendungswechsel pro Nutzer und Modus während der Studie

Bei Teilnehmer S01 lässt sich die bessere Bewertung für den *Fokus*-Modus wahrscheinlich darauf zurückführen, dass, verglichen mit dem *Intervall*-Modus signifikant weniger Anwendungswechsel stattfinden. Das führt, wie Diagramm 7.4 zeigt, in diesem Fall zu deutlich weniger Bilder in der Rückschau.

Die Teilnehmer S05 und S10 bewerten den *Fokus*-Modus, Teilnehmer S10 zudem den *Lese*-Modus deutlich schlechter als den *Intervall*-Modus, der von beiden mit 4 Punkten neutral bewertet ist. Bei Teilnehmer S05 ähnelt sich die Zahl der Anwendungswechsel zwischen dem *Intervall*-Modus und dem *Fokus*-Modus, während sich bei Teilnehmer S10 die Anwendungswechsel um Faktor vier unterscheiden. In beiden Fällen werden im *Fokus*-Modus deutlich mehr *Screenshots* abgespeichert als im *Intervall*-Modus. Teilnehmer S11 hingegen ist mit der Rückschau des *Lese*-Modus am zufriedensten, obwohl er dabei die wenigsten Anwendungswechsel durchführt. Es lässt sich in diesem Fall also kein allgemeiner Zusammenhang zwischen den Anwendungswechseln und der Bewertung der Nützlichkeit der Rückschau herleiten.

Die Anzahl der aufgenommenen *Screenshots* scheint in keinem allgemeinen Zusammenhang mit der Nützlichkeit der Rückschau zu stehen. Denn obwohl bei Teilnehmer S09 in beiden Modi annähernd gleich viele Bilder in der Rückschau enthalten sind, bewertet dieser die Rückschau des *Intervall*-Modus deutlich besser.

7.5 Aktivitäten gut wiedergegeben

Nach Diagramm 6.5 (Seite 44) zu urteilen schneiden alle Modi sehr ähnlich ab, wobei der *Intervall*-Modus dabei geringfügig besser bewertet ist. Der Median hingegen stellt den *Lese*-Modus als denjenigen Modus dar, dessen Rückschau am ehesten die wichtigsten Aktionen des Tages enthält.

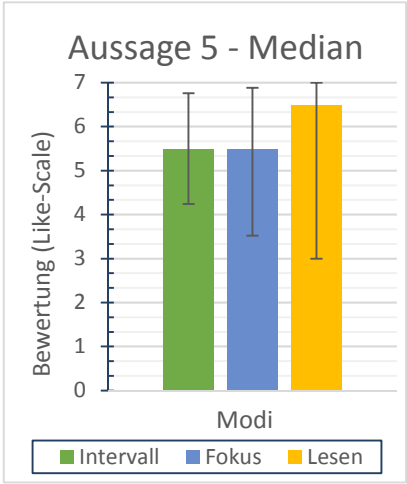


Diagramm 7.10: Medianwerte der Bewertung der Wiedergabe der Aktionen

Bei sechs der Teilnehmer unterscheidet sich die Bewertung zwischen den einzelnen Modi nur um einen Punkt, während bei den anderen Teilnehmern die Bewertung um bis zu 5 Punkte schwankt. Die Teilnehmer S03, S05 und S06 bewerten dabei den *Fokus*-Modus am besten, während Teilnehmer S09 den *Intervall*-Modus, Teilnehmer S11 den *Lese*-Modus und Teilnehmer S08 sogar beide Modi am höchsten bewertet.

Insgesamt scheinen alle drei Modi zufriedenstellend zu sein, da sowohl der Mittelwert als auch der Median bei allen drei Modi über fünf Punkte liegt. Weitere Rückschlüsse lassen allerdings aus den Bewertungen nicht ableiten.

7.6 Wichtigste Aktivitäten enthalten

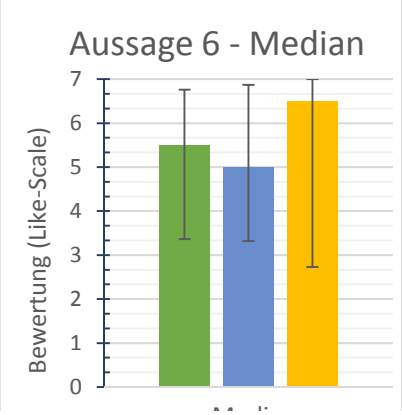


Diagramm 7.11: Medianwerte der Bewertung der Vollständigkeit der Rückschau

Lediglich der *Lese*-Modus ist teilweise mit einem Punkt bewertet, während die Bewertungen der anderen Modi stets über Eins liegen. Eine Erklärung ist, dass der *Lese*-Modus zu wenig Bilder erzeugt, weil nur gelesene Texte eingesammelt werden. Dadurch lassen sich möglicherweise nicht alle relevanten Tagesaktivitäten festhalten.

Wie aus Diagramm 7.4 hervor geht besteht die Rückschau bei Teilnehmer S05 im *Fokus*-Modus aus 2000 Bilder, bei Teilnehmer S09 hingegen aus 1000 Bilder. Bei einem Arbeitstag mit 8 Stunden entspricht dies einem Bild pro 15 bzw. 30 Sekunden. Bei einem Vergleich von Diagramm 7.4 mit Diagramm 6.6 wird ersichtlich, dass die Bewertung der Rückschau besser ausfällt, wenn mehr Bilder in der Rückschau enthalten sind. Doch auch mit durchschnittlich 15 bzw. 30 Bildern pro Sekunde bewerten beide Probanden die Aussage neutral. Die Teilnehmer S02, S08 und S11 hingegen bewerten den *Lese*-Modus mit sehr deutlichem Unterschied zu den beiden anderen Modi am besten. Dies steht in starkem Kontrast zur Bewertung der Teilnehmer S05 und S09, entspricht aber dem Trend, den Diagramm 7.11 anzeigt.

Insgesamt scheinen die Modi gute Ergebnisse zu liefern, da der Großteil der Bewertungen positiv ausfällt.

7.7 Aktivitäten nachvollziehbar

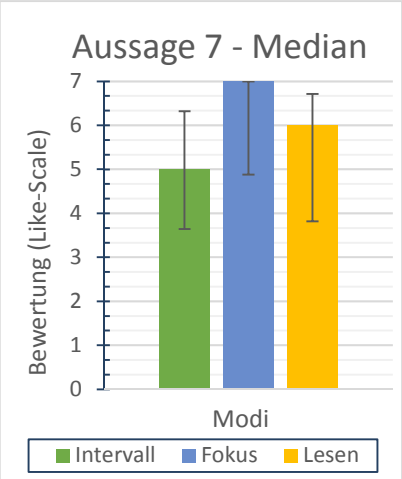


Diagramm 7.12: Medianwerte der Bewertung der Nachvollziehbarkeit der Aktionen

Wie sowohl der Mittelwert (vgl. Diagramm 6.7, Seite 46) als auch der Median (vgl. Diagramm 7.12) zeigen, lassen sich mit allen Modi die täglichen Aktivitäten gut nachvollziehen. Unter den einzelnen Bewertungen finden sich nur wenige, die unter die vier Punkte Marke fallen. Die meisten Teilnehmer bewerten mindestens einen Modus mit der vollen Punktzahl. Die Teilnehmer S07 und S11 bewerten immerhin den *Lese*-Modus mit sechs Punkten.

Nur Teilnehmer S09 kann mit keinem der drei Modi seine täglichen Aktivitäten nachvollziehen. Aus der Anzahl der Bilder (vgl. Diagramm 7.4, Seite 56) der drei Modi lässt sich kein Zusammenhang ableiten. Lediglich die Kommentare des Teilnehmers geben Aufschluss über die schlechte Bewertung. Der *Intervall*-Modus und der *Fokus*-Modus legen zu insgesamt viele *Screenshots* an, wobei auch viele ähnliche *Screenshots* enthalten sind. Während beim *Intervall*-Modus die redundanten Bilder zu der schlechten Bewertung führen, ist es im *Fokus*-Modus die zu große Menge an Bildern, durch die der Teilnehmer seinen Tagesablauf nicht mehr nachvollziehen kann. Der *Lese*-Modus hingegen enthält dem Teilnehmer zu wenig *Screenshots* (vgl. Diagramm 7.4, Seite 56), wodurch sich die Aktivitäten des Tages ebenfalls nicht nachvollziehen lassen.

Sowohl im Mittelwert (Diagramm 6.7, Seite 46) als auch in Diagramm 7.12 ist der *Fokus*-Modus, dicht gefolgt vom *Lese*-Modus, am besten bewertet. Obwohl die Unterschiede groß genug sind, um daraus Rückschlüsse ziehen zu können, drängt sich zumindest die These auf, dass der *Fokus*-Modus wohl deswegen am besten bewertet ist, weil er auch die meisten *Screenshots* angelegt, weswegen sich die Aktionen am besten nachvollziehen lassen. Ebenfalls möglich ist, dass der *Lese*-Modus, obwohl er am wenigsten Bilder speichert, dadurch überzeugt, dass seine Bilder am relevantesten sind, wie Diagramm 6.7 (Seite 46) zeigt. Diese Thesen müssten jedoch in einer ausführlicheren Studie bewiesen werden, da sie auf Basis der zur Verfügung stehenden Datenmenge noch sehr spekulativ sind.

7.8 Relevanz der Screenshots

Nicht nur der Mittelwert in Diagramm 6.8 (Seite 47) sondern auch der Median in Diagramm 7.13 zeigen, dass der *Fokus*-Modus insgesamt am schlechtesten und der *Lese*-Modus am besten bewertet ist. Somit scheint der *Lese*-Modus tatsächlich die relevantesten Screenshots abzuspeichern.

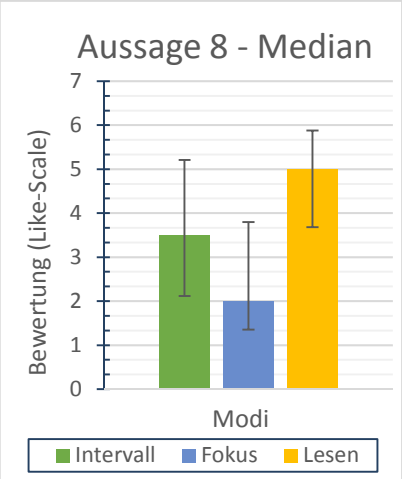


Diagramm 7.13: Medianwerte der Bewertung der Relevanz der Screenshots

Ein Grund für die schlechte Bewertung des *Fokus*-Modus könnte sein, dass häufig ein *Screenshot* von der *Taskleiste* gemacht wird, da viele Windows-Nutzer ihre aktive Anwendung über die *Taskleiste* wechseln und diese dabei zwangsläufig ansehen. Ebenfalls könnten die vielen doppelten Bilder ein Grund für die schlechte Bewertung sein, weswegen auch der *Intervall*-Modus deutlich schlechter als der *Lese*-Modus bewertet ist.

7.9 Rückschau archivieren

Diese schlechten Bewertungen können daher rühren, dass die meisten Teilnehmer zum Zeitpunkt der Studie unter 35 Jahren alt waren. Lediglich Teilnehmer S12 war mit 65 Jahren schon im Rentenalter. So wäre es möglich, dass ältere und eventuell vergesslichere Menschen darum die Rückschau eher abspeichern möchten als jüngere Menschen.

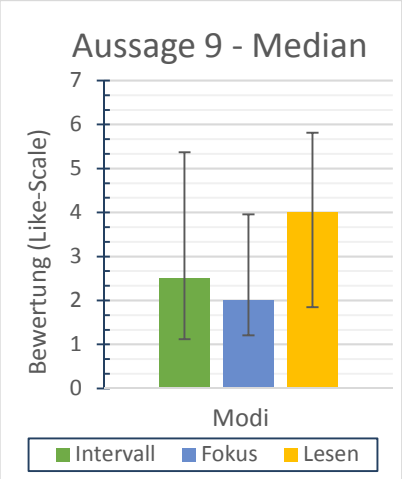


Diagramm 7.14: Medianwerte der Bewertung ob die Rückschau archiviert werden sollte

Die einzelnen positiven Bewertungen der anderen Teilnehmer könnten sich so erklären lassen, dass die Rückschau nicht immer Daten enthält, bei denen es sinnvoll ist, sie abzuspeichern. Welche Daten die Rückschau enthält hängt nicht nur vom Modus ab, der die Daten einsammelt sondern von dem, was der Benutzer am PC macht. Liest der Benutzer beispielsweise den ganzen Tag E-Mails von Kunden, macht ist es wahrscheinlich nicht sinnvoll, dies in der Rückschau erneut abzuspeichern, da der Schriftverkehr bereits im E-Mail-Programm archiviert ist. Wenn jedoch ein Student für eine Prüfung lernt und dafür viele verschiedene Publikationen liest, kann es durchaus Sinn ergeben, die Rückschau abzuspeichern. Viele der Studenten waren zum Zeitpunkt der Studie in ihrer Prüfungsphase. Im Abschlussinterview geben diese häufig an, dass das Textarchiv sie gut beim Lernen unterstützt hat.

Dem Median nach zu urteilen archivieren die Teilnehmer am ehesten die Rückschau, die vom *Lese*-Modus erzeugt wird. Diesen Trend bestätigt auch der Mittelwert. Somit könnte auch ein Zusammenhang zwischen der Relevanz der Bilder und dem Wunsch, die Rückschau abzuspeichern bestehen.

7.10 Retrospektive

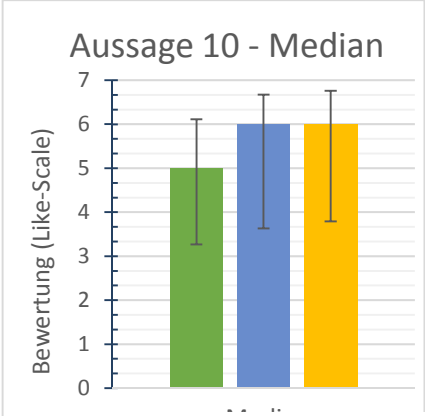


Diagramm 7.15: Medianwerte der Bewertung der Nützlichkeit der Rückschau

Die größtenteils positiven Bewertungen über alle Modi hinweg zeigen, dass eine Retrospektive der Aktivitäten des Tages hilft, sich diese ins Gedächtnis zu rufen. Vor allem die Berufstätigen Teilnehmer finden die Rückschau sehr hilfreich, um am Ende des Tages ihre Arbeitszeiten nachvollziehen und buchen zu können.

Dass der *Fokus*-Modus insgesamt am besten abgeschlossen hat ist nur wenig verwunderlich (vgl. Diagramm 7.4). Demnach speichert der *Fokus*-Modus durchschnittlich knapp 1450 *Screenshots* pro Tag. Bei einem Arbeitstag mit 8 Stunden speichert dieser Modus durchschnittlich alle 20 Sekunden einen neuen *Screenshot*. Der *Intervall*-Modus speichert bei einem 8-Stunden Tag immerhin noch alle 1,5 Minuten einen *Screenshot*, während der *Lese*-Modus nur alle 6,5 Minuten einen *Screenshot* speichert. Dennoch wird der *Lese*-Modus ähnlich gut bewertet wie die beiden anderen Modi.

Obwohl alle Teilnehmer unterschiedlich viel Zeit am PC verbringen scheint dies auf die Qualität der Rückschau kaum einen Einfluss zu haben. Im Vergleich von Diagramm 6.1 (Seite 40) mit Diagramm 6.10 (Seite 49) lassen sich keine Parallelen erkennen, weswegen die Effektivität offenbar ebenfalls keinen Einfluss darauf hat, ob eine Rückschau hilfreich ist oder nicht. Auch wenn die Zahl der Bilder pro Modus (vgl. Diagramm 7.2, Seite 55) mit den Ergebnissen verglichen werden, lassen sich keine Zusammenhänge erkennen, weswegen sich anzunehmen ist, dass für die Rückschau nicht zwangsläufig viele Bilder benötigt werden.

Es fällt jedoch auf, dass sich die Relevanz der Bilder des *Lese*-Modus von der Bewertung der Rückschau in jedem Fall um höchstens 2 Punkte unterscheidet, wobei dabei die Relevanz der Bilder lediglich in zwei Fällen (S03 und S10) besser bewertet ist als die Nützlichkeit der Rückschau.

7.11 Ältere Daten erwünscht

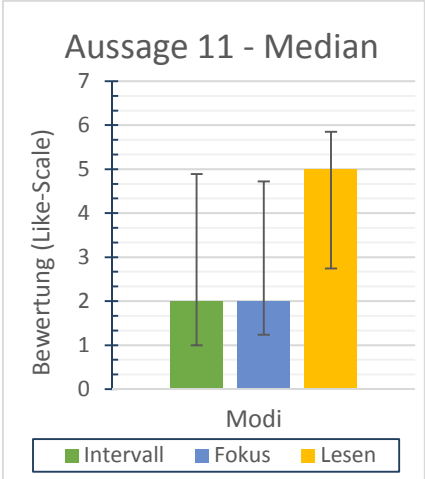


Diagramm 7.16: Medianwerte der Bewertung ob die Rückschau ältere Daten hätte enthalten sollen

Auch in Diagramm 7.16 setzt sich die Bewertung im *Lese*-Modus deutlich von den anderen beiden Modi ab. Eine mögliche Erklärung dafür ist, dass dieser Modus mit durchschnittlich nur 73 Bildern pro Tag (vgl. Diagramm 7.2, Seite 55) so wenig Bilder enthält, dass es sinnvoll ist, die Rückschau durch Einträge der letzten Tag zu ergänzen.

Besonders sinnvoll ist dies, wenn der Benutzer in den letzten Tagen ähnliche Aufgaben erledigt hat. Doch nicht nur die Entwickler geben an, dass sie gerne Daten aus den vorherigen Tagen in der Rückschau hätten, sondern auch die anderen Teilnehmer stimmen dieser Aussage größtenteils zu. Es ist daher sinnvoll, dem Benutzer die Möglichkeit zu geben, die Rückschau um Daten zu erweitern. Allerdings darf die Rückschau nicht standardmäßig um Daten von vorherigen Tagen erweitert werden, da die vielen negativen Bewertungen zeigen, dass dies eher selten gewünscht wird.

7.12 Bedenkliche Inhalte

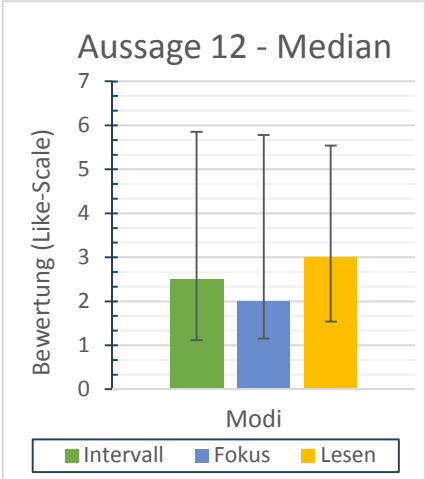


Diagramm 7.17: Medianwerte der Bewertung der Bedenklichkeit der Inhalte

Im Durchschnitt sind die Bewertungen eher negativ. Das heißt die Inhalte der Rückschau sind für die Benutzer tendenziell unbedenklich. Dies zeigt auch Diagramm 7.17. Dennoch geben 50% der Teilnehmer mindestens einmal an, dass die Rückschau Inhalte enthält, welche die Teilnehmer als (stark) bedenklich einstufen.

Es ist jedoch unklar, was der einzelne Teilnehmer jeweils unter „bedenklich“ versteht, da die Bedenklichkeit der Inhalte von jedem etwas anders wahrgenommen wird. Ein Grund für die positiven Bewertungen ist sein, dass die meisten versierten PC-Nutzer wissen, dass prinzipiell alle Daten gestohlen werden können. Eventuell werden die abgespeicherten *Screenshots* deswegen als bedenklich empfunden, weil sie private Informationen enthalten, die potentiell auch gestohlen und missbraucht werden können. Manche Nutzer stehen darum der Speicherung der Daten immer kritisch gegenüber, während andere Nutzer damit kein Problem haben.

Insgesamt zeigt die Bewertung, dass alle drei Modi das gleiche Potential haben, bedenkliche Inhalte in der Rückschau wiederzugeben, sofern der Anwender entsprechende Inhalte am PC nutzt.

7.13 Inhalte nicht archivieren

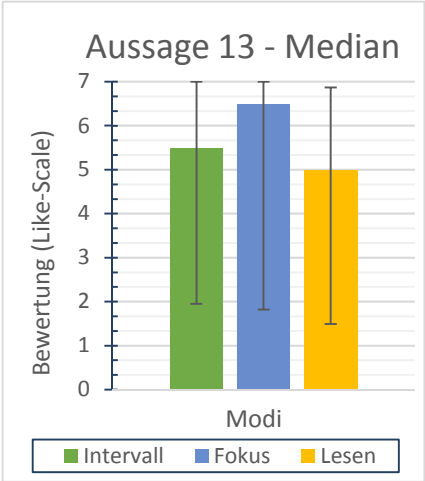


Diagramm 7.18: Medianwerte der Bewertung die Inhalte nicht abspeichern zu wollen

Der Vergleich von Diagramm 6.12 (Seite 51) mit Diagramm 6.13 (Seite 52) zeigt, dass die Benutzer immer dann Daten aus der Rückschau nicht archivieren wollen, wenn die Rückschau bedenkliche Inhalte enthält. Da die Diagramme sich aber nicht ganz decken, scheint dies nicht der einzige Grund dafür zu sein, Inhalte nicht archivieren zu wollen.

Wenn die Rückschau bedenkliche Inhalte enthält, möchte der Teilnehmer meist nicht, dass die Rückschau archiviert wird. Teilnehmer S10 gibt in allen Fällen an, die Rückschau nicht archiviert zu wollen, obwohl diese keine für ihn bedenkliche Inhalte enthält. Daraus lässt sich ein weiterer Grund ableiten, warum Daten der Rückschau nicht archiviert werden sollen: Diagramm 6.8 (Seite 47) zeigt, dass der *Lese*-Modus am ehesten die relevanten Informationen sammelt. Über den *Intervall*-Modus und den *Fokus*-Modus wird im abschließenden Interview sehr häufig geäußert, dass diese zu viele redundante *Screenshots* abspeichern. Demnach ist die Redundanz ein weiterer Grund, weswegen die Teilnehmer die Rückschau nicht archivieren möchten.

Im Vergleich mit Diagramm 6.9 (Seite 48) fällt außerdem auf, dass die Teilnehmer S03, S05, S06, S08 und S10 die Inhalte der Rückschau nicht archivieren möchten. Auch im Vergleich mit Diagramm 6.12 (Seite 51) lassen sich viele Gemeinsamkeiten finden. Da der Prototyp keine Funktion bietet, die entsprechenden Einträge vor dem Speichern auszusortieren, wollen die Teilnehmer vermutlich deswegen die gesamte Rückschau nicht archivieren.

Insgesamt zeigen der Mittelwert und der Median in Diagramm 7.18, dass alle Modi gleichermaßen *Screenshots* abspeichern, die die Nutzer nicht abspeichern möchten.

7.14 Suchfunktion genutzt

Da die Suchfunktion nur eine Zeitraumsuche erlaubt hat, wurde sie während der kurzen Studie von den Nutzern nur sehr selten genutzt. Die Suchfunktion ist bei den meisten Teilnehmern im abschließenden Interview auch der größte Kritikpunkt. So geben alle Teilnehmer (bis auf Teilnehmer S12) an, dass sie sich für die Zukunft eine Stichwortsuche wünschen, da eine Zeitraumsuche die Ergebnisse nur sehr grob filtern kann.

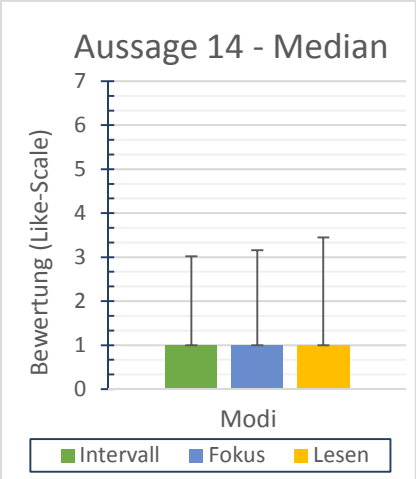


Diagramm 7.19: Medianwerte der Bewertung ob die Suchfunktion genutzt wurde

7.15 Hilfreiche Suchfunktion

Viele Teilnehmer geben im nachträglichen Interview an, dass sie das Textarchiv eine Volltextsuche benötigt. Aus diesem Grund haben viele Teilnehmer die Suchfunktion neutral oder negativ bewertet, da diese, insbesondere für einen solch kurzen Zeitraum, nicht ausreichend umfangreich ist. Nur Teilnehmer S12 bewertet die Suchfunktion als hilfreich, wobei auch die Teilnehmer S04, S08, S09 und S11 die Suchfunktion an jeweils einem Tag ebenfalls besser bewerten.

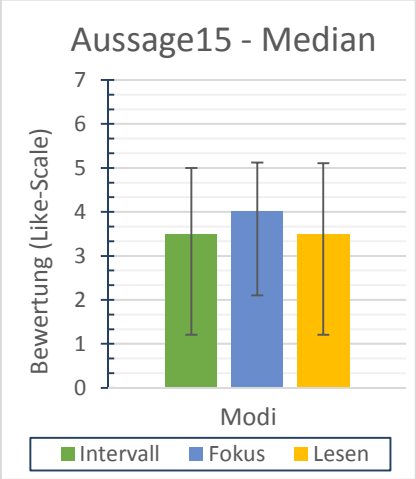


Diagramm 7.20: Medianwerte der Bewertung ob die Suchfunktion hilfreich war

Dabei fällt aber auch auf, dass die Teilnehmer S01, S02, S04, S05 und S06 die Suchfunktion nicht hilfreich finden obwohl sie angeben, die Suchfunktion nicht zu nutzen.

Aus den Ergebnissen abzuleiten, dass eine Suchfunktion überflüssig ist, wäre jedoch falsch, da ausnahmslos alle Teilnehmer im abschließenden Interview betonen, dass ein Textarchiv dringend eine vernünftige Suchfunktion benötigt.

7.16 Resultate der Suchfunktion

Häufig sind die Resultate der Suchfunktion neutral bewertet, da die wenigsten Teilnehmer die Suchfunktion nutzen und diese als hilfreich empfinden. Die Tendenz zeigt aber, dass die Suchfunktion wenig hilfreich ist, weil sie keine brauchbaren Resultate liefert.

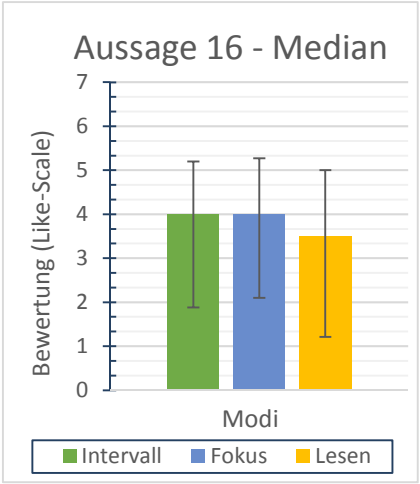


Diagramm 7.21: Medianwerte der Bewertung der Resultate der Suchfunktion

7.17 Performanz Analyse und abschließendes Interview

Die Performanz-Analyse (vgl. Tabelle 6.1, Seite 54) zeigt deutlich, dass der Algorithmus noch nicht optimal läuft. Dies gilt vor allem dann, wenn nur wenig Text gelesen wird. Dabei ist zu erwähnen dass der Algorithmus in der Studie mit Standard-Werten ausgeführt wurde. Diese Standard-Werte sind bei Testläufen während der Entwicklung des Algorithmus entstanden. Deswegen ist es nicht verwunderlich, dass der Algorithmus noch keine befriedigende Erkennungsrate hat. Dennoch zeigt sich in der Studie teilweise ein deutlicher Unterschied zu den beiden anderen Modi.

7.17.1 Wie war die Handhabung?

Alle Teilnehmer geben an, dass sich das Programm einfach bedienen lässt und sie mit der Bedienung grundlegend zufrieden sind. Lediglich die Suchfunktion wird oft kritisiert, da diese keine brauchbaren Ergebnisse liefert. Einem Teilnehmer ist diese zudem zu unübersichtlich.

7.17.2 Hat sie das Programm behindert?

Keiner der Teilnehmer fühlt sich durch das Programm behindert. Allerdings hatten zwei Teilnehmer während der Studie immer wieder Probleme mit dem Eye-Tracker, weil dieser nicht erkannt wurde. Dieses Problem liegt aber an der Tobii-Software bzw. am Tobii-Eye-Tracker-Treiber, der das Gerät teilweise nicht richtig erkennt.

7.17.3 Fühlten sie sich durch den Eye-Tracker oder das Programm beobachtet oder anderweitig unwohl?

Zehn der Teilnehmer geben an, dass sie sich durch den Eye-Tracker oder das Programm nicht oder nur zu kurzfristig beobachtet fühlen. Lediglich zwei Teilnehmer geben an, sich durch den Eye-Tracker und das Tracking-Programm überwacht zu fühlen.

Die Teilnehmer, welche die Studie an einem Laptop durchgeführt haben, hatten teilweise das Problem, dass der Eye-Tracker ihre *Taskleiste* überdeckt hat.

7.17.4 Würden sie das Programm in Zukunft nutzen wollen?

Grundsätzlich ist keiner der Teilnehmer abgeneigt, in Zukunft ein Textarchiv zu nutzen. Aber es sind sich alle Teilnehmer einig, dass das Textarchiv eine Volltextsuche ermöglichen muss, sodass die Suchfunktion auch brauchbare Ergebnisse liefern kann. Für viele Teilnehmer spielt auch die Datensicherheit eine wichtige Rolle. Es muss also sichergestellt werden, dass die Daten privat bleiben. Etwa die Hälfte der Teilnehmer äußert, dass das Programm lernen sollte, was wichtig ist oder zumindest über eine Black- und White-Liste für Anwendungen verfügen sollte. Ebenfalls sollte das Archiv in der Lage sein, Duplikate zu erkennen.

7.17.5 Wann hat ihnen die Tagesübersicht am Besten gefallen / wann waren die Ergebnisse für sie am Besten?

Von den zwölf Teilnehmern finden zehn den Modus mit der Lese-Erkennung, zwei den *Intervall*-Modus am besten. Aufgrund der hohen Anzahl an *Screenshots* die der *Fokus*-Modus anlegt, wird dieser Modus von keinem Teilnehmer favorisiert.

8 Zusammenfassung

Ziel der Bachelorarbeit war es, einen Prototyp für ein Textarchiv zu entwickeln, der die Daten eines Eye-Trackers nutzt, um mit einer Lese-Erkennung möglichst nur die, für den Anwender relevanten Informationen abzuspeichern. Weiterhin sollte gezeigt werden, dass sich Eye-Tracker auch für den Einsatz im Alltag eignen und dadurch in Zukunft eine Vielzahl von Anwendungen von den Daten eines Eye-Trackers profitieren können. Außerdem sollte der Prototyp in einer Benutzerstudie getestet werden um weitere Erkenntnisse für zukünftige Arbeiten zu erlangen.

Hierbei wurden insgesamt drei Verfahren eingesetzt. Eines davon sollte die Inhalte für das Textarchiv ohne den Einsatz eines Eye-Trackers abgreifen, während zwei Verfahren die Daten eines Eye-Trackers für die Bewertung der Relevanz der Inhalte nutzten.

Die Studie ergab, dass das Verfahren mit der Lese-Erkennung in keinem Fall erkennbar schlechter abgeschnitten hat, als die beiden anderen Verfahren.

Darüber hinaus ergab die Studie, dass das Verfahren, welches auf Basis der Daten des Eye-Trackers eine Lese-Erkennung durchführt im Vergleich zu den beiden anderen Verfahren deutlich weniger *Screenshots* abspeichert (vgl. Diagramm 7.4, Seite 56). Dabei wird der Umfang der täglichen Rückschau sowie die Relevanz der enthaltenen Bilder, als auch die für die Rückschau benötigte Zeit bei diesem Verfahren mit Abstand am besten bewertet.

Weiterhin ergab die Studie, dass die Teilnehmer für die Durchsicht der Rückschau zwischen sechs bis acht Minuten aufgewendet haben, wobei das Verfahren dabei keine Rolle gespielt hat. Da jedoch das Verfahren mit der Lese-Erkennung die geringste Anzahl an *Screenshots* abgespeichert hat, wurde hier im Schnitt pro Bild mehr Zeit investiert, was zu einer besseren Retrospektive führt. Zudem konnte durch die Studie festgestellt werden, dass das Verfahren mit der Leseerkennung trotz der geringeren Anzahl an *Screenshots* die Aktivitäten ähnlich gut wiedergibt wie die beiden anderen Verfahren.

Dies zeigt, dass Verfahren zur Bewertung der Relevanz von Inhalten durch die Daten eines Eye-Trackers profitieren und damit die Relevanz der Inhalte besser bewerten können.

Zudem hat sich bei der Durchführung der Studie gezeigt, dass Eye-Tracker durchaus Alltagstauglich sind. Die Erkennungsgenauigkeit war bei den genutzten Eye-Trackern selbst bei der Verwendung mehrerer Monitore befriedigend. Lediglich die Distanz der Augen zum Eye-Tracker stellte teilweise noch ein Problem dar.

9 Ausblick

Um aus dem Prototyp ein vollwertiges Textarchiv zu entwickeln, müssen noch einige Verbesserungen vorgenommen werden. Zudem wurden von den Studienteilnehmern einige Anregungen gegeben, welche Funktionen ein solches Textarchiv enthalten soll, um ihre Bedürfnisse erfüllen zu können.

Zunächst müssen die Daten, die das Textarchiv speichert dringend in einer Datenbank organisiert werden, um zum Beispiel Metadaten (Quelle, Erstellungsdatum, Änderungsdatum, Schlüsselwörter, …) bequem abrufen und in der Suchfunktion nutzen zu können. Bei einem dateibasierten Speichersystem ist der Verwaltungsaufwand dafür einfach zu groß: Wenn die Suchfunktion eine Schlüsselwort-Suche unterstützen soll, müssten, wenn die Schlüsselwörter nicht zentral verwaltet sind, zunächst alle verfügbaren Schlüsselwörter zusammengesucht werden. Die Verwaltung der Schlüsselwörter ist nicht nur aufwändig sondern dadurch auch fehleranfällig. Zudem stellen Datenbanken optimierte Verfahren zur Verfügung um Daten abzurufen, weswegen diese auch besser skalieren. Dies ist bei der zu erwartenden hohen Datenmenge eines Textarchives dieser Art äußert wichtig.

Weiterhin benötigt ein Textarchiv eine Volltextsuche, um Benutzern ein effizientes und bequemes Suchen zu ermöglichen. Dazu ist es notwendig, dass die Einträge durchsucht werden können, was durch *Screenshots* nicht gegeben ist. Entsprechend müssen die Daten entweder mittels anderer Verfahren abgegriffen werden oder der Text aus *Screenshots* mittels einer *OCR*-Software extrahiert werden.

Der Algorithmus zur Lese-Detektion arbeitet noch nicht mit einer zufriedenstellenden Zuverlässigkeit und kann daher ebenfalls verbessert werden. Einerseits ist dies möglich, indem der Algorithmus optimiert wird, andererseits kann eine ausreichend große Benutzerstudie dabei helfen, die optimalen Einstellungen für den Algorithmus zu finden (vgl. Kapitel 4.3.2.1, Seite 29). Zudem sollte der Algorithmus dahingehend verbessert werden, dass er die Lesebewegung auch dann noch erkennt, wenn das Dokument während des Lesens scrollt. Eine weitere Option ist es, den Algorithmus so umzubauen, dass er sich an den Benutzer gewöhnt.

Das Textarchiv sollte in der Lage sein Duplikate zu erkennen und zu vermeiden: Wenn beispielsweise ein Anwender ein Dokument mehrfach liest, muss das Textarchiv in der Lage sein zu erkennen, dass für dieses Dokument bereits ein Eintrag vorliegt. Es soll dann den bestehenden Eintrag erweitern, anstatt einen neuen Eintrag anzulegen.

Das Textarchiv kann so erweitert werden, dass die Relevanz jedes Eintrags vom Benutzer bewertet werden kann. Anhand dieser Bewertung soll das Textarchiv Rückschlüsse darauf ziehen können, welche Informationen für den Anwender eine hohe Relevanz haben um unerwünschte Einträge in der Rückschau zunehmend zu vermeiden.

Bei der Rückschau sollte es möglich sein, einzelne irrelevante Daten zu markieren und direkt aus dem Textarchiv zu löschen. Weiterhin sollte die Option zur Verfügung stehen, Daten aus vorherigen Tagen in die Rückschau aufzunehmen.

Ein wichtiger Punkt für viele Teilnehmer der Studie war die Sicherheit der Daten. Das Textarchiv muss unbedingt vor dem Zugriff Dritter geschützt sein, um die hoch sensiblen Daten, die darin enthalten sind, zu schützen. Ein erster Ansatz dazu ist, dass die Datenbank, in der die Daten organisiert werden, verschlüsselt wird. Da aber ein solches Textarchiv aufgrund der enthaltenen Daten ein lohnendes Ziel für Hacker darstellt, sollten weitere Verfahren entwickelt werden, welche für einen größtmöglichen Schutz der Daten sorgen können.

Die Daten, welche im Textarchiv gespeichert sind, sollen miteinander verknüpft werden können. Zudem sollte es möglich sein, die Daten durch Annotationen zu ergänzen.

Da die Menschen häufig an verschiedenen Endgeräten arbeiten (Desktop-PC, Laptop, Smartphone) sollten die Daten aus dem Textarchiv zwischen den Geräten synchronisiert werden können. Dabei darf aber die Datensicherheit zu keinem Zeitpunkt gefährdet werden, weswegen ein Konzept für die sichere Synchronisierung der Daten erarbeitet und eingebunden werden sollte.

10 Quellenverzeichnis

Apache Software Foundation (Hrsg.). (05. 10 2015). *Apache log4net*. Von Apache Logging Services: https://logging.apache.org/log4net/ abgerufen

Bell, G., & Gemmell, J. (20. 04 2007). Digitales Gedächtnis- Erinnerung total. *Spektrum der Wissenschaft*, S. 9. Von http://www.spektrum.de/magazin/erinnerung-total/869379 abgerufen

Biedert, R., Buschner, G., & Dengel, A. (12. 10 2009). *Text 2.0 - A Brief Introduction.* Abgerufen am 27. 09 2015 von http://text20.net/node/6

Bradley, I. (12. 01 2014). *KittenCode: Releasing WPF diagram under Creative Commons license.* Abgerufen am 29. 09 2015 von https://kittencode.wordpress.com/2014/01/12/releasing-wpf-diagram-under-creative-commons-license/

Buschner, G., Dengel, A., & van Elst, L. (2008). *Eye Movements as Implicit Relevance Feedback.*

Campbell, C. S., & Maglio, P. P. (2001). *A Robust Algorithm for Reading Detection.* Orlando, USA: ACM.

Emam, A., & Youssef, A. E. (2012). *Do Females Read Faster than Males?* King Saud University, Department Of Information Systems.

Fleschner, F., & Matting, M. (08. 07 2013). Angriff aufs Private - Der gläserne Mensch. *FOCUS Magazin*(28 / 2013), S. 4. Abgerufen am 17. 09 2015 von http://www.focus.de/digital/internet/tid-32440/angriff-aufs-private-der-glaeserne-mensch_aid_1036529.html

FOCUS (Hrsg.). (12. 06 2012). *Den Dauerstress hält das Gehirn kaum aus.* Abgerufen am 16. 09 2015 von http://www.focus.de/gesundheit/ratgeber/psychologie/tid-13038/informationsflut-und-staendige-erreichbarkeit-den-dauerstress-haelt-das-gehirn-kaum-aus_aid_360262.html

Fowler, M. (19. 07 2004). *Presentation Model*. Von Martin Fowler: http://martinfowler.com/eaaDev/PresentationModel.html abgerufen

Gemmell, J., Bell, G., & Lueder, R. (21. 02 2006). MyLifeBits: A Personal Database for Everything. (Microsoft Bay Area Research Center, Hrsg.) San Francisco, USA. doi:http://research.microsoft.com/pubs/64157/tr-2006-23.pdf

Gossman, J. (08. 10 2005). *Introduction to Model/View/ViewModel pattern for building WPF apps*. Von msdn Blog - Tales from the Smart Client: http://blogs.msdn.com/b/johngossman/archive/2005/10/08/478683.aspx abgerufen

Isaacs, E., Konrad, A., Walendowski, A., Lenning, T., Hollis, V., & Whittaker, S. (2013). Echoes From the Past: How Technology Mediated. *Echoes From the Past: How Technology Mediated* (S. 10). Paris: ACM.

Kunze, K., Andreas, B., Utsumi, Y., Shiga, Y., & Kise, K. (2013). I know what you are reading - Recognition of Document Types Using Mobile Eye Tracking. *International Symposium on Wearable Computers (ISWC),* (S. 4).

Levy, S. (20. 09 2014). Mode-Erscheinung "Fear of Missing out". *Spiegel Online*. Von http://www.spiegel.de/netzwelt/reeperbahnfestival/fomo-mode-erscheinung-fear-of-missing-out-a-992740.html abgerufen

Microsoft (Hrsg.). (10. 02 2012). *MSDN - Das MVVM Pattern*. Von https://msdn.microsoft.com/en-us/library/hh848246.aspx abgerufen

Microsoft (Hrsg.). (2015). *MSDN - Introduction to WPF*. Von https://msdn.microsoft.com/en-us/library/aa970268(v=vs.100).aspx abgerufen

Microsoft (Hrsg.). (2015). *MSDN - Kompilieren von Anwendungen mit .NET Native*. Abgerufen am 27. 09 2015 von https://msdn.microsoft.com/de-de/library/dn584397(v=vs.110).aspx

Microsoft (Hrsg.). (2015). *What Is Managed Code?* Abgerufen am 13. 10 2015 von MSDN: https://msdn.microsoft.com/de-de/library/windows/desktop/bb318664(v=vs.85).aspx

Morgenroth, M. (2014). 2.1 Der gläserne Konsument. In M. Morgenroth, *Sie kennen dich! Sie haben dich! Sie steuern dich! Die wahre Macht der Datensammler* (S. 287). Droemer. Abgerufen am 18. 09 2015 von https://books.google.de/books?id=C9ltAwAAQBAJ&pg=PT31&lpg=PT31&dq=Datensammler+Kundenmanipulation&source=bl&ots=t9KYm9UFT_&sig=j4_q8_5ysBDU2yWrknXcNiYL82I&hl=de&sa=X&ved=0CCsQ6AEwAGoVChMI18mVt4v_xwIVBr8UCh1_mQeM#v=onepage&q=Datensammler%20Kundenmanipulati

Moser, K., Preising, K., Göritz, A. S., & Paul, K. (2002). *http://www.baua.de/cae/servlet/contentblob/851270/publicationFile/51844/Fb967.pdf.* Universität Erlangen-Nürnberg, Lehrstuhl für Wirtschafts- und Sozialpsychologie. Bundesanstalt für Arbeitsschutz und Arbeitsmedizin. Von http://www.baua.de/cae/servlet/contentblob/851270/publicationFile/51844/Fb967.pdf abgerufen

Nathan, A. (08. 07 2010). *Why WPF, and What About Silverlight?* Von informIT: http://www.informit.com/articles/article.aspx?p=1608130&seqNum=3 abgerufen

Rayner, K. (2009). *Eye movements and attention in reading, scene perception, and visual search.* Psychology Press.

Recall (Hrsg.). (kein Datum). *RECALL - About*. Von http://recall-fet.eu/about/ abgerufen

Rheinische Post. (26. 08 2014). Andrea Nahles: "Mein Ziel ist Anti-Stress-Verordnung“. *Rheinische Post*. Abgerufen am 17. 09 2015 von http://www.rp-online.de/politik/deutschland/andrea-nahles-mein-ziel-ist-anti-stress-verordnung-aid-1.4477896

Riempp, V. (01. 09 2015). Entwicklung einer App zur Leseverfolgung auf Mobilgeräten. Stuttgart, Baden-Württemberg, Deutschland.

Schwichtenberg, H. (21. 01 2008). *Quellcode einiger .NET-Bibliotheken jetzt verfügbar*. Von heise Developer: http://www.heise.de/developer/artikel/Quellcode-einiger-NET-Bibliotheken-jetzt-verfuegbar-352033.html abgerufen

Schwichtenberg, H. (12. 11 2014). *Microsoft: .NET wird komplett Open Source*. Von heise Developer: http://www.heise.de/developer/meldung/Microsoft-NET-wird-komplett-Open-Source-2452033.html abgerufen

Soumyasch. (20. 10 2007). Die .NET-Framework-Hierarchie. Abgerufen am 27. 09 2015 von https://de.wikipedia.org/wiki/.NET#/media/File:DotNet.svg

Statista GmbH (Hrsg.). (2009). *Wie lange nutzen Sie Facebook täglich*. Abgerufen am 16. 09 2015 von http://de.statista.com/statistik/daten/studie/151220/umfrage/taegliche-nutzungsdauer-von-facebook-nach-stundenanzahl-2009/

Tatje, C. (02. 10 2013). Das Recht auf Vergessen. *Zeit Online*, S. 6. Abgerufen am 17. 09 2015 von http://www.zeit.de/2013/41/privatsphaere-internet-datenschutz

Tobii Technology AB (Hrsg.). (04 2014). Tobii-EyeX-Controller. Abgerufen am 27. 09 2015 von http://developer.tobii.com/eye-tracker-mounting-guidelines

Wikipedia. (30. 08 2015). *Liste von Android-Versionen*. Abgerufen am 17. 09 2015 von https://de.wikipedia.org/wiki/Liste_von_Android-Versionen

Youtube (Hrsg.). (16. 09 2015). *Youtube - Statistik*. Von https://www.youtube.com/yt/press/de/statistics.html abgerufen

11 Abbildungsverzeichnis

A) Anhang

Auf den folgenden Seiten sind weiterführende Grafiken sowie die Fragebögen und die Einverständniserklärung zu finden, die bei der Studie genutzt wurden.

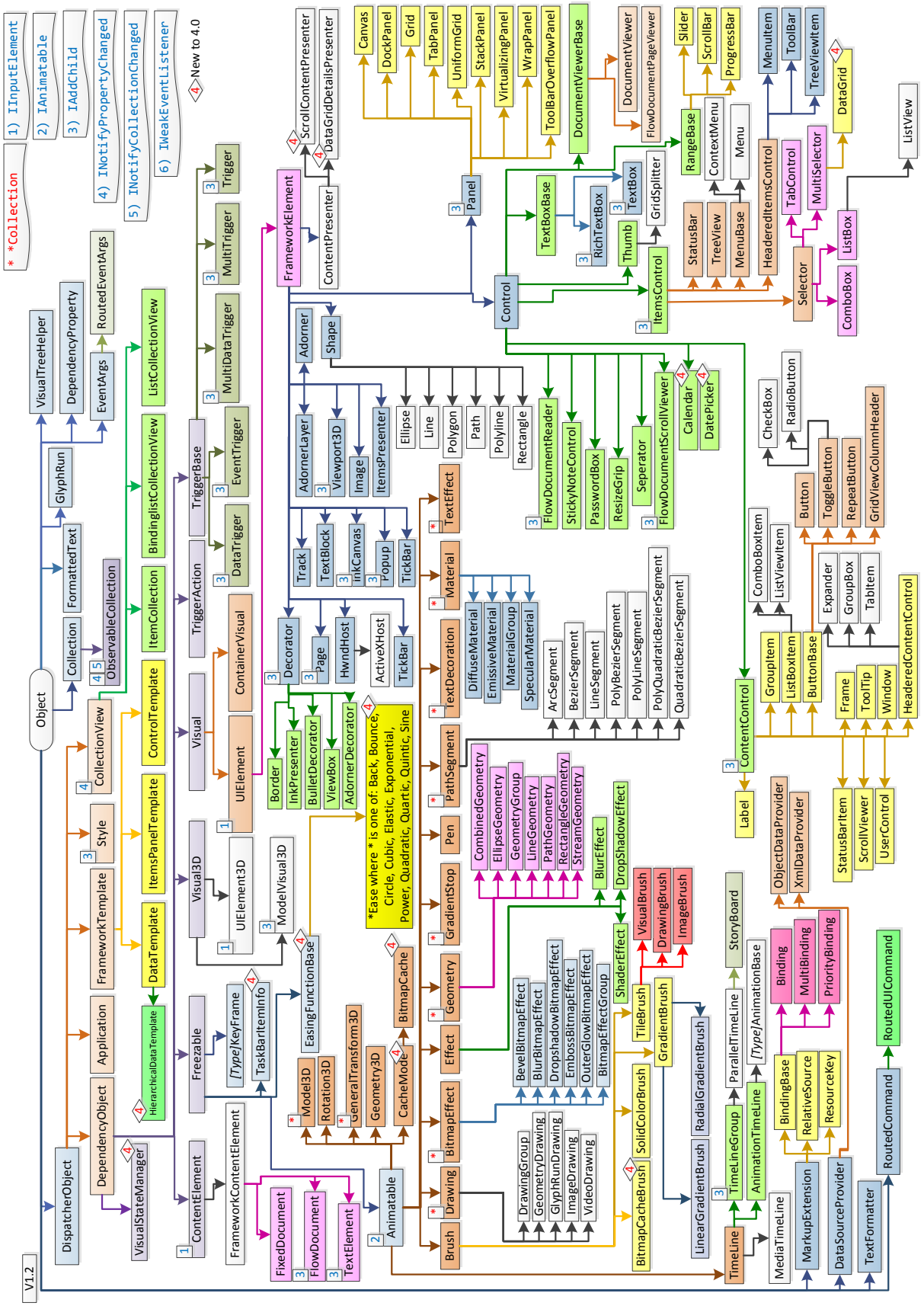


Abbildung A.1: Übersicht über die Klassen der WPF. (Bradley, 2014)

Studie: Desktop Aktivitäts-Rückschau

Universität Stuttgart

Teilnehmer ID: _____

Belehrung

Vielen Dank für die Teilnahme an unserer Studie.

In dieser Studie soll die Nützlichkeit von automatischen Aktivitäts-loggern und der daraus erzeugten Rückschau ermittelt werden. Die Rückschau wird basierend auf Eye-tracking Daten erstellt. Die gesammelten Nutzungsdaten werden anonym behandelt und nur für den Zweck dieser Forschungsstudie verwendet. Die dabei entstehenden Screenshots werden nur lokal abgespeichert und verlassen das Gerät unter keinen Umständen. Ebenso wenig die dargestellten Inhalte der Aktivitätsfenster.

Sie können Ihre Teilnahme jederzeit abbrechen. Bei Fragen, wenden Sie sich bitte direkt an den durchführenden Experimentator,
Gerd Matheis (matheigd@studi.informatik.uni-stuttgart.de), oder an:

Tilman Dingler

University of Stuttgart

Human-Computer Interaction

Email: tilman.dingler@vis.uni-stuttgart.de

Studie: Desktop Aktivitäts-Rückschau

Universität Stuttgart

Teilnehmer ID: _____

Zustimmung zur Teilnahme an der Studie

- ☐ Ich habe das Informationsblatt gelesen und verstanden.
- ☐ Ich habe den Zweck der Studie verstanden und bin bereits daran teilzunehmen.
- ☐ Mir ist bewusst, dass ich den EyeTracker jederzeit deaktivieren kann
- ☐ Ich habe verstanden, dass ich meine Teilnahme an der Studie jederzeit beenden kann.
- ☐ Ich bin einverstanden, dass die Studie aufgezeichnet wird und nehme zur Kenntnis, dass die Aufnahmen nur zu Studienzwecken verwendet und nach Auswertung gelöscht werden.

Teilnehmer ID _____ (vom Studienleiter auszufüllen)

Unterschriften

Teilnehmer _____ Datum _____

Studienleiter _____ Datum _____

Teilnehmer ID: _____

Eingangsfragebogen

A. Demographie

Geschlecht: □ männlich □ weiblich □ keine Angaben

Alter: _____

Beruf / Studiengang: _____

Brillenträger: □ ja □ nein

Verwendung eines externen Monitors: □ ja □ nein

B. Technologieumgang

Wie viele Stunden verbringen Sie durchschnittlich **unter der Woche** pro Tag am Rechner?

- □ weniger als eine Stunde
- □ 1-2 Stunden
- □ 2-4 Stunden
- □ 4-6 Stunden
- □ 6-8 Stunden
- □ >8 Stunden

Studie: Desktop Aktivitäts-Rückschau

Universität Stuttgart

Teilnehmer ID: _____

Verwenden Sie ein System zur Archivierung und Verwaltung von Gelesenem?

□ nein

□ ja, nämlich _____

Nützen Sie ein System zur Verfolgung Ihrer Arbeitsfortschritte?

□ nein

□ ja, nämlich _____

Erklärung

Ich versichere, diese Arbeit selbstständig verfasst zu haben. Ich haben keine anderen als die angegebenen Quellen benutzt und alle wörtlich oder sinngemäß aus anderen Werken übernommene Aussagen als solche gekennzeichnet. Weder diese Arbeit noch wesentliche Teile daraus waren bisher Gegenstand eines anderen Prüfungsverfahrens. Ich habe diese Arbeit bisher weder teilweise noch vollständig veröffentlicht. Das elektronische Exemplar stimmt mit allen eingereichten Exemplaren überein.

Stuttgart, den ____ . ____ . _____ _____