Universität Regensburg

Cross-Device Interaction mit markerbasiertem Window Manager

Bachelorarbeit im Fach B.A Medieninformatik

am Institut für Information und Medien, Sprache und Kultur (I:IMSK)

Vorgelegt von:	Alexander Kugler
Adresse:	Birkenweg 6, 93102 Pfatter
E-Mail (Universität):	Alexander.Kugler@stud.uni-regensburg.de
E-Mail (privat):	alex-kugler@gmx.de
Matrikelnummer:	1866834
Erstgutachter:	Dr. Raphael Wimmer
Zweitgutachter:	Prof. Dr. Niels Henze
Betreuer:	Andreas Schmid
Laufendes Semester:	9. Semester B.A Medieninformatik/Politikwissenschaft(N.F)/Geschichte(N.F)
Abgegeben am:	1.12.2022

Inhaltsverzeichnis

Abbildungsverzeichnis

Zusammenfassung

Durch die zunehmende Verbreitung von Smartphones im Alltag und anderen Lebensbereichen (Arbeit, Studium und privat) ergibt sich für die Nutzer des öfteren die Situation, dass diese Daten von einem PC auf ihr Smartphone übertragen müssen (und umgekehrt). Diese Übertragungen können für die Nutzer in ihren Arbeitsläufen ablenkend sein, da es oftmals erforderlich ist die aktuelle Aufgabe zu unterbrechen um die Übertragung einzuleiten (Beispielsweise das Handy per Kabel verbinden, die Cloud aufrufen und die Datei zum hochladen suchen usw.). Bisherige Ansätze um solche Unterbrechungen zu verringern sind entweder auf ihre jeweiligen Systemarchitekturen beschränkt (wie z.B Apple-Airdrop für Apple-Geräte) oder erfordern weiterhin Schritte wie das suchen und gezielte hochladen der Datei. Um einen Weg zu finden, die Dateiübertragung so unterbrechungsfrei wie möglich zu gestalten, wurde mithilfe des User Centered Design - Prozesses iterativ eine Anwendung implementiert, mit welcher die Nutzer ihre Daten möglichst ohne Unterbrechungen vom PC auf das Smartphone übertragen können. Dafür soll den Nutzern ein Marker in der Desktopumgebung eingeblendet werden (Dem aktuell aktiv genutzten Fenster), welcher von den Nutzern eingescannt werden kann, um die aktuell geöffnete Datei übertragen zu können. Hierfür wurden mit Interviews die Anforderungen an ein solches System erhoben, welches dann im Zuge des Prozesses als ein erster Prototyp implementiert wurde. Dieser Prototyp wurde im laufe einer qualitativen Nutzerstudie anschließend evaluiert, um Daten für eine weiterentwicklung zu gewinnen. Mithilfe der in der Evaluation gesammelten Daten konnte eine weitere, fortgeschrittene Version des Prototypen implementiert werden, welche in einer weiteren Nutzerstudie und einer technischen Evaluation weiter evaluiert wurden. Es wurde eine Anwendung implementiert, welche den Nutzern durch das einscannen eines eingeblendeten Markers die Möglichkeit erlaubt, die aktuell offene Datei von einem lokalen, verschlüsselten Webserver herunterzuladen. Dieses System könnte das Potenzial besitzen, sich als eine mögliche Alternative zu gängigen Übertragunsmethoden (Kabel, E-Mail, Cloud) zu etablieren.

Abstract

With the increasing distribution of smartphones in the context of Work, study and private life, users oftentimes encounter the situation, of having to transfer files from their PC's to their smartphones and vice versa. The transfer of files can be distracting for the users, as they are often forces to interrupt their current task, to begin the transfer by either plugging in their phone or uploading the file to a cloud service, which requires the search for the file in question. Previous solutions are either limited to the respective architecture of the devices (I.e Apple-Airdrop, which is only for Apple-Devices) or still have the requirement of searching and uploading the file required. To implement an alternative, which allows the user to transfer a file uninterrupted, a use centered design (UCD) process was conducted. The alternative has the intention of integrating a Marker into the Desktop-environment (In that case the currently active window) which allows the user to transfer the currently opened file by scanning the displayed Marker.Starting with interviews, to specify the requirements, a first prototype was implemented and evaluated via an user-study. With the data and insights collected from the study, a second, more advanced prototype was implemented. This protoype was also evaluated via an user study and another evaluation, focused on the technical aspects. The result is an application, which allows the user to download the currently file from an local, encrypted server, by scanning a marker displayed on the desktop. This application could be potentially used as an alternative of established file transfer methods (i.e connecting via cable, E-Mail or Cloud)

1 Aufgabenstellung

Mit dem Einzug von Mobilgeräten wie Smartphones und Tablet in die Arbeitswelt bieten sich neue Herausforderungen und Fragen, wie man zwischen stationären Geräten (wie PC's oder andere Displays) und mobilen eine einfache und effiziente Kommunikation und Datenübertragung gestalten kann. Die Forschungsgebiete der Cross-Device Interaction und der Visible Light communication befasst sich unter anderem mit dieser Frage, wobei sich Arbeiten in diesem Gebiet unter anderem die eingebauten Kameras der mobilen Geräte zunutze machen.

Die Arbeit verfolgt das Ziel mithilfe eines Nutzerzentrierten Ansatzes eine Anwendung zu entwickeln, welche mithilfe der von Windows zur Verfügung gestellten Schnittstellen in die Windows-Umgebung integriert werden kann. Diese Anwendung soll dem Nutzer durch das Einscannen des angezeigten Barcodes erlauben, Dateien von seinem Rechner auf das Smartphone zu übertragen. Die Anwendung soll auf das jeweils aktive Fenster in der Windows-Umgebung reagieren und dementsprechend, sofern es sich um eine Datei handelt, die aktuell geöffnete Datei ermitteln und in einem entsprechenden Ordner zur Verfügung stellen. Im Falle eines Browserfensters soll stattdessen die aktuelle URL ermittelt werden. Die Übertragung der betreffenden Datein soll über einen lokal gehosteten Webserver erfolgen, welcher grundlegende Sicherheitsmaßnahmen implementiert wie eine Verschlüsselung per SSL/TLS-Zertifikaten und einer Authentifizierung durch Benutzername und Passwort. Der Barcode soll den entsprechenden Link zur Verfügung stellen, mit dem die Datei dann vom lokalen Webserver heruntergeladen werden kann, bei einer URL wird dann entsprechend die URL im Barcode zur Verfügung gestellt.

Beginnend mit einer Einarbeitung in die Themen der Cross-Device Interaction und der Visible Light Communication sowie einer Vorstellung bislang implementierter, markerbasierter Ansätze zur Dateiübertragung und der Implementierung

eines Proof of Concept Prototypen wird anschließend eine Anforderungserhebung durchgeführt. Damit sollen die Anforderungen der Nutzer an ein solches System ermittelt werden. Mit der Anforderungserhebung sollen auch erste Features für einen ersten Prototypen spezifiziert werden. Dieser Prototyp soll in einer Nutzerstudie daraufhin evaluiert werden, um anschließend mit den Ergebnissen einen weiteren, fortgeschrittenen Prototyp zu entwickeln. Anhand einer weiteren Evaluation sollen weitere Features und nötige Anpassungen ermittelt werden, welche für eine zukünftige Weiterentwicklung relevant sind. Die hier vorgelegte Arbeit dokumentiert die ersten zwei Durchläufe des UCD-Zyklus, sowie einen Ausblick auf eine zukünftige Weiterentwicklung.

2 Einleitung

Seit der Einführung des Smartphones nimmt die Anzahl an Smartphone-Usern beständig zu, allein im Jahr 2021 sollen 62,61 Millionen Menschen in Deutschland ein Smartphone verwendet haben[1]. Mit einer solchen weitreichenden Verbreitung stellt sich diesbezüglich die Frage, für welche Zwecke Smartphones (und auch andere Mobilgeräte wie Tablets) von den jeweiligen Nutzern verwendet werden. In ihrer Arbeit untersuchen Bröhl et al. (2018) verschiedene Altersgruppen und deren Verwendung von PC's, Smartphones und anderen Mobilgeräten für unterschiedliche Aufgaben und wie sich diese Nutzung von Generation zu Generation unterscheidet. Bröhl et al. (2018) konnten ermitteln, dass die jeweilige Verwendung von der Altersgruppe und der intendierten Aufgabe abhängt, so benutzen bestimmte Altersgruppen für bestimmte Aufgaben bevorzugt ein Mobilgerät, während in anderen Altersgruppen hingegen noch Desktop-PC's verwendet werden, während andere Gruppen bestimmte Aufgaben überhaupt nicht ausführten. Mit dem Einzug der Smartphones und anderer Mobilgeräten in den Alltag der Nutzer ergibt sich unter anderem am Arbeitsplatz das Phänomen, dass man potentiell mehrere Geräte zur Verfügung hat, mit welchen man seine Aufgaben erledigen kann. Arbeiten wie Santosa & Wigdor (2013) und Dearman & Pierce (2008) untersuchen anhand von Interviews unter anderen, auf welche Art und Weise die Nutzer in ihrem Arbeitsumfeld mehrere Geräte verwenden, ebenso die Auswirkungen auf den Workflow der Nutzer durch diese. Hierbei sind die Nutzer auch mit mehreren Hürden konfrontiert, welche sich durch die Aufteilung auf verschiedene Geräte ergibt, eine davon ist die Übertragung von Dateien von einem Gerät auf das andere am Arbeitsplatz, welches die Verwendung von zumeist mehreren, unterschiedlich Aufwändigen Methoden[2]

[1] https://de.statista.com/statistik/daten/studie/198959/umfrage/anzahl-der-smartphonenutzer-in-deutschland-seit-2010/

[2] Genannte Methoden sind hierbei unter anderem der Transfer von Daten durch USB-Sticks, SD-Karten, über Cloud Services oder lokale Server Santosa & Wigdor (2013); Dearman & Pierce (2008)

erfordertSantosa & Wigdor (2013); Dearman & Pierce (2008). Im Zusammenhang mit dem Transfer von Dateien beschreiben Pyla et al. (2006, 2009)) in ihren Arbeiten unter anderem das Konzept eines Task-Disconnects, welcher sich beim Arbeiten ereignen kann. Es handelt sich bei einem Task-Disconnect um eine Unterbrechung, welche sich ereignet, wenn die Nutzer zusätzliche Tasks ausführen müssen, die mit dem Haupttask nicht unmittelbar zu tun haben und diesen unterbrechen, ein Problem welches häufig beim Datentransfer auftaucht, ein Problem, welches man unter anderem versucht mit sog. kontinuierlichen Interfaces zu umgehenPyla et al. (2009) bzw. die Wichtigkeit von einer Unterbrechungsfreien Task-Migration betontPyla et al. (2006). Als einen möglichen Ansatz, um auch die User-Experience zu verbessern, bietet es sich somit folglich an, einen Ansatz zu verfolgen, mit welchen die Nutzer ihre Aufgaben bzw. Daten ohne Probleme und ohne Unterbrechungen an ihre Mobilgeräte transferieren können. Um den Datentransfer zwischen unterschiedlichen Geräten einfacher zu gestalten, existieren auch Lösungen, welche sich die jeweiligen Systeme zunutze machen. So existieren Ansätze wie *Apple-Airdrop* [3], Ansätze von *Google Files*[4], *KDE conncect*[5] und *Samsung Quick-Share*[6]. Diese Ansätze besitzen jedoch die Einschränkung, dass diese nur für bestimmte Plattformen oder Gerätekombinationen ausgelegt sind. So erfordern Apple-Airdrop und Samsung Quick Share die entsprechenden Geräte (Iphone bzw.Samsung), während Google Files und KDE-Connect Plattform unabhängiger arbeiten, aber immer noch die Notwendigkeit haben, dass der Nutzer die entsprechenden Dateien zum Übertragen findet und überträgt, was die bereits vorher erwähnten Task-Disconnects ebenso provozieren kann. Um die User-Experience in diesem Fall zu verbessern und um einen Ansatz zu finden, welche die Tak-Disconnects gering hält, soll im Folgenden der Ansatz vorgestellt werden, den diese Arbeit gewählt hat.

[3] https://support.apple.com/de-de/HT204144
[4] https://support.google.com/files/answer/10514188?hl=de
[5] https://kdeconnect.kde.org/
[6] https://www.samsung.com/de/support/mobile-devices/quick-share-und-nearby-share-verwenden/

3 Eigener Ansatz

Diese Arbeit verfolgt das Ziel einen Prototypen zu implementieren, wo die Nutzer durch das Einscannen eines auf dem Desktop eingeblendeten Markers die aktuell geöffnete Datei oder das aktive Browserfenster auf ein Mobilgerät übertragen können. Die Anwendung soll basierend auf dem *User Centered Design Cycle*(UCD), welcher von Autoren wie Norman (2013)[7] beschrieben wurde und in der ISO-Norm 9241-210 (ISO (2020)) definiert ist, implementiert und evaluiert werden. Dieser Zyklus besteht aus vier Schritten (*Observation*,*Idea Generation*,*Prototyping*,*Testing* nach Norman (2013)) und kann nach Bedarf wiederholt werden. Für diese Arbeit wurde der Zyklus zweimal durchlaufen. Im Vorfeld der Implementierung erfolgt eine Betrachtung der bisherigen Ansätzen, welche im Rahmen einer markerbasierten Interaktion bereits umgesetzt wurden. Um die technische Umsetzbarkeit der geplanten Anwendung zu prüfen, wurde ebenfalls im Vorfeld ein Proof of Concept Prototyp implementiert. Basierend auf den ersten beiden Schritten des *UCD* folgt eine Anforderungserhebung mithilfe von Interviews. Mit diesen Interviews sollen die Anforderungen der Nutzer an die geplante Anwendung ermittelt und durch eine Auswertung erste Features ermittelt und spezifiziert werden. Im dritten Schritt des Zyklus soll ein erster Prototyp, basierend auf den vorherigen Ergebnissen implementiert werden. Als vierter Schritt folgt eine Evaluation des Prototypen mithilfe von Nutzerstudien. Basierend auf den Ergebnissen der Evaluation folgt ein zweiter Durchlauf des *UCD*. Im Anschluss erfolgt eine technische Evaluation des Prototypen, um die Scanberkeit von Markern in Abhängigkeit der verwendeten Mobilgeräte, Bildschirme und deren Abständen zueinander zu prüfen. Nach Abschluss des Design-Prozesses und der technischen Evaluation folgt ein Fazit mit einem Ausblick auf weitere Vorgehensweisen für eine Weiterentwicklung des Prototypen.

[7] (In diesem Fall auch als *Human Centered Design* bezeichnet

4 Verwandte Arbeiten

Die hier vorgestellten Arbeiten wurden mit einer Kombination aus Schlagwortsuche sowie mit dem Schneeballverfahren ermittelt. Als Suchplattformen wurden hierbei *Google Scholar*, der *Regensburger Katalog*, sowie die *ACM Digital Library*[8] und *IEE Xplore*[9] verwendet. Im folgenden soll eine Vorstellung der Gebiete der Cross Device Interaktion und der Visible Light Communication erfolgen, anschließend erfolgt eine Betrachtung bisheriger, kamerabasierter und markerbasierter Ansätze.

4.1 Cross Device Interaction

Als Cross-Device Interaction lässt sich die Art und Weise beschreiben, wie ein Nutzer in einer Umgebung mit mehreren Geräten interagiert, die konkrete Definition und Anwendung des Begriffes kann variieren (Scharf et al. (2013)). Aus einer technischen Perspektive kann man die Cross-Device Interaction als eine Interaktionsform bezeichnen, in welcher mehrere (unterschiedliche) Geräte miteinander interagieren und zusammenarbeiten, um bestimmte Aufgaben zu erfüllen. Dabei ist es unerheblich, um welche Art von Geräten es sich handelt, im folgenden sollen allerdings mit einem Fokus auf Mobilgeräte zuerst einige allgemeine Ansätze vorgestellt werden.

4.1.1 Allgemeine Cross Device Ansätze

So lässt sich die Cross device Interaction einsetzen, damit die Nutzer beispielsweise mithilfe ihrer Mobilgeräte mit Displays interagieren können, z.B. direkt als eine Art Stylus, wie in Arbeiten von Schmidt et al. (2012) und Strohmeier (2015) vorgestellt, wo Mobilgeräte verwendet werden, um direkt mit den Oberflächen und Displays zu

[8] https://dl.acm.org/
[9] https://ieeexplore.ieee.org/Xplore/home.jsp

interagieren. Andere Interaktionsmöglichkeiten werden durch Einsatzmöglichkeiten als handybasierte Frameworks wie *Mobile Plus* von Oh et al. (2017) geboten, wo die Nutzer mithilfe des für Android konzipierten Frameworks mit verschiedenen Geräten und Anwendungen direkt vom Handy aus interagieren können, wobei hier über das Handy bestimmte Aktionen an den Anwendungen vorgenommen werden können. Weiterhin lassen sich Cross-Device Ansätze in kollaborativen Arbeiten verwenden, wie es die Frameworks *PolyChrome* (Badam & Elmqvist (2014)) und *PolyVis* (Alsaiari et al. (2019)) machen und den Nutzern die Möglichkeit erlauben, kollaborativ mit verschiedenen Geräten und Displays zu arbeiten. Bereits bei diese Aufzählung wird ersichtlich, dass Cross Device Interaktionen verschiedene Vorgehensweisen als Cross-Device Interaktion bezeichnen kann. Um Klarheit zu schaffen, verfolgen Arbeiten, wie Scharf et al. (2013) das Ziel, Taxonomien[10] für dieses Gebiet aufzustellen. Für eine weitergehende Definition und Einordnung organisiert die Taxonomie nach Brudy et al. (2019), die Cross-Device Interaction anhand von Kriterien wie *Schlüsselcharakteristiken*[11], *Kontext der Anwendung*, *Art der Erfassung* und der dafür *verwendeten Interaktionstechniken*. Eine in der Taxonomie genannte Möglichkeit zur Erfassung und Interaktion ist die Verwendung einer Kamera, welche es erlaubt, Inhalte zu erfassen und damit zu interagieren Brudy et al. (2019).

4.1.2 Kamerabasierte Ansätze in der Cross Device Interaction

Im Rahmen der Cross-Device Interaction wurden Ansätze entwickelt, welche mit einer Handykamera den Inhalt auf dem Bildschirm eines anderen (zumeist stationären) Gerätes oder Displays zu erfassen. Wenn der Nutzer den entsprechenden Inhalt erfasst, kann er diesen entweder auf sein Mobilgerät transferieren oder mit den Inhalten/Anwendungen auf dem Display interagieren. Ein Teil der Ansätze verwendet Algorithmen zur Extraktion von Features und weiteren Informationen aus einem mit der Handykamera aufgenommenen Foto. So erlaubt *DeepShot* von Chang & Li (2011) den Nutzer durch das abfotografieren des Computerbildschirms den ak-

[10] Die Taxonomie von Scharf et al. (2013) unterscheidet hierbei drei Bereiche, *Der Besitz des Gerätes, Der Zugang und die Distanz*

[11] Es handelt sich um sechs Dimensionen:Zeitlich, Konfiguration, Beziehung, Skalierung, Dynamik und Raum

tuell durchgeführten Task oder Inhalt (wie z.B die Navigation auf einer Karte) auf das Handy zu übertragen und von dort aus weiterzuführen, der Inhalt und Zustand der Anwendung wird mit dem *SURF-Algorithmus*[12] erfasst und der Anwendung als *URI*[13] zur weiteren Verfügung gestellt. Bei Hagiwara et al. (2019) wird das aufgenommene Foto verwendet, um die aktuell angezeigte Anwendung zu ermitteln. Diese wird auf das Mobilgerät mithilfe von Screen Mirroring übertragen, wodurch der Nutzer auf dem Mobilgerät den aktuellen Status der Anwendung einsehen und mit dieser Interagieren kannHagiwara et al. (2019).

Ein ähnlicher Ansatz ist auch bei Hu et al. (2013) erkennbar, wo hier dem Nutzer eine Interaktion mit einem größeren Display ermöglicht wird, indem die betreffende Region auf dem Display fotografiert wird. Boring et al. (2007) arbeiten ebenfalls mit großen (öffentlichen) Displays, wo der Nutzer eine Region auf dem Bildschirm abfotografieren kann und der dort angezeigt Inhalt (wie z.B eine Datei) auf einem Webserver zur Verfügung gestellt wird, wo der Nutzer per URL darauf zugreifen kann. Ein weiterer kamerabasierten Ansatz findet sich bei Schmid et al. (2021), wo durch ein Fotografieren der betreffenden Bildschirmregion der Nutzer (nach einem entsprechenden Abgleich durch mehrere Algorithmen) einen entsprechenden Screenshot erhält.

Einen hybriden Ansatz wählen Kajan et al. (2014), hier wird die Interaktion durch eine vorher eingerichtete Verbindung vorgenommen und mit einer Kombination aus Feature-Matching und AR kann der Nutzer mit den Inhalten auf dem PC interagieren. Badam & Elmqvist (2019) verwenden für ihre Arbeit ein Framework, welches es den Nutzern erlaubt, sich Inhalte auf einem Display durch das Einscannen eines dort eingeblendeten Qr-Codes auf dem Mobilen Gerät anzeigen zu lassen. Ein weiterer markerbasierter Ansatz findet sich bei Kajan et al. (2013), hierbei wird ein Marker auf dem Bildschirm angezeigt, bei einer Erfassung durch das mit dem PC vorher verbundenem Gerät können dann anschließend Inhalte oder Tasks migriert werden. Die bisher vorgestellten Ansätze machen sich zum Großteil Algorithmen zunutze, um die fotografierten Inhalte zu erfassen und auszuwerten. Durch das er-

[12] https://people.ee.ethz.ch/ surf/index.html
[13] Uniform Resource Identifier

forderliche Fotografieren besteht somit eine gewisse Abhängigkeit von der Bildqualität, welche unter anderem durch die Qualität der Kamera und weiteren visuellen Effekten wie dem Moiré-Effekt negativ beeinflusst werden kann. Es bietet sich folglich an alternative Ansätze und Möglichkeiten zu betrachten, welche hier vor allem im Gebiet der Visible Light Communication auffindbar sind. Davor soll eine kurze Betrachtung des Themengebiets der Visible Light Communication erfolgen.

4.2 Visible Light Communication

Die Visible Light Communication ist ein Ansatz, bei welchem das für den Menschen sichtbaren Licht im Bereich von 380 - 750 Nanometer verwendet wird, um Informationen zu Übertragen (Khan (2017); Pohlmann (2010); Matheus et al. (2019)). Bedingt durch die Verwendung von sichtbarem Lich ist die Visible Light Communication dementsprechend auch den physikalischen Limits unterworfen, welche für die Verwendung von Licht gelten. So müssen bei der Kommunikation unter anderem Faktoren wie die Line of Sight, Flackern, die Helligkeit der verwendeten LED's und die Störung durch andere Lichtquellen (wie natürliches Licht) beachtet werden, welche sich dementsprechend auf Designentscheidungen auswirken können (Matheus et al. (2019)). Der grundlegendste Aufbau eines VLC-Systems besteht aus zwei Komponenten, einem Transmitter, mit welchem die Informationen übertragen werden (in den meisten Fällen wird eine LED[14] verwendet und einem Receiver, welcher für die Erfassung (in der Regel mit einer Photodiode) und Umwandlung der Lichtsignale in elektrische Spannung zuständig ist (Khan (2017); Pohlmann (2010); Matheus et al. (2019)). Damit die Informationen auch tatsächlich mithilfe von Licht übertragen werden können, müssen diese zuerst in ein entsprechendes Lichtsignal umgewandelt werden, welches mithilfe unterschiedlicher Modulationstechniken[15] erfolgen kann, welche jeweils unterschiedliche Vorteile und Limits haben (Khan (2017); Pohlmann (2010); **?**. Eine der grundlegendsten Modulationstechniken ist

[14] Es sind aber auch andere Lichtquellen möglich,Khan (2017) benennt unter anderem Laser und RGB-LED's, auch bei Pohlmann (2010) wurden unter anderem ebenfalls RGB-LED's, fluoreszierende Lichter und resonant-cavity LED's als mögliche Quellen genannt

[15] In diesem Absatz werden zwei der grundlegenden Techniken vorgestellt, es existieren noch weitere die hier aber nicht im Detail behandelt werden, für weitere Technicken und ihre Details seien die Arbeiten von Khan (2017) und Matheus et al. (2019) empfohlen

das sogenannte On-Of Keying(OOK), in welchem die Informationen übertragen werden, indem die LED ein und ausgeschaltet wird, jeder Zustand (also ein oder aus) repräsentiert hierbei ein Bit (0 oder 1), diese Methode hat bisweilen das Problem einer geringen Übertragungsrate (Khan (2017); Pohlmann (2010); **?**. Color Shift Keying ist eine andere Modulationsmethode, wo das Signal sich aus den unterschiedlichen Farbintensitäten einer RGB-LED zusammensetzt, wodurch eine höhere Kapzität möglich ist (Khan (2017); Matheus et al. (2019)). Durch die verschiedenen Möglichkeiten die Signale zu übertragen, erschließt sich auch ein breites Feld an Anwendungen für die Visible Light Communication. Einsatzgebiete der Visible Light Communication beinhalten unter anderem Verwendung im Straßenverkehr, bedingt durch die vorhandene Infrastruktur (wie Ampeln oder Autoscheinwerfer mit LED's), welche mehrere Verwendungsansäze wie Informationssysteme oder zusätzliche Sicherheitsmechanisnem (Rehman et al. (2019); Matheus et al. (2019)). Weiterhin kann die VLC auch innerhalb von Krankenhäusern als eine Möglichkeit genutzt werden, zu der Überwachung der Patienten und zur Verwaltung und Übertragung vertraulicher Patientendaten (Rehman et al. (2019)). Ein etwas ungewöhnlicherer Verwendungszweck wäre auch die Kommunikation Unterwasser, sowohl zur Kommunikation, als auch zur Steuerung von Vehikeln (Khan (2017); Matheus et al. (2019)). Durch die Verbreitung von LED's lassen sich die visible Light communication auch indoor einsetzen (Matheus et al. (2019); Rehman et al. (2019))

Eine Vorgehensweise zur Einrichtung einer solchen Kommunikation ist die Verwendung von LED's, solche Ansätze machen sich Arbeiten wie Paul & Sohag (2016) zunutze, wo mit einer blinkenden LED Informationen zwischen zwei Laptops übertragen werden (diese sind bei diesem Experiment sowohl Transmitter als auch Receiver), während Hocaoğlu et al. (2019) einen Ansatz konstruieren, mit welchem vorallem Musik und Video übertragen werden können. Bei solchen Ansätzen müssen aber nicht zwangsläufig spezielle Transmitter und Receiver konstruiert werden, es besteht auch die Möglichkeit, Displays als Transmitter und Kameras von Mobilgeräten als Receiver zu verwenden. Solche Ansätze finden sich zum einen bei Kays (2015), wo mithilfe eines Videodisplays die Daten übertragen werden, indem das

Display als ein Transmitter genutzt wird, damit der Nutzer mithilfe der Smartphone Kamera die entsprechenden Informationen auslesen kann. Ein ähnlicher Ansatz findet sich auch bei Yamsang et al. (2017), hier werden allerdings noch zusätzliche Positionsmarker verwendet, um die Erfassung zuverlässiger zu machen und unterschiedliche Kameraqualitäten zu kompensieren. Die Verwendung von Markern und auch in einigen Fällen Barcodes sind ein Ansatz, welcher sich bei mehreren weiteren Arbeiten im Bereich der Visible Light Communication finden lässt, welche im folgenden näher betrachtet werden sollen.

4.3 Übertragung von Informationen mit Marker

Die Übertragung von Informationen mit Markern kann durch das Einscannen eines sichtbaren oder unsichtbaren Markers durch eine Handykamera erfolgen. Diese Marker können sich im Inhalt, Form, Aussehen und den Methoden unterscheidet, mit welchen sie erstellt werden.

4.3.1 Sichtbare Marker

Bei sichtbaren Markern wird den Nutzern ein erkennbarer Marker in einer beliebigen Form präsentiert, welchen die Nutzer mit der Kamera erfassen können, um den Inhalt auf ihr Gerät zu übertragen, die zu übertragenden Informationen werden in der Regel im Barcode encoded. Die Arbeit von X. Liu et al. (2008) ist ein solcher Ansatz, wo die zu übertragende Datei mit einem Encoder in Segmente zerlegt wird und anschließend als ein selbst erstellter, zweidimensionaler Barcode angeordnet wird, Mit einem Decoder auf dem betreffenden Mobilgerät kann dieser erfasst und die entsprechende Datei extrahiert werden.

Einen ähnlichen Ansatz verfolgen Boubezari et al. (2016); Zeng et al. (2014), hier wird der Fokus auf die Kommunikation zwischen zwei Mobilgeräten gesetzt.

Marker können auch (modifizierte) Varianten eines QR-Codes sein, so verwenden Wang et al. (2014) in ihrer Arbeit einen sogenannten *RDCODE*, welcher ein dreischichtiger Barcode ist und sich aus einem Layout, einer Schicht für die Fehlerkorrektur und Anordnung der Informationen und einer dritten Schicht für die

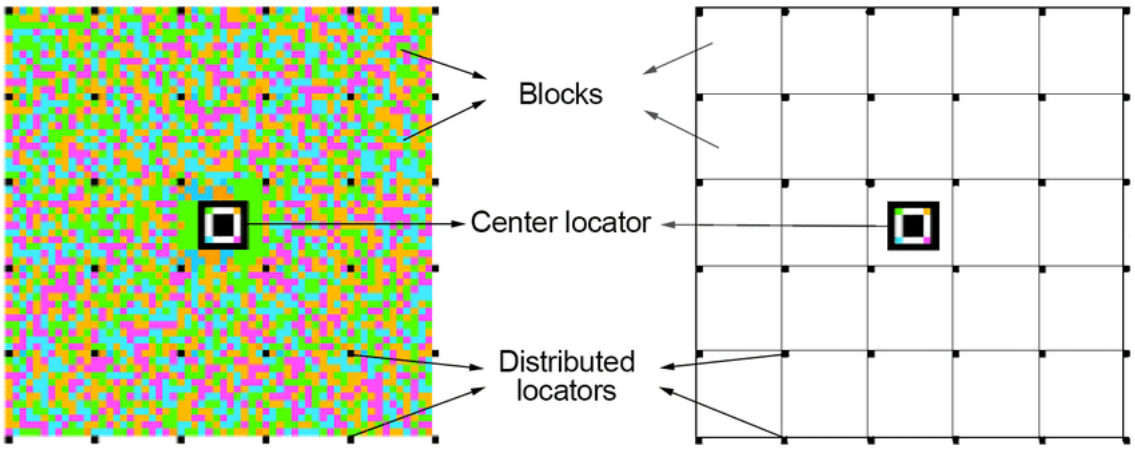

Abbildung 1: Layout eines RDCodes, wie er bei Wang et al. (2014) dargestellt wird

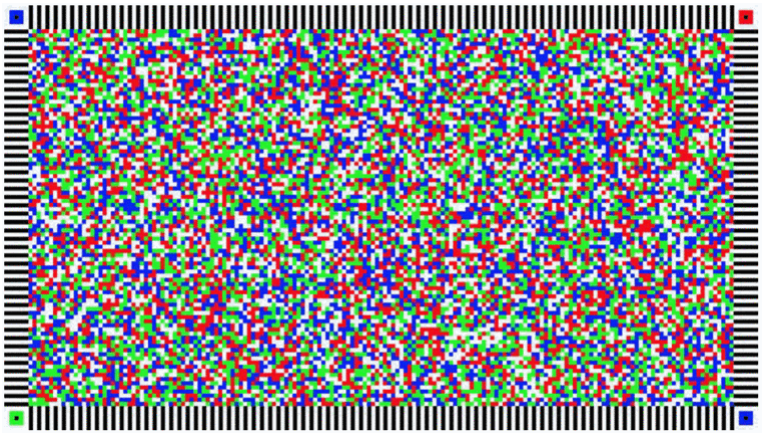

Abbildung 2: Dastellung eines farbigen Barcodes, wie er von Stafford et al. (2017) verwendet wird

eigentliche Anwendung zusammensetzt.

Arbeiten wie solche von Stafford et al. (2017); Hao et al. (2012); Bufalino et al. (2020) verwenden für ihre Arbeiten selbst erstellte und farbige, zweidimensionale Barcodes, welche sich sowohl im Aufbau, als auch in weiteren technischen Details unterscheiden[16], die Informationen werden in Farbblöcken encoded, welche dann in der Form des eigentlichen Markers angeordnet werden.

Arbeiten wie Toh et al. (2016); Fath et al. (2014); W. Liu et al. (2020) verfolgen ein ähnliches Prinzip, sich die erhöhte Kapazität durch mehrere Farben zunutze zu

[16] Vor allem in Bezug auf die Anzahl der verwendeten Farben muss bedacht werden, dass es ein Limit an verwendbaren Farben gibt, es ist hierbei eine Abwägung zwischen mehr Informationen vs. erhöhte Anfälligkeit für Fehler. Die Arbeiten sind da verschiedener Meinungen, Stafford et al. (2017) vertreten die Ansicht, dass mit mehr als 4 Farben gearbeitet werden kann, wenn man mit entsprechend moderner Technik testet. Hao et al. (2012); Bufalino et al. (2020) vertreten hingegen die Ansicht, dass mehr als vier Farben mehr Fehler beim Decodieren verursachen können

	Traditional QR Code	Color QR Code
Real-time transmission rate	150Kbit/s	320Kbit/s
Non-real-time transmission rate	300Kbit/s	900Kbit/s

Abbildung 3: Übertragungsrate eines Farbigen QR-Codes im Vergleich zu einem regulären Toh et al. (2016)

machen, indem hier multiplexen bzw. farbigen QR-Codes, welche durch ein Überlagern mehrerer verschiedenfarbiger QR-Codes eine erhöhte Kapazität für die Datenübertragung bieten.

Eine weitere Markerform ist ein regulärer, schwarz-weißer Quick-Response Code (QR), welche unter anderem als Marker für Frameworks oder Protokolle zur Kommunikation zwischen Mobilgeräten verwendet werden Xie et al. (2013); Freire & Di Francesco (2016); C.-M. Lee et al. (2017) oder direkt zur Übertragung von z.B Audiosignalen J. R. Lee et al. (2016). Eine Variation dieses Ansatzes liefern Mondal et al. (2018) mit einem Framework, welches die Informationen im QR-Code vor dem eigentlichen Encoden mit einem *AES-128 Algorithmus* verschlüsselt, sowie einen Key zum Decoden bereitstellt, um zusätzliche Sicherheit zu bieten. Im Bereich der Sicherheit diskutieren B. Zhang et al. (2015) verschiedene Angriffsmöglichkeiten und Gegenmaßnahmen bei einem sichtbaren Marker, als weitere Sicherheitsmaßnahmen werden Konzepte wie die Limiterung des sichtbaren Winkels[17] oder ein gemeinsamer, geheimer Key, welcher für die Übertragung genutzt wird und von den Nutzern gemeinsam erstellt wird. Alapetite (2010) verfolgt das Ziel, mithilfe eines dynamisch generierten und im Browser eingeblendeten QR-Codes[18] die aktuell aktive Website zu migrieren, der Code enthält hierbei die aktuelle Sitzung (URL + SitzungsID), was ein Fortführen der Sitzung auf einem Mobilgerät erlaubt.

4.3.2 unsichtbare Marker

Eine mögliche Alternative zu sichtbaren Markern sind jene Marker, welche für die Nutzer nicht sichtbar sind, aber durch eine Kamera erfasst werden können, wo-

[17] Für einen Angreifer bedeutet dies, dass dieser ab einem bestimmten Winkel den verwendeten Marker nicht erfassen kann

[18] Die Anwendung erfolgt durch eine eigens implementierte Webandwendung

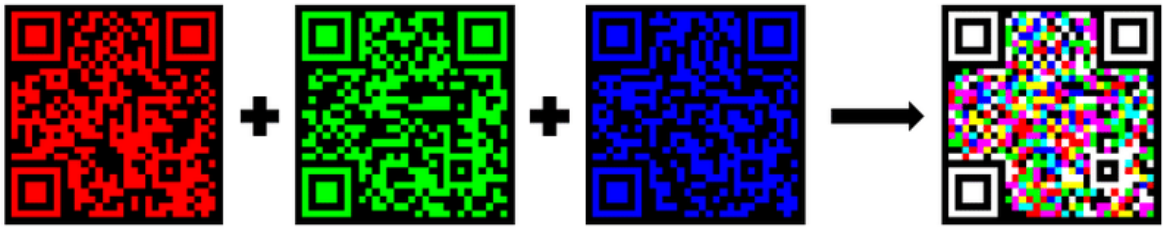

Abbildung 4: Darstellung eines Farbigen QR-Codes und seiner Komponenten in der Arbeit von Toh et al. (2016)

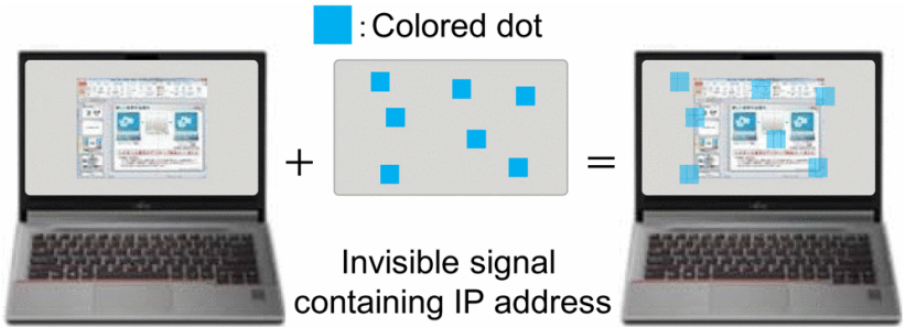

Abbildung 5: Darstellung der Verteilung der unsichtbaren Signale auf dem Bildschirm, wie es bei Kuraki et al. (2016) dargestellt wird

durch dann Informationen übertragen werden können. Hierbei können die Marker in einem grafischen Element (wie z.B ein Bild) integriert werden, wie bei der Arbeit von Collomosse & Kindberg (2008), wo der eigentliche Marker als ein Muster aus verschieden hellen Punkten im einem Bild existiert, welches in ein Bild eingefügt wird.

K. Zhang et al. (2018); Wang et al. (2015); Shi et al. (2015) verfolgen in ihren Arbeiten die Ansätze, die codierten Informationen unsichtbar in Videos zu integrieren, welche für den Nutzer nicht wahrnehmbar sind, aber mithilfe einer Kamera erfasst werden können. Shi et al. (2015) machen sich die verschiedenen Wahrnehmungen von Mensch und Technik zunutze und implementieren Barcode mit einer Framerate von 120 FPS, welche für den Menschen nicht sichtbar sind, aber mithilfe einer Kamera trotzdem erfasst werden können. Hingegen machen sich K. Zhang et al. (2018) daran, ihre Informationen in unterschiedlichen Helligkeitswerten zu implementieren, währen

Als Alternative zur Integration eines Markers bieten sich auch Ansätze an, wel-

che nur mit bestimmten Teilen/Kanälen des Bildschirmes interagieren. Kuraki et al. (2016) arbeiten mit einer Veränderung der RGB-Werte auf dem Bildschirm, indem über den Bildschirm für den Nutzer unsichtbare Signale mit veränderten RGB-Werten gelegt werden. Diese können von der Kamera erfasst werden und ermitteln mit Software auf dem Computer die aktuell offene Datei, welche per FTP-Server oder äquivalenten Methoden auf das Mobilgerät übertragen wird (Kuraki et al. (2016)). Alternativ verwenden Li et al. (2015) für ihren Ansatz den Alpha-Kanal, wo über das eigentliche Bild ein neuer Layer (der sogenannte Communication Layer gelegt wird), wo die Informationen nicht in Form von RGB-Werten, sondern unterschiedlichen Helligkeitswerten zur Verfügung gestellt werden, die Daten werden encoded, indem der darübergelegte Kanal (anstatt direkt die Pixelwerte) manipuliert werden. Beim Einsatz von QR-Codes als Marker besteht theoretisch die Möglichkeit, mithilfe von Ansätzen wie sie von Cui et al. (2019); Abe et al. (2018); Gao et al. (2015) vorgestellt werden, QR-Codes als (kaum) sichtbare Elemente in ein Bild zu integrieren. Mit solchen Ansätzen könnte es somit möglich sein die entsprechenden sichtbaren Marker auch bei Bedarf in unsichtbare umzuwandeln, dies könnte dann aber entsprechende weitere Anpassungen an der jeweiligen Übertragungsmethode nötig machen.

4.3.3 Webbasierte und Open Source Ansätze

Für die Übertragung von Dateien mithilfe eines QR-Codes existieren auch Webseiten oder Software dazu. So erlauben die Webseiten *Scansfer*[19] und *QrClip*[20] dem Nutzer eine Datei (nach einem vorherigen Upload) in einen QR-Code umzuwandeln. Dieser QR-Code kann dann eingescannt werden um die Dateien zu empfangen,*Scansfer* verwendet einen einfachen Upload, während *QrClip* Verschlüsselungen und weiteren Sicherheitsmechanismen, wie Authentifikationscodes und Nutzeraccounts, anbietet. Die Anwendungen sind vom Browser aus verwendbar, besitzen allerdings Limits bezüglich der Anzahl und der zulässigen Größe der zu über-

[19] http://scansfer.com/
[20] https://www.qrclip.io/

Abbildung 6: Menü der Anwendung *Scantransfer*, unter der URL `https://scantransfer.net/` downloadbar

tragenen Dateien.[21]. Bei einem vorher erforderlichen Upload besteht die Notwendigkeit, die Datei im Vorfeld zu suchen und auf der Webseite hochzuladen, letzteres kann für die Nutzer eine potentielle Hemmschwelle darstellen, vor allem, wenn es sich um sensiblen Dateien handelt. Weiterhin existieren Anwendungen wie *scantransfer*[22], wo die Nutzer sich eine Anwendung auf den Desktop installieren können, um die Übertragung durchzuführen. Eine weitere Anwendung ist *qrcp* [23], welche in *Go*[24] geschrieben ist und Ableger in anderen Programmiersprachen inspiriert hat[25]. Es handelt sich um eine Anwendung, welche über die Kommandozeile gesteuert wird und die Dateien über einen dafür eigens eingerichteten, temporären Webserver überträgt. In der Kommandozeile muss der Nutzer in den Speicherort der zu übertragenden Datei manövrieren und anschließend den Befehl zum Start der Übertragung ausführen. In der Konsole wird daraufhin der QR-Code eingeblendet[26] und der temporäre Webserver wird erstellt, von welchem man sich die Datei durch das

[21] Dieses Limit wird hierbei unter anderem durch die Existenz eines Accounts und einer potentiellen kostenpflichtige Mitgliedschaft beeinflusst
[22] https://scantransfer.net/download.html
[23] https://github.com/claudiodangelis/qrcp
[24] https://go.dev/
[25] Unter anderem Python, Javascript und C-Sharp
[26] Alternativ kann man sich den QR-Code in einem separatem Browserfenster anzeigen lassen

Abbildung 7: Screenshot eines Übertragungsprozesses mithilfe von *Qrcp*, einsehbar unter `https://github.com/claudiodangelis/qrcp/`

einscannen des QR-Codes herunterladen kann. Wenn der Nutzer mit der Übertragung fertig sein sollte kann das Programm beendet werden, der Webserver wird dabei ebenfalls beendet.

4.4 Zusammenfassung

Die Gebiete der Cross-Device Interaction und der Visible Light Communication verwenden mehrere Ansätze, um den Nutzern eine Kollaboration bzw. Datenübertragung durch (unsichtbare) Marker zu ermöglichen. Der Nutzer kann durch das Einscannen oder Fotografieren eines Markers oder des Bildschirms die Übertragungen oder Interaktionen starten. Diese Ansätze sind in manchen Fällen auf bestimmte Hardware, Geräte oder Betriebssysteme limitiert, wie z.B auf Mobilgeräte oder bestimmte Displays. Weiterhin können Einschränkungen durch technische Faktoren gegeben sein, wie die Bildwiederholungsrate, visuelle Effekte (wie der z.B vorher genannte Moiré-Effekt oder auch Rolling Shutter unter bestimmten Bedingungen) oder die Qualität der verwendeten Handykamera. Weiterhin können Faktoren wie die Performance der verwendeten Algorithmen, die maximal übertragbare Daten-

kapazität, zulässige Dateiformate[27] sowie Sicherheitsbedenken weitere Limits darstellen. Die Arbeit verfolgt insoweit das Ziel einen universellen Ansatz zu finden, welcher eine entsprechende Kombination aus einer sicheren Übertragung findet und diese sowohl unabhängig vom Betriebssystem als auch vom Dateiformat gestaltet.

[27] Dies bezieht sich darauf, was mit dem jeweiligen Ansatz an Dateiformaten übertragen werden kann, J. R. Lee et al. (2016) haben z.B einen Ansatz, welcher primär zur Übertragung von Audiodateien gedacht ist

5 Proof of Concept Prototyp

Basierend auf den bisherigen Stand der Technik wurde ein Proof of Concept Prototyp entwickelt, um die technische Umsetzbarkeit der geplanten Anwendung zu prüfen. Die Ziele der technischen Entwicklung bei diesem Prototyp waren die Integration eines Markers in die Desktop-Umgebung und eine Möglichkeit den Marker auf Veränderungen am Fenster (Bewegung,Veränderung der Größe,Wechsel des Fensters) reagieren zu lassen. Weitere Ziele waren die Ermittlung des Inhalts vom aktuell aktiven Fenster, sowie die Einrichtung eines lokalen Servers zur Übertragung der aktuell geöffneten Datei, auf welchen über das Einscannen des Markers zugegriffen werden sollte. Bei dem hier vorgestellten Prototypen handelt es sich um einen vertikalen Prototypen. Die Intergration des Markers und die Reaktion auf Events am Fenster wurde vollständig implementiert, für eine fertige Anwendung müsste die Erfassung des aktuell offenen Fenster verbessert werden (Sodass auch Browserfenster erfasst werden können) und die Übertragung der Datei über den Webserver um Sicherheitsmaßnahmen ergänzt werden. Die in diesem Prototyp implementierten Funktionen und Ansätze wurden in späterem Iterationen als technische Grundlagen genutzt.

5.1 Implementierung

Der hier vorgestellte Prototyp wurde unter Windows 10 mit Python 3.10 implementiert, unter Verwendung der von Windows zur Verfügung gestellten APIs und Dll's[28]. Als zentrales Package wurde hierbei das pywin32-Package[29] genutzt, welches für Python zugriff auf die win32-API[30] von Windows erlaubt. Diese API bietet

[28] Dynamic Link Library von Windows, für genauere Details:`https://learn.microsoft.com/de-de/troubleshoot/windows-client/deployment/dynamic-link-library`
[29] `https://pypi.org/project/pywin32/`
[30] `https://learn.microsoft.com/en-us/windows/win32/api/`

unter anderem die Möglichkeit, Informationen und Eigenschaften von Programm-Fenstern zu ermitteln. Die Implementierung der grafischen Elemente erfolgte mithilfe des GUI-Toolkits pyqt, Version 6.4[31], ein Python Binding des QT-Toolkits[32]. PyQT erlaubt hierbei, eine Auswahl an verschiedenen Möglichkeiten GUI's zu erstellen, anzupassen und zu manipulieren.

5.1.1 Mögliche Implementierung auf anderen Betriebssystemen

Eine frühe Variante des hier vorgestellten Protoyps wurde unter in Linux unter einem Debian-Betriebssystem[33] mit einem xfwm4 - Window Manager implementiert[34], welcher das X11-Protokoll verwendet[35]. Diese Variante ist eine frühe Version der Implementierung, welche die Funktionalität besitzt, einen Marker als Fensterelement einzublenden und auch auf Events vom Fenster zu reagieren. Um den hier vorgestellten Prototypen auf anderen Systemen verwenden zu können, müssen mehrere Dinge beachtet werden. Zum einen ist es erforderlich, für jedes Protokoll (und auch Window-Manager) eine Möglichkeit zu finden, das aktuell aktive Fenster sowie dessen Inhalt zu ermitteln. Weiterhin muss man eine Möglichkeit finden, den betreffenden Pfad des dargestellten Inhaltes zu ermitteln und über einen Webserver zum Download bereitzustellen. Der Erfolg einer solchen Implementierung hängt von den Möglichkeiten des jeweiligen Window-Managers ab[36]. Zum Schluss muss man auch auf dem betreffenden System eine Möglichkeit finden den entsprechenden Marker anzuzeigen.

5.1.2 Grundlegende Funktionsweise für den Nutzer

Wenn die Anwendung vom Nutzer gestartet wird, wird das aktive Windows-Fenster im Vordergrund ermittelt und es werden der Titel sowie der Prozess geprüft, welcher die Anwendung aktiv offen hält. Wie in der Abbildung 20 dargestellt, erfolgt

[31] https://www.riverbankcomputing.com/software/pyqt/
[32] https://www.qt.io/
[33] https://wiki.debian.org/de/FrontPage?action=show&redirect=StartSeite
[34] https://docs.xfce.org/xfce/xfwm4/start
[35] https://www.x.org/wiki/
[36] Zur Referenz hier eine Dokumentation des Window-Managers, welcher unter den (neueren) Windows-Betriebssystemen verwendet wird https://learn.microsoft.com/en-gb/windows/win32/dwm/dwm-overview

dieser Ablauf sowohl bei der Erstinitialisierung, als auch bei jedem neu ausgewählten Fenster. Mithilfe dieser beiden Parameter wird durch im Hintergrund ablaufende Funktionen der entsprechende Pfad zur aktuell Datei ermittelt, sofern der Nutzer aktuell eine Datei offen hat. Dieser Pfad wird mit der *Rechner-IP*, dem ermittelten *Dateipfad* und *Serverport* des im Hintergrund laufenden Webservers zu einer URL kombiniert. Diese URL wird durch einen Barcode Generator als Inhalt des Markers gesetzt, welcher dem Nutzer in der Titelleiste des Fensters angezeigt wird. Wenn der Nutzer diesen Marker einscannt, kann er dann über den angebundenen, lokalen Webserver die Datei unter der entsprechenden URL herunterladen. Wenn der Nutzer ein neues Fenster auswählt, wird der Marker entsprechend automatisch vom Programm angepasst. Sollte der Nutzer das Fenster bewegen, skalieren oder minimieren, so bewegt sich der Marker mit dem Fenster (erste beiden Optionen) oder wird beim Minimieren ausgeblendet bzw. eingeblendet beim Wiederherstellen.

5.2 Architektur und Aufbau

Die Anwendung lässt sich in mehrere Komponenten unterteilen, welche für verschiedene Bereiche in der Anwendung zuständig sind. Eine Komponente ist das *Backend*, zuständig für die Ermittlung der aktuell offenen Dateien, der IP-Adresse des Rechners im Netzwerk und den Inhalt des Markers. Eine weitere Komponente ist der lokale Webserver, welcher für die Bereitstellung der Dateien zuständig ist. Die Frontend-Komponente ist sowohl für die visuelle Gestaltung und Positionierung des Markers zuständig, als auch für weitere Operationen am Marker in Reaktion auf Systemevents, welche das Fenster betreffen.

5.2.1 Server

Der Rechner auf dem die Anwendung läuft dient gleichzeitig als Server, welcher die Dateien für den Download zur Verfügung stellt. Um den Rechner als Server im Netzwerk einzurichten verwendet die Anwendung in dieser Iteration einen *simpleHTTP-Server*, welcher Teil des *HTTP-Servers* Pakets in Python ist[37] und sich durch seine

[37] `https://docs.python.org/3/library/http.server.html`

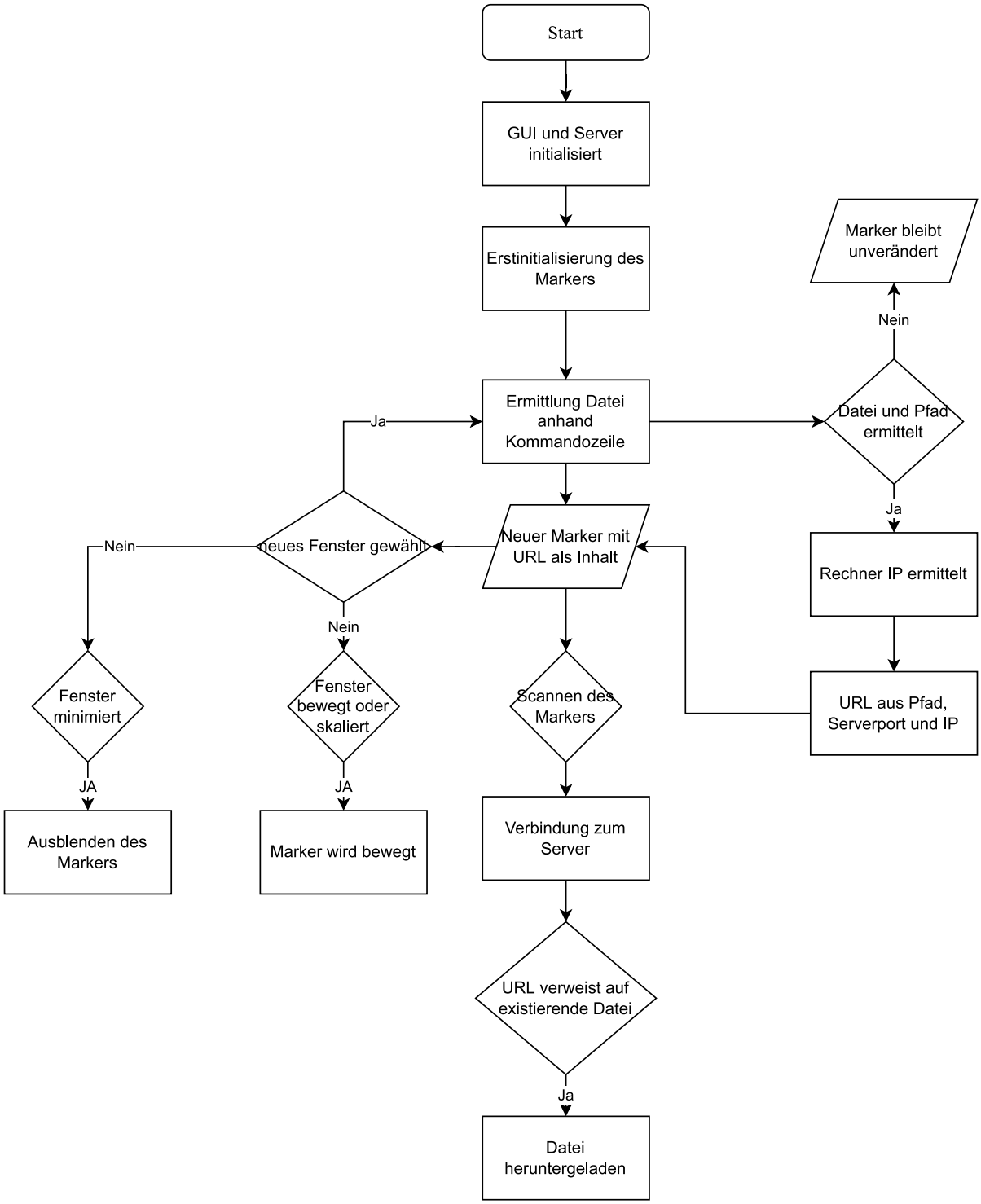

Abbildung 8: Visualisierter Ablauf der verschiedenen Funktionellen Abläufe im Prototypen(eig.Abb)

einfache Einrichtung auszeichnet. Der Server wurde in einem seperaten Daemon-Thread[38] gestartet, welcher so lange aktiv bleibt, wie die Anwendung aktiv ist. Für diesen Prototypen wurden zwei Server eingerichtet, welche mit ihren eigenen Ports und Threads ausgestattet waren und als Directory jeweils eine Festplatte (Entweder C: oder D:) zur Verfügung stellen. Eine Verbindung zum Server kann direkt über die IP-Adresse des Rechners hergestellt werden, listet dafür aber sämtliche Inhalte auf der Festplatte. Mithilfe der kombinierten URL, welche vom Backend erstellt wird, besteht die Möglichkeit, sich über andere Geräte mit dem Rechner zu verbinden und somit die jeweils benötigte Datei gezielt herunterzuladen. An dieser Stelle sei angemerkt, dass der *simpleHTTP* Server laut eigener Dokumentation nur grundlegende Sicherheitschecks und kein https-Protokoll implementiert[39], weswegen die aktuelle Version der Anwendung in nicht-privaten Umgebungen ein Sicherheitsrisiko darstellen kann.

5.2.2 Backend

Reaktion auf Events

Die Reaktion auf Windows Events wurde mithilfe einer Kombination aus einem Ctype[40] in Python, Windows Dll's[41] und einer Callback-Funktion eingerichtet, welche die Events filtert und verarbeitet. Basierend auf der Dokumentation in Win32 für Hooks[42],die Systemhook-Funktion [43] und einer Beispielhaften Implementierung in Python [44] wurde eine modifizierte Version für den Prototypen umgesetzt. Hierbei filtert der Callback die Windows Events innerhalb eines Wertebereiches[45] und führt abhängig vom registrierten Wert[46] eine entsprechende Funktion. Die gefilter-

[38] https://docs.python.org/3/library/threading.html
[39] https://docs.python.org/3/library/http.server.html#http.server.SimpleHTTPRequestHandler
[40] https://docs.python.org/3/library/ctypes.html
[41] https://learn.microsoft.com/en-us/troubleshoot/windows-client/deployment/dynamic-link-library
[42] https://learn.microsoft.com/de-de/windows/win32/winmsg/about-hooks
[43] https://learn.microsoft.com/en-gb/windows/win32/api/winuser/nf-winuser-setwindowshookexa
[44] https://stackoverflow.com/questions/15849564/how-to-use-winapi-setwineventhook-in-python
[45] https://learn.microsoft.com/en-gb/windows/win32/winauto/event-constants
[46] Die Eventkonstanten werden als Hexadezimalwerte gehandhabt

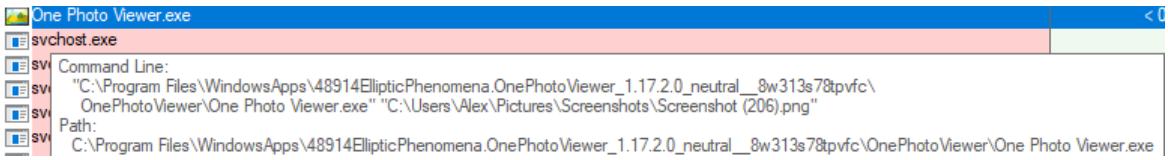

Abbildung 9: Foto eines Prozesses, in diesem Fall eine Imageviewer-Anwendung mit sichtbaren Kommandozeilenparametern, der zweite Parameter enthält hier den Dateipfad zu der aktuell geöffnetem Datei im Programm(eig.Abb)

ten Events sind das *wechseln eines aktiven Fensters*(0x0003 bzw. 3) , das *Minimieren eines aktiven Fensters*(0x0017 bzw. 23) und das *Bewegen eines aktiven Fensters*(0x000B bzw. 11). Bei der Systemhook muss beachtet werden, dass die Performance der Anwendung oder des Systems beeinträchtigen kann[47], es ist folglich erforderlich die Events möglichst präzise und in einem kleinen Wertebereich zu filtern.

Erfassung der Dateien und Generieren des Markers

Für die Ermittlung der Datei werden der Name des Fensters im Vordergrund und der Name des für den Fenster zuständigen Prozesses ermittelt. Für die Ermittlung des Prozesses wird das Package psutil[48] verwendet, welches es erlaubt Informationen zu aktuell laufenden Prozesses zu ermitteln. Anhand des Handles[49] des aktuell offenen Fensters im Vordergrund wird der mit dem Fenster assoziierte Prozess ermittelt. Wird ein entsprechender Prozess gefunden, so werden anschließend die Kommandozeilenparamter, welche der Prozess verwendet überprüft. Hierbei wird standardmäßig der zweite Parameter geprüft, der erste enthält in der Regel den Startbefehl für das Programm selbst, während der zweite Parameter in der Regel den Parameter für die Datei, mitsamt Pfad, enthält. Bei einem solchen Vorgehen besteht die Möglichkeit, dass ein Programm keinen zweiten Parameter besitzt oder auch dass die Parameter nicht aktualisiert werden, wenn eine neue Datei intern in der Anwendung geöffnet wird[50]. Dieses Problem tritt ebenfalls auf, wenn mehrere Dateien im gleichen Programm geöffnet werden. Mit diesem Ansatz lässt sich

[47] `https://learn.microsoft.com/de-de/windows/win32/winmsg/about-hooks`
[48] `https://pypi.org/project/psutil/`
[49] `https://learn.microsoft.com/en-us/windows/win32/sysinfo/handles-and-objects`
[50] Dies kann der Fall sein, wenn man im aktuellen Programm über den internen Dateibrowser eine Datei öffnet

somit immer nur die als erstes vom Programm geöffnete Datei damit ermitteln[51]. Als Endergebnis wird der Dateipfad (sofern ermittelbar) mit der Rechner-IP und dem Serverport zu einer URL kombiniert. Aus dieser URL wird mithilfe des Packages Python-Barcode[52] ein neuer Marker generiert. Python-Barcode erlaubt es nach Bedarf eindimensionale oder zweidimensionale, rechteckige Barcodes zu erstellen. Der Barcode wird als eine PNG-Datei gespeichert, diese wird vom Frontend in das Interface eingebunden. Bei der Auswahl eines neuen Fensters wird der Prozess wiederholt und der vorherige Marker durch den neu generierten ersetzt (sofern ein Pfad ermittelt wird, ansonsten bleibt der Barcode unverändert).

5.2.3 Frontend

Visuelle Gestaltung

Hierbei wurde der Marker in einem QLabel[53] eingeblendet, welches für das Anzeigen von Bildern (und gegebenenfalls von Texten) geeignet ist. Der Marker im Label wurde über eine Q-Pixmpap[54] gesetzt, eine Zeichenfläche die das unter anderem das Anzeigen von Bildern erlaubt. Mithilfe von sogenannten Window-flags, mit welchen eine Modifikation des PyQT-Windows möglich ist, wurde das Hauptfenster mit dem Label angepasst. Dabei wird der Hintergrund des Fensters transparent gemacht und die Fensterdekorationen (Rahmen, Titelleiste und andere Steuerelemente) entfernt, sodass am Ende nur der Barcode sichtbar ist, wie im Endresultat der Abbildung 3 zu sehen ist. Zusätzlich wird der Input von Mouseevents für den Marker deaktiviert, wodurch das der Marker nicht bewegt oder anderweitig mit der Maus manipuliert werden kann. Bei angezeigten Marker in dieser Anwendung handelt es sich um ein Overlay, welches im an der Position der Titelleiste des aktuellen Fensters platziert wird. Wenn Systemevents von der Callback-Methode registriert werden, wird der Marker mit move-operation bewegt oder ausgeblendet. Die Positionen zur Platzierung des Markers werden basierend auf den Koordinaten und der

[51] Dies würde dann erfordern, dass für jede neue Datei das Programm enstrechend geschlossen und neu gestartet werden muss

[52] `https://pypi.org/project/python-barcode/`

[53] `https://doc.qt.io/qt-6/qlabel.html`

[54] `https://doc.qt.io/qt-6/qpixmap.html`

Abbildung 10: Beispielhaftes Einblenden des Markers als Overlay(eig.Abb)

Ausmaße des aktuellen Fensters im Vordergrund ermittelt und als Ausgangspunkte für die Bewegung und Platzierung des Marker verwendet.

5.3 Ergebnisse und Schlussfolgerungen für die weitere Entwicklung

Im Verlaufe der Implementierung wurde ein erster Prototyp implementiert, dessen Komponenten sich in abgeänderter und erweiterter Form in späteren Iterationen wieder finden lassen. Bestandteile, welche in späteren Versionen modifiziert verwendet wurden, waren die Erfassung von Systemevents, die Art und Weise, wie der Marker auf dem Fenster angezeigt wird, und das ermitteln der aktuell geöffneten Datei im offenen Fenster. Basierend auf diesen Erkenntnissen, wurde im weiteren Verlauf der Arbeit eine Studie zur Anforderungserhebung durchgeführt, welche die Konzepten aus diesem Prototypen als Grundlage hatten. Mit diesem Prototypen kann eine grundsätzliche Umsetzbarkeit des Konzeptes, der Integration eines Markers in die Desktopumgebung nachgewiesen werden.

6 Anforderungserhebung

Beginnend mit den ersten beiden Schritten des UCD-Zyklus (Observation und Idea Generation nach Norman (2013)) folgte eine Anforderungserhebung, um basierend auf den Ergebnissen der Erhebung und der bisherigen technischem Möglichkeiten der ersten Entwicklung, Daten für die Entwicklung eines Prototyps zu generieren. Als Methode der Anforderungserhebung wurden qualitative Interviews mit Probanden durchgeführt. Das Ziel der Interviews war die Ermittlung der Vorstellungen und Anforderungen der Nutzer an eine solche Anwendung sowie die Verhaltensweisen und der Vorgehensweise der Nutzer zur Übertragung von Dateien von einem stationären Rechner auf ein mobiles Endgerät(Smartphone, Tablet). Die Interviews wurden anschließend in einem zweiten Schritt ausgewertet, um basierend darauf Features für den ersten Prototyp zu generieren.

6.1 Probanden

Für die Interviews der Anforderungserhebung wurden fünf Probanden aus dem privaten Umfeld rekrutiert. Drei Probanden sind weiblich, zwei davon männlich mit einem Altersbereich von 24 - 50 Jahre. Drei Personen sind Studierende verschiedener Fachrichtungen, eine Person ist selbstständig berufstätig und eine Person ist angestellt. Alle Probanden waren laut eigener Aussage erfahren im Umgang mit stationären Rechnern sowie mit Diensten und Hardware zur Übertragung von Dateien.

6.2 Studiendesign

Für die Anforderungserhebung wurde ein semi-strukturiertes Interview konzipiert und durchgeführt, basierend auf den vorgestellten Methoden von Lazar et al. (2017).

Die Fragen im Interview waren in vier Blöcke aufgeteilt und umfassten Fragen zum *präferierten visuellen Design*, *Verhalten und Methoden bei der Übertragung von Dateien*, *Vorstellungen zur Sicherheit und Feedback* und *Steuerung und Interaktionsmöglichkeiten*. Der Block zur visuellen Gestaltung beinhaltet einen interaktiven Teil, in welchem die Nutzer Bilder von verschieden großen Markern, mit unterschiedlichen Designs, an den von ihnen präferierten Positionen platzieren konnten.

6.3 Aufbau und Durchführung

Die Probanden wurden in separaten Sitzungen über *Teamspeak*[55] oder *Discord*[56] interviewt, die Interviews wurden nach vorherigem Einverständnis der Probanden aufgezeichnet. Beginnend mit einer kurzen Begrüßung wurde den Probanden der Zweck der Studie präsentiert. Dafür wurde im Rahmen einer kurzen Demonstration und der Vorführung eines Videos vom Proof of Concept Prototypen das geplante Prinzip der geplanten Anwendung demonstriert, um den Probanden das grundlegende Konzept der Anwendung verständlich zu machen. Nach dem ersten Block mit Fragen zu den Methoden der Nutzer für die Dateiübertragung wurde der nächste Block mit der interaktiven Komponente eingeleitet. Den Probanden wurde mithilfe von *Teamviewer*[57] zugriff auf den Rechner des Versuchsleiters gewährt. Den Probanden wurde im Bildbearbeitungsprogramm *GIMP*[58] das Bild eines Programmfensters und sechs Marker in verschiedenen Größen präsentiert (originale Größe, wie von Python-Barcode generiert, sowie zwei skalierte Varianten, insgesamt jeweils drei PDF417 und QR-Codes), welche dann an der von ihnen präferierten Position angeordnet werden sollten. Nachdem die Probanden alle Marker platziert hatten, wurden sie nach ihren Präferenzen bezüglich ihrer favorisierten Markerformen, Größen und Positionen befragt. Die Studie kehrte nach diesem Block wieder zum regulären Ablauf zurück. Die Probanden wurden nach ihren Vorstellungen befragt, wie der Marker im Fenster auf entsprechende Aktionen am Fenster reagieren soll (Bewegung, minimieren, Auswahl eines neuen Fensters). Weiter-

[55] `https://www.teamspeak.com/en/`
[56] `https://discord.com/`
[57] `https://www.teamviewer.com/de/`
[58] `https://www.gimp.org/`

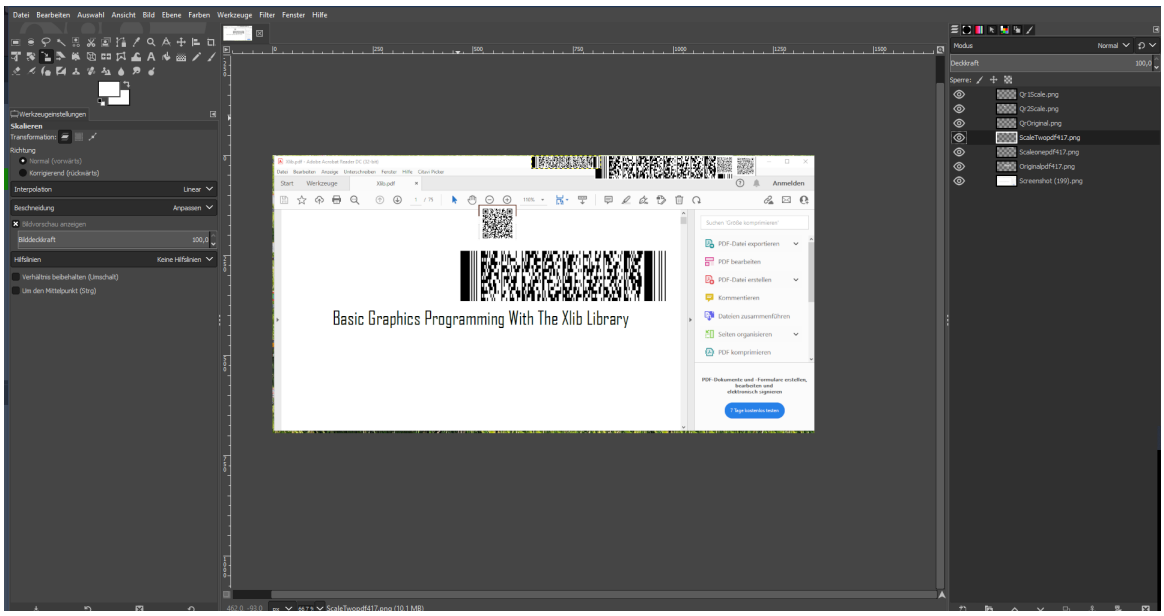

Abbildung 11: Der visuelle Task für die Probanden, hier mit allen sechs Möglichen Markern eingeblendet. Diese Aufnahme wurde nach einem Interview aufgenommen, folglich kann es sein, dass sich einzelne Marker überlagern(eig.Abb)

hin wurden die Probanden noch nach ihren Vorgehensweisen beim Einscannen von Barcodes befragt. Zum Schluss folgten Fragen zur Sicherheit der Übertragung und dem Bedarf an Feedback durch die Anwendung. Nach Abschluss des Interviews wurde den Probanden für ihre Teilnahme gedankt und sie wurden verabschiedet.

6.4 Ergebnisse

Die aufgezeichneten Interviews wurden für eine bessere Lesbarkeit sinngemäß transkribiert und geglättet. Diese Transkripte wurden dann einer thematischen Kodierung unterzogen, wie sie von Blandford et al. (2016) vorgestellt wird, basierend auf den Prinzipien von Braun & Clarke (2006), und in übergeordnete Themen eingeteilt. Basierend auf den Ergebnissen der Kodierung konnten die folgenden übergeordneten Themen ermittelt werden:*Feedback, Arbeitsabläufe der Nutzer, Größe und Form des Codes, Erwartungen, Bedienung und Positionierung des Codes*. Im folgenden sollen die jeweiligen Themen kurz zusammengefasst vorgestellt werden.

Arbeitsabläufe

Die Nutzer befinden sich laut eigenen Aussagen regelmäßig in Situationen, wo eine Datenübertragung vom PC auf das Handy erforderlich ist, diese findet in verschiedenen Kontexten statt und umfasst sowohl das Studium, Arbeit oder auch Privates. Die Datenformate, welche am häufigsten übertragen werden, sind Dokumente und Bilddateien. Häufig gennante Methoden sind die Übertragung per Messenger wie *Whatsapp*[59] Clouddienste, vor allem *Google Drive*[60] und Dropbox[61]). Weitere Methoden sind das Verbindung des Smartphones über Kabel mit dem Rechner oder durch ein manuelles abtippen der URL bei Webseiten. Die Verwendung der jeweiligen Vorgehensweise hängt von der Dateigröße und der Sensibilität der Dateien ab, große Dateien werden vor allem über Cloud-Dienste verschickt, kleinere Dateien werden per Messenger übertragen. Wenn es sich um sensible(re) Daten handelt werden diese entweder per E-Mail oder Kabel übertragen.

Erwartungen

Die Dateien sollten mit einer möglichst gleichbleibenden Qualität, einer möglichst hohen Geschwindigkeit und einer einfachen Bedienung übertragbar sein. Die Übertragung sollte soweit gesichert sein, dass die zu übertragenden Dateien nicht durch unbefugte abgefangen werden können.Die Nutzer bevorzugen es, wenn sie alle für sich wichtigen Funktionen der Anwendung (Einblenden, Position und Größe) selber kontrollieren können und äußern auch ein Bedürfnis an einer entsprechenden Flexibilität der Anwendung, vor allem im Kontext der Größe und der Positionierung des Markers.

Bedienung und Sicherheit

Im Kontext der Sicherheit betonen die Nutzer die Eigenverantwortung im Umgang mit den eigenen Dateien oder empfehlen Methoden wie eine Authentifizierung per Passwort oder ein Log, um Geräte und Interaktionen nachverfolgen zu können. Um

[59] `https://www.whatsapp.com/?lang=de`
[60] `https://www.google.com/intl/de/drive/`
[61] `https://www.dropbox.com/`

den Marker aufzurufen werden Methoden wie *integrierte Buttons im Interface*, *Tastenkombinationen*, *Menüoptionen* im ausgewählten Fenster oder *Kontextmenüs* empfohlen.Bei der Verwendung von Tastenkombinationen wurden als verschiedene Möglichkeiten genannt, diese können entweder vorgegeben oder frei konfigurierbar sein.

Größe und Form des Codes

In der Frage der präferierten Form ist eine Tendenz zum QR-Code zu erkennen,trotz der besser geeigneten Form des PDF417 für die Titelleiste, bedingt durch die verbreitetere Bekanntheit von QR-Codes. Die ideale Größe des Markers ist von der jeweiligen Situation, der jeweiligen Einblenddauer und dem Kontext abhängig. Bei einer temporären Einblendung soll der Marker so groß wie möglich sein, bei einer permanenten Präsenz wird der Marker in einer kompakten Größe bevorzugt, welche in etwa die Höhe der Titelleiste des Fensters hat. Kleinere Marker sollten in den Freiräumen der Titelleiste platziert werden, große Marker hingegen bevorzugt in der Mitte des Bildschirms. Teile der Probanden würden eine permanente Präsenz nicht als störend empfinden, wenn der Marker entsprechend klein und unauffällig genug ist, während für den anderen Teil eine permanente Präsenz grundsätzlich störend wäre.

Verhalten des Codes

Im Bezug auf das Verhalten des Markers bei Events, welche das Fenster betreffen, sollte der Marker sich entsprechend mit dem Fenster mitbewegen bzw. seine Größe und Sichtbarkeit anpassen, basierend auf dem aktuellen Status vom Fenster. Die Nutzer bevorzugen es auch den Marker auf eigenes Verlangen ein- und ausblenden zu können, sodass der Marker auch permanent verfügbar sein sollte.

Feedback

Direktes Feedback seitens der Anwendung am Rechner wird nicht gefordert, wobei eine Art von Verlaufshistorie von einigen Probanden als ein wünschenswertes Feature gesehen wird. Ein Verlauf, welcher Vergangene Interaktionen anzeigt, wird als

ein wünschenswertes Feature betrachtet wird. Ebenfalls haben die Probanden kein explizites Bedürfnis an zusätzlichem Feedback am Smartphone, da bereits auf dem Gerät die Möglichkeit besteht, den Fortschritt und den Erfolg eines Scanversuches verfolgen zu können.

6.5 Diskussion und Spezifikation der Anforderungen

Nachdem die Ergebnisse der jeweiligen Kodierungen zusammengefasst wurden, erfolgt im folgenden eine kurze Spezifikation der Features welche in der geplanten Anwendung basierend auf den Erkentnissen der Erhebung implementiert werden sollen.

Verhalten des Markers

Für die Anforderungen in diesem Bereich ergeben sich für den bisherigen Ansatz im Proof of Concept Prototypen und dem Umgang mit Systemevents kein Bedarf an Veränderungen bezüglich der Erfassung. Es bietet sich an, entsprechende Erweiterungen durchzuführen, um die Erfassung zuverlässiger, performanter und nach Möglichkeit visuell ansehnlicher für die Nutzer zu machen.

Feedback

Basierend auf den Ergebnissen ergibt sich keine besondere Notwendigkeit eine explizite Feedbackfunktion einzufügen, da die Nutzer laut eigenen Aussagen den Fortschritt bereits am Smartphone und Rechner mitverfolgen können, wodurch eine Feedback funktion redundant wäre.

Bedienung der Awendung

Laut den Nutzern wird die Sicherheit in die Hände des betreffenden Nutzers gelegt, sodass dieser selber verantwortlich ist. Für den kommenden Prototyp ist ein Sicherheitsmechanismus noch nicht eingeplant, spätere Iterationen sollen aber einen rudimentären Sicherheitsmechanismus implementieren (Wie z.b eine simple Passwortabfrage). Die Steuerung der Anwendung sollte, um verschiedenen Präferenzen

entgegenkommen, eine Steuerung der Anwendung sowohl mit Maus als auch mit Tastatur zu erlauben. Wenn eine Steuerung per Tastatur erlaubt wird, müssen Konfigurationsmöglichkeiten geboten oder universelle Kombinationen ermittelt werden.

Erwartungen

Die Anwendung sollte nach Bedarf Konfigurationsmöglichkeiten für die Nutzer implementieren, mit denen die jeweiligen Aspekte angepasst werden können. Die Anwendung muss in der Lage sein, Dateien, unabhängig der Größe, in einer für den Nutzer angemessenen Geschwindigkeit, übertragen zu können. Basierend auf den Erkentnissen über die Arbeitsabläufe der Nutzer müssen Interface und Bedienung für die Nutzer entsprechend einfach gestaltet werden, sodass diese nicht Komplexer sind, als gängige, von den Nutzern verwendete Methoden, aber dennoch einfacher, als bestehende Methoden, was mit der Integration des Marker in die Desktopumgebung und dem Einscannen bereits erreicht sein sollte.

Größe und Form des Codes

Basierend auf den Präferenzen und Vertrautheit der Nutzer bietet es sich hier an, den bisherigen PDF-417 durch einen QR-Code zu ersetzen. Im Kontext der Größe ergeben sich verschiedene Ansätze. Der Marker kann so gestaltet werden, dass er ein permanent sichtbares Element in der Desktopumgebung darstellt, dies erfordert einen Kompromiss zwischen einer kompakten, unauffälligen Größe und der Einscanbarkeit des Markers. Eine alternative Möglichkeit ist ein Marker, welcher nur im Bedarfsfall eingeblendet wird und dessen Größe und Position von den Nutzern nach Bedarf angepasst werden kann.

Im Anbetracht der Ergebnisse aus der Erhebung ergeben sich zwei Gestaltungsmöglichkeiten für die Anwendung. Entweder kann man sich entscheiden den Ansatz mit dem Permanent eingeblendeten Marker beizubehalten, mit einer veränderten Form und Größe oder alternativ eine Anwendung, mit welcher die Probanden die Möglichkeit haben selbst über das Design, die Positon und die Größe zu entscheiden. Aus diesem Grunde wurde im dritten Schritt des UCD-Zyklus die Ent-

scheidung getroffen, zwei Varianten des Prototypen zu implementieren, um eine endgültige Entscheidung zwischen einem aktiv gesteuerten und einem passiv angezeigten Marker treffen zu können.

7 Erster Prototyp

Im dritten Schritt des UCD-Zyklus (Protoyping nach Norman (2013)) wurde, basierend auf den Ergebnissen der Anforderungserhebung, zwei Varianten des Prototypen entwickelt. Diese Prototypen unterscheiden sich bezüglich der Interaktionsmöglichkeiten, welche dem Nutzer zur Verfügung gestellt werden. Die erste Variante dieses Prototyps blendet den Marker permanent in der Desktopumgebung ein und erlaubt dem Nutzer keine Interaktionsmöglichkeiten. Die zweite Variante erlaubt mithilfe von Hotkeys die Manipulation der Größe und Sichtbarkeit des Markers, der Nutzer kann den Marker mit der Maus bewegen und platzieren. Bis auf diesen Unterschied verwenden die beiden Prototypen dasselbe Backend und Frontend und unterscheiden sich lediglich in der zuständigen Komponente für die Interaktion untereinenader. Der im Kapitel Proof of Concept Prototyp vorgestellte Proof of Concept Prototyp wurde hier als technische Basis für die beiden Prototypen verwendet

7.1 Veränderungen im Vergleich zum Proof of Concept Prototypen

Das grundlegende Prinzip der Anwendung ist gleich geblieben, es wurden bei diesem Prototypen die Funktionen zur Erfassung der Dateien modifiziert und erweitert. Um die aktuell geöffnete Datei zu übertragen, muss der Nutzer das Fenster mit der geöffneten Datei auswählen, die Anwendung kann jetzt auch Tabs in Anwendungen, Browsertabs und auch ausgewählte Windows-Anwendungen erfassen. Die Übertragung durch den lokalen Webserver wird jetzt auf einen festgelegten Ordner beschränkt, welcher die aktive Datei für die Übertragung enthält. Visuell wurde der Marker angepasst, durch einen QR-Code ersetzt und mit Funktionen zum Anpassen der Größe und Position versehen.

7.2 Implementierung und Erweiterung des Proof of Concept Prototypen

Die beiden Prototypen wurden wie der Proof of Concept Prototyp unter Windows 10 mit Python 3.10 implementiert. Die vorherigen Komponenten wurden erweitert bzw. überarbeitet, die überarbeiteten Komponenten werden im Folgenden aufgezählt, sowie die jeweiligen Anpassungen für den spezifischen Prototypen, welcher davon betroffen ist. Für die QR-Codes wurde das Python-Package *Segno*[62] verwendet, dieses Package bietet viele Möglichkeiten, verschiedene QR-Codes in unterschiedlichen Bilderformaten, Skalierungen, Farben und anderen Eigenschaften zu erstellen. Für Operationen am Browser wurde das Paket *pyshorteners*[63] verwendet, um die URLS zu verkürzen und somit Platz in den QR-Codes zu sparen.

7.2.1 Backend

Server

Der Prototyp verwendet einen *SimpleHTTP-Server*, welcher als Directory einen separaten Ordner nutzt. Im Vergleich zum 5 wird hier nur ein Server verwendet und nicht mehr die ganze Festplatte zur Verfügung gestellt. Der Server ist noch nicht mit erweiterten Sicherheitsmaßnahmen ausgestattet, wodurch eine Verwendung dieses Prototypen außerhalb eines vertrauenswürdigen Netzwerks nicht empfohlen ist. Der Server ist bezüglich der Kapazität limitiert, das Übertragen einer Datei mit einer Größe von mehr als einem Megabyte kann zu Fehlern führen. Der Ordner, welcher vom Server benutzt wird, stellt die im Fenster aktuell geöffnete Datei zur Verfügung, und die angezeigte Datei kann von diesem Ordner aus heruntergeladen werden. Der Ordner stellt nur maximal jeweils eine Datei zur Verfügung und zwar die Datei, welche im aktuell aktiven Fenster geöffnet ist. Die Datei wird durch Funktionen im Hintergrund ermittelt und in den Ordner kopiert, solange das entsprechende Fenster aktiv ist. Sollte ein neues Fenster gewählt werden, wird die vorherige Datei aus dem Ordner gelöscht und die neue Datei wird eingefügt, sodass zu jedem

[62] `https://pypi.org/project/segno/`
[63] `https://pypi.org/project/pyshorteners/`

Zeitpunkt maximal eine Datei im Ordner ist[64].

7.3 Ermitteln der Dateien

Die Ermittlung der zu übertragenen Dateien wurde um neue Funktionen erweitert, unter anderem auch die Möglichkeit, einige ausgewählte Windows-Anwendungen zu erfassen[65]. Zusätzlich wurde eine Funktion eingefügt, mit welcher die Erfassung der URL in der Adressleiste eines aktuellen, aktiven Browserfenster möglich ist. Die Anwendung prüft im Verlauf des Prozesses die Fenstertitel und abhängig vom jeweiligen Namen im Titel werden verschiedene Mechanismen zum Erfassen des Inhaltes angewendet. Die Anwendung prüft bei jedem neu ausgewählten Fenster zuerst, ob es sich um einen Browser, eine Anwendung oder eine der spezifisch genannten Windows-Anwendungen handelt.

7.3.1 Browser

Die Anwendung prüft zuerst, ob es sich bei dem aktiven Fenster um einen Browser handelt, hierfür wird der Name des Fensters (welcher in der Regel auch den Namen des Browsers enthält) mit einer liste mit Browsernamen abgeglichen. Wenn eine Übereinstimmung vorliegt, wird im folgenden die URL der aktuell aktiven Website ermittelt. *Pywin32* bietet bei Browsern keine direkte Möglichkeit die URL zu ermitteln, weswegen ein Umweg über die Anwendung *Accessability-Insights*[66] gewählt wurde. Mithilfe von *Accessability-Insights* lassen sich die Namen der Elemente eines Fensters ermitteln, was bei den Browsern verwendet wird, um das jeweilige Element für die Adressleiste zu ermitteln, welche die URL der Website als Inhalt enthält. Der Name des Adressleistenelements wird von Package pywinauto[67], ein Package, welches die Automatisierung von Nutzeroberflächen erlaubt, verwendet, um den Inhalt des Elementes (Die URL) zu extrahieren. Dieser ermittelte Inhalt wird mit dem Package pyshorteners[68] in eine tinyurl[69] umgewandelt. Im Falle eines

[64] Bei URL's wird keine Datei kopiert, der Ordner zu solchen Zeitpunkten leer
[65] Darunter fallen die Apps *Fotos, Groove Music, Windows Media Player und Filme und TV*
[66] `https://accessibilityinsights.io/`
[67] `https://pypi.org/project/pywinauto/`
[68] `https://pypi.org/project/pyshorteners/`
[69] `https://tinyurl.com/app`

Browserfensters wird keine Datei kopiert, stattdessen wird die URL direkt als Inhalt des Markers gesetzt, der Webordner ist in diesem Fall nicht erforderlich.

7.3.2 Reguläre Anwendungen

Dateien, welche in einem Drittprogramm (wie z.B Adobe Acrobat, Imageviewer oder Notepad++) oder in anderweitigen Windows-Anwendungen geöffnet werden (Word, Powerpoint, Excel, Notepad) können mithilfe des *pywin32*-Packages und unter der Zuhilfenahme des Ordners *Recent*[70] unter Windows 10 erfasst werden. Der Recent Ordner enthält eine Übersicht der zuletzt geöffneten Dateien und Ordnern, hierbei handelt es sich bei den Dateien um Verknüpfungen, welche auf den eigentlichen Speicherort der Dateien verweisen. Zu Beginn werden von der Anwendung das aktive Fenster im Vordergrund und dessen Name in der Titelleiste ermittelt. Mit dem Inhalt des Recent Ordners wird eine Liste der aktuell dort enthaltenen Verknüpfungen erstellt und mit dem Fensternamen abgeglichen. Wenn eine Übereinstimmung gegeben ist, wird anhand eines Dispatchs, welcher unter Zuhilfenahme eines COM-Objektes[71] und der entsprechenden Funktion in pywin32 erfolgt[72] der Pfad zu der übereinstimmenden Datei ermittelt. Um mögliche Fehler durch eine doppelte Übereinstimmung (z.B wenn der Name der Datei auch gleichzeitig der Name eines Ordners ist) zu vermeiden, wird im Vorfeld des Kopiervorganges geprüft, ob sich unter den Funden Ordner befinden, diese werden daraufhin von der Funktion ignoriert. Wenn die übereinstimmende Datei gefunden ist, wird diese anschließend mithilfe eines OS-Befehls[73] in den Webserver-Ordner kopiert und dort so lange verwahrt, bis das betreffende Fenster gewechselt wird. Bei der Auswahl eines neuen Fensters wird, unabhängig ob ein Fund erfolgt, die vorherige Datei aus dem Ordner mithilfe eines OS-Befehls entfernt

[70] entweder erreichbar unter dem Pfad `C:\Users\Username\AppData\Roaming\Microsoft\Windows\Recent` oder mit der Kombination Windowstaste+R mit dem Wort "Recentäls Eingabe

[71] `https://learn.microsoft.com/en-gb/windows/win32/com/component-object-model--com--portal?redirectedfrom=MSDN`

[72] `http://timgolden.me.uk/pywin32-docs/html/com/win32com/HTML/QuickStartClientCom.html`

[73] `https://docs.python.org/3/library/os.html`

7.3.3 Windows- Anwendungen

Für den Fall, dass der Nutzer eine der folgenden Anwendungen (Fotos, TV und Filme, Groove Musik oder Windows Media Player) aktiv hat wurde ein separates Verfahren entwickelt, um den aktuell geöffneten Inhalt in diesen Anwendungen zu ermitteln. Dieses zusätzliche Verfahren ist deswegen erforderlich, da mit dem regulären Verfahren zwar der Name des Fensters ermittelt wird, bei diesen vier Anwendungen allerdings nur den Namen der Anwendung anzeigt, ohne dass sich aus dem Titel Rückschlüsse auf den Inhalt ziehen lassen. Die aktuell geöffnete Datei lässt sich aber dennoch ermitteln, da der eigentliche Titel der geöffneten Datei in einem Element im Fenster selber angezeigt wird. Mithilfe des schon bei Browsern verwendeten Ansatz mit Accessability Insights und pywinauto kann der entsprechende Dateititel aus dem Fensterelement extrahiert werden. Zusätzlich besteht bei diesen Anwendungen auch das Problem, das Dateien welche innerhalb der Anwendung geöffnet werden, nicht zwangsläufig im *Recent* - Ordner vermerkt werden, weswegen ein anderers Vorgehen zur Ermittlung der Datei erforderlich ist. Der Name der Anwendung bzw. des Fensters wird von *psutil* verwendet, um wie beim Proof of concept Prototypen den zur Anwendung zugeordneten Prozess zu ermitteln. Von diesem Prozess werden mit *psutil* die Namen der vom Prozess geöffneten file-handles[74] mit dem Titel der aktuell wiedergegebenen Datei abgeglichen, die file-handles enthalten den vollständigen Pfad zu der jeweiligen Datei. Wenn eine Übereinstimmung gegeben ist, so wird diese Datei in den Webordner kopiert.

7.4 Frontend

Das visuelle Design wurde im Vergleich zum Proof of Concept bezüglich der Reaktion auf Events und dem Aussehen des Markers angepasst.

7.4.1 Anpassung des visuellen Designs

Der ursprüngliche PDF-417 Barcode wurde, basierend auf den Ergebnissen der Anforderungserhebung, durch einen regulären, schwarz-weißen QR-Code ersetzt. Die-

[74] `https://learn.microsoft.com/en-us/windows/win32/fileio/file-handles`

ser QR-Code wird mithilfe der Anwendung *segno*[75] erstellt, welche es erlaubt QR-Codes in unterschiedlichen Größen, Formen und Farben zu erstellen. Als Inhalt des Markers wird erneut eine URL gesetzt, welche auf den Webordner verweist. Bei dieser URL handelt es sich jetzt um eine Adresse, welche sich aus dem *Rechner-IP*,*Serverport* und jetzt dem *Inhalt des Webordners* zusammensetzt, es wird kein direkter Dateipfad mehr als Adresse bereitgestellt. Weiterhin kann der Inhalt des Markers jetzt auch aus einer Tinyurl bestehen, wenn es sich bei dem aktuell offenen Fenster in einem Browser handelt, welcher auf die aktuell aktive Website verweist.

7.4.2 Angepasste Reaktion auf Events

Der Filter wurde um neue Events erweitert, mit welchen es auch möglich ist festzustellen, ob ein neuer Tab ausgewählt wurde. Hierfür wurde das Event für das Beenden eines Mausklicks (0x0009 bzw. 9) hinzugefügt. Wenn ein neues Fenster ausgewählt wird, wird der Fenstertitel des vorherigen Fensters jetzt mit dem neu ausgewählten abgeglichen. Wird ein Mausklick in einem Fenster ausgeführt, dessen Name sich am Ende des Events verändert hat (wenn man z.B im Texteditor ein neues Tab auswählt), so wird dies entsprechend registriert und als das aktive Fenster erkannt.

7.5 Die beiden Prototypenvariatne

Basierend auf den Erkentnissen der Anforderungserhebung im Kapitel6 wurden in dieser Iteration zwei Varianten dieses Prototypen implementiert. Diese beiden Prototypen sollten eine Antwort auf die Frage liefern, ob die Nutzer jetzt tatsächlich eine Anwendung bevorzugen, welche sie aktiv steuern und anpassen können, oder ob die Nutzer eine passive Anwendung bevorzugen, welche keinen zusätzliche Aufwand erfordert.

[75] `https://segno.readthedocs.io/en/latest/`

7.5.1 Erste Variante - permanenter Marker

Der Nutzer besitzt bei diesem Prototyp keine direkte Möglichkeit, den Marker zu steuern. Der Marker wird hier in einer skalierten Größe von 60x60 Pixeln angezeigt und reagiert auf Windows-Events. Die Größe wird hier bei der Skalierung eines Fensters nicht mehr verändert, um das Größenverhältnisse für eine Erfassung des Markers sicherzustellen.

7.5.2 Zweite Variante - Marker mit Hotkeys

Der Marker des zweiten Prototypen kann vom Nutzer in der Desktopumgebung mit der Maus an jede beliebige Position des Bildschirms bewegt werden. Der Marker besitzt in dieser Variante die Eigenschaft, durch Hotkeys seine Größe und Sichtbarkeit zu verändern. Der Marker ist zusätzlich mit einer *reset-funktion* ausgestattet, um den Code im Bedarfsfall an die 0,0 Koordinate des Hauptbildschirms zu platzieren, für den Fall, wenn der Nutzer den Überblick verliert, wo der Marker ursprünglich platziert wurde. Die Hotkey-Funktion wurde in diesem Prototypen mit dem python-package keyboard[76] implementiert.

7.6 Ergebnisse und weitere Entwicklung

Basierend auf den Ergebnissen der Anforderungserhebung wurde ein Prototyp implementiert, welcher in zwei Ausführungen vorliegt, eine, wo der Marker permanent eingeblendet ist und nicht gesteuert werden kann, und eine, wo der Marker durch Hotkeys und Maus gesteuert werden kann. Beide Varianten wurden anschließend in einer Nutzerstudie evaluiert. Die Nutzerstudie hatte das Ziel, sowohl Daten für die weitere Entwicklung zu sammeln, als auch die Frage zu beantworten, welche Art von Anwendung von den Probanden bevorzugt wird: eine Anwendung, welche für die Nutzer ohne Aufwand zu bedienen ist, oder eine Anwendung, die es den Nutzern erlaubt, die erforderlichen Funktionen selbst steuern zu können.

[76] `https://pypi.org/project/keyboard/`

8 Evaluation des ersten Prototypen

Entsprechend dem viertem Schritt (*Testing*) im Zyklus nach Norman (2013) wurde im Rahmen einer Nutzerstudie eine Evaluation der beiden Varianten des Prototypen durchgeführt. Basierend auf den Ergebnissen der Evaluation sollen damit weitere Features bzw. erforderliche Verbesserungen für die nächste Iteration des Prototypen generiert werden.

8.0.1 Probanden

Für die Studie wurden 5 Probanden aus dem privaten Umfeld rekrutiert. Drei der Testpersonen sind männlich, zwei weiblich. Der Altersbereich beträgt 24-50 Jahre, drei der Probanden waren Studenten unterschiedlicher Fachrichtungen, zwei der Probanden selbstständig berufstätig. Zwei der Probanden (Die Probanden P2 und P4) waren bereits bei der Anforderungserhebung 6 mitbeteiligt.

8.0.2 Studiendesign

Die Evaluation wurde als eine Nutzerstudie konzipiert und hatte die Erfassung von qualitativen und quantitativen Daten als Ziel. Die Studie bestand aus zwei Blöcken, im ersten Block konnten sich die Nutzer mit den Prototypen vertraut machen und mussten einen Task erledigen, wobei die benötigte Zeit und die Erfolgsrate erfasst wurde. Während der Einführungsphase hatten die Probanden bereits erste Möglichkeiten, ihre Eindrücke und Äußerungen im Rahmen von Thinking Aloud zu äußern. Der zweite Block bestand aus einem kurzen semi-strukturierten Interview, in welchem die Nutzer die Möglichkeit hatten, Feedback zu den Varianten zu äußern und auch eigene Verbesserungsvorschläge, Bedürfnisse oder Probleme aufzuzählen. Die Nutzerstudien wurden basierend auf den Empfehlungen aufgebaut, wie sie von Blandford et al. (2016) und Lazar et al. (2017) ausgesprochen werden.

Aufbau und Durchführung

Die Studie wurde mit den Probanden vor Ort auf einem *Acer Aspire V3-772G* mit einer Auflösung von 1920x1080 durchgeführt. Nachdem die Probanden begrüßt wurden, folgte eine Einführung bezüglich des Zwecks und dem weiterem Ablauf der Studie, welche insgesamt eine Dauer von Ca.30 - 45 Minuten pro Probanden hatte. Der erste Block war eine Einführungsphase, in welcher den Probanden jeweils eine Variante des Prototypen präsentiert wurde, zur Vermeidung von Sequenzeffekten wurde die Reihenfolge der Prototypen bei den Probanden variiert. Drei Probanden wurde zuerst der Permanente und dann der interaktive Prototyp präsentiert und getestet, bei zwei Probanden wurde die Reihenfolge umgekehrt. Die Probanden konnten in dieser Phase den Prototypen beliebig lange ausprobieren, während dieser Phase konnten die Probanden Eindrücke, Meinungen und Ideen äußern, zusätzlich wurden Beobachtungen über das Verhalten und die Vorgehensweisen der Probanden notiert. Nachdem die Probanden sich mit dem präsentierten Prototypen vertraut gemacht hatten, wurden diese angewiesen, einen Task zu erledigen. Hierbei wurden den Probanden 12 Dateien in unterschiedlichen Formaten präsentiert (.txt, URL,.jpg, .mov), die Nutzer hatten die Aufgabe, alle Dateien nacheinander zu öffnen und deren dazugehörigen Marker einzuscannen, hierbei wurden die benötigte Zeit und die Erfolgsrate notiert. Nachdem der Task abgeschlossen war, wurde das Vorgehen mit dem anderen Prototypen wiederholt. Nachdem beide Durchläufe abgeschlossen waren, folgte am Ende ein Interview mit den Nutzern. In diesem Interview wurden die Nutzer unter anderem bezüglich ihrer Präferierten *Prototypen-Varianten,Kritikpukte* und auch persönlcihe *Verbesserungsvorschlägen* befragt. Nachdem das Interview beendet war, wurde den Probanden gedankt und sie wurden verabschiedet.

8.0.3 Ergebnisse der zeitlich getimten Tasks

Für jede Variaton des Prototypen erfolgten Messungen, während der zeitlich getimte Task durchgeführt wurde. Es wurde die benötigte Zeit für die Erfüllung, sowie die Anzahl der Erfolgreich eingescannten Marker gemessen. Die Marker besaßen

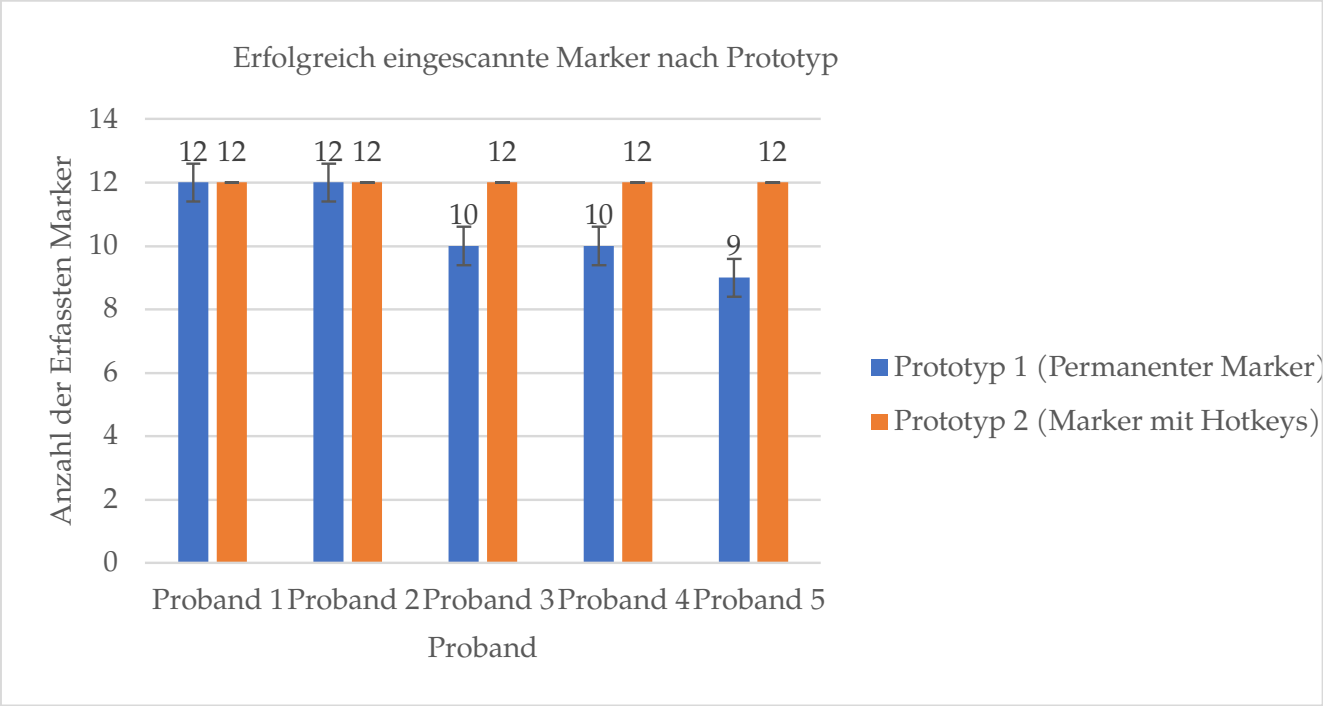

Abbildung 12: Die Erfolgsrate der Permanenten und der Hotkey Variante im Vergleich, insgesamt konnten bei dieser Aufgabe 12 Marker maximal eingescannt werden(eig.Abb)

alle unterschiedliche Inhalte und unterscheiden sich bezüglich der Länge ihres Inhaltes, da einige der Dateinamen länger waren als andere. Ein Marker galt hierbei als erfolgreich eingescannt, wenn auf dem Handy des Probanden der Marker erfasst und dessen Inhalt angezeigt wurde (also, in dem Fall die enthaltene URL)

Bei den Messungen ergab sich, dass bei der Hotkey-Variante die Probanden alle Marker erfolgreich einscannen konnten, während es bei der permanenten Variante, vor allem bei Markern mit längeren URL's (aufgrund längerer Dateinamen) bei Probanden wie P3-P5 zu Situationen führte, wo der Marker überhaupt nicht erfasst werden konnte, trotz Korrekturen des Aufnahmemodus oder der Distanz. Bei der benötigten Zeit ergaben sich Unterschiede bezüglich der Prototypen, generell brauchten die Probanden beim permanenten Prototyp länger (Geometrisches Mittel = 03:07:28) als bei der Hotkey-Variante (Geometrisches Mittel = 02:50:12). Diese Zeitersparnis lässt sich potentiell damit begründen, dass aufgrund der einstellbaren Größe die Zeit zum Erfassen des Markers verringert wird, allerdings wirken sich auch noch andere Faktoren aus, unter anderem die Verarbeitungsgeschwin-

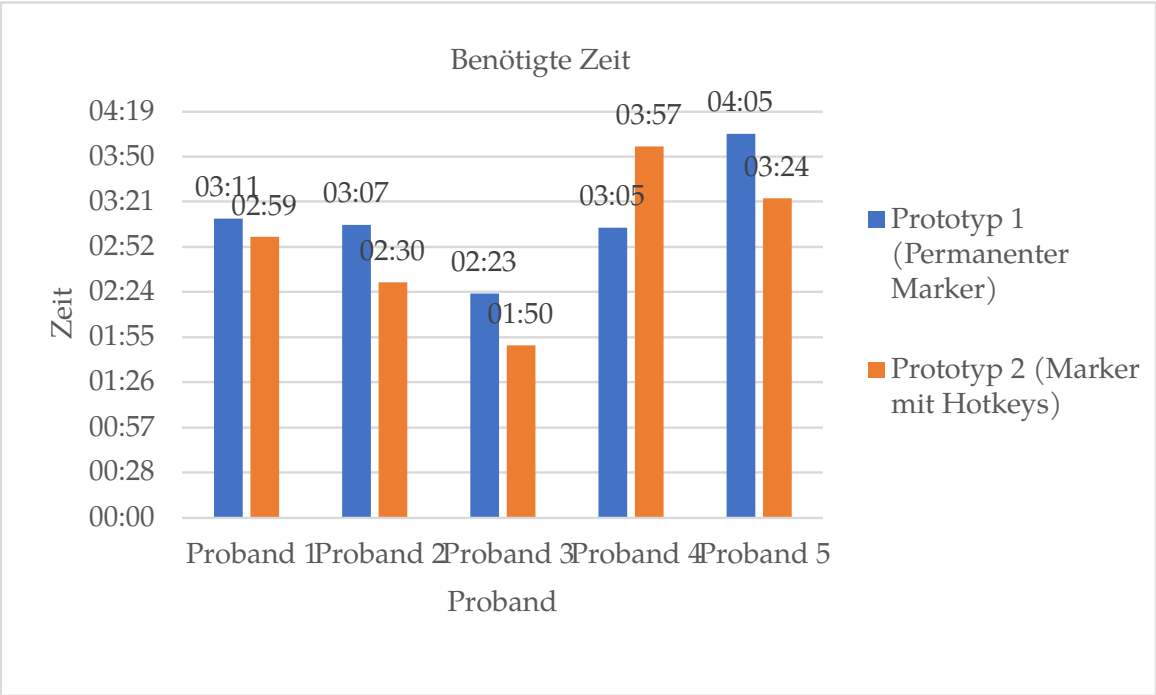

Abbildung 13: Die Erforderliche Zeit pro Proband(eig.Abb)

digkeit des Mobilgeräts selber, sowie die individuelle Geschwindigkeit der Probanden bei solch einer Aufgabe. Für diese Zeitmessung wurde das geometrische Mittel basierend, aufgrund der geringen Größe der Stichprobe (5 Probanden), weshalb die Messung des regulären Mittelwertes die Gefahr zu großer Abweichungen birgt (Sauro & Lewis (2012),S.30ff).

8.1 Ergebnisse der Interviews

Die Interviews wurden sinngemäß transkribiert, geglättet und kombiniert, mit den Notizen aus der Einführungsphase einer thematischen Kodierung unterzogen, wie sie im Kapitel6 vorgenommen wurde. Hierbei konnten die Themen *Präferenzen, Visuelle Gestaltung, Kognitive Problemfelder und Verbesserungsansätze der Probanden* ermittelt werden, welche im weiteren Verlauf zusammengefasst vorgestellt werden sollen.

Präferenzen

Von der Größe des Markers haben die Probanden keine expliziten Vorstellungen, die ideale Größe ist diejenige, welche es den Probanden erlaubt, die Aufgabe entsprechend erfolgreich durchzuführen. Die Platzierung des Markers wird an einer (für den Nutzer) konstanten Position in der Titelleiste des Fensters bevorzugt (die Genaue Lage variiert von Probanden zu Probanden). Beide Prototypen haben für die Probanden jeweils unterschiedlich positive Merkmale. Der passive Marker wird dahingehend positiv bewertet, dass sich dieser laut den Probanden, visuell akzeptabel in die Desktopumgebung einfügt. Zusätzlich besitzt der passive Marker den Vorteil, dass die Nutzer keinen zusätzlichen Aufwand bei der Bedienung betreiben müssen, da der Marker sich automatisch mit dem Inhalt des Fensters mitverändert, wodurch nur noch ein Einscannen erforderlich ist. Der Interaktive Marker hat dahingehend den Vorteil, dass sich mit dieser Variante die gewünschte Kontrolle über die Anwendung ausüben lässt und die Nutzer den Marker somit nach ihren individuellen Bedürfnissen anpassen und Bedienen können.

8.1.1 Kognitive Problemfelder

Die Hotkey-Variante stellt für die Nutzer in unterschiedlichen Ausmaßen einen zusätzlichen Aufwand dar. So gibt ein Teil der Nutzer an, mit dem Erlernen und Verwenden der Hotkey-Kombinationen keine Probleme zu haben (P3 und P5). Die Hotkey-Kombinationen werden von den Nutzern unterschiedlich beurteilt, aus der Sicht einiger Nutzer sind diese Kombination entweder zu kompliziert zu merken oder die Anzahl der Kombinationen (bei diesem Prototyp fünf Kombinationen), oder der Hotkey-Buttons (jeweils drei pro Kombination) überfordert die Nutzer beim Erlernen des Prototypen. Andere Nutzer hingegen beurteilen die ausschließliche Nutzung und auch das Erlernen der Hotkey Kombinationen als kein Problem für den weiteren Arbeitsablauf, jedoch wird im Vergleich zur Hotkey Variante dennoch die permanente Variante von allen Probanden als die bequemere der beiden beurteilt, bedingt durch die automatische Funktionsweise, ohne das der Marker extra eingeblendet oder bewegt werden muss.

8.1.2 Visuelle Gestaltung

Die Größe des permanent eingeblendeten Markers wird von den Probanden als zu klein eingestuft, weswegen beim Einscannen entweder ein nahes Rangehen an den Bildschirm erforderlich ist oder das Einscannen nur unter erschwerten Bedingungen (oder gar nicht) funktioniert. Die Problematik mit der Größe wird vor allem bei der Verwendung verschiedener Smartphones und QR-Reader-Anwendungen ersichtlich, so erlebten die Probanden P1,P3,P5 mit ihren verwendeten Smartphones Probleme beim Einscannen der Marker. Bei den verbliebenen Probanden P1 und P2 hingegen wurden der permanente Marker ohne Probleme erfasst. Eine Präsenz des Markers wird von einem Teil der Probanden als störend und ablenkend eingeschätzt.

Verbesserungsansätze der Probanden

Eine vorgeschlagene Verbesserung ist eine generelle Vergrößerung des Markers oder die Möglichkeit sich die Markergröße individuell einstellen zu können. In diesem Kontext verweisen die Nutzer auf bereits bekannte Methoden wie ein Anklicken des Markers mit der Maus, eine Zoomfunktion oder eine Steuerung des Markers über ein Kontextmenü mit verschiedenen Optionen. Weitere vorgeschlagene Optionen sind eine Ein- und Ausklappfunktion des Markers oder eine Möglichkeit, sich den Marker entsprechend über ein Menü konfigurieren zu können. Um für die Nutzer den Aufwand durch die Verwendung von Hotkeys zu reduzieren, werden als Maßnahmen unter anderem eine Verringerung der Hotkey-Kombinationen (auf z.b zwei statt drei Tasten) oder ein Zusammenlegen der Funktionen (sodass insgesamt weniger Funktionen erforderlich sind) empfohlen. Eine alternative Maßnahme, die empfohlen wird, ist den Nutzern die Möglichkeit zu gewähren, sich die Hotkey-Kombinationen selber konfigurieren zu können, sodass jeder seine eigenen Präferenzen auswählen kann.

8.2 Diskussion und Bedeutung für die weitere Entwicklung

Durch die Evaluation konnten mehrere Kritikpunkte an den aktuellen Entwicklungsansätzen entdeckt werden. Das zentralste Ergebnis der entwicklung war die feststellung, dass die Probanden eine Anwendung bevorzugen, welche wenig Aufwand von ihrer Seite erforder, allerdings genügend Kontrolle über bestimmte Funktionen gewährt wird, sodass sich die Nutzer alles nach eigenem Belieben konfigurieren können. Es wurden für die weitere Iteration die folgenden Features ermittelt

8.2.1 Konfigurationsmöglichkeit für Hotkeys und Marker

Basierend auf den Problemen, welche die Nutzer mit dem permanenten Marker (zu kleine Größe) und mit den Hotkeys hatten (entweder zu viele Kombinationen oder Kombinationen zu kompliziert), bietet es sich im folgenden an den Nutzern, mithilfe eines Konfigurationsmenüs, die Möglichkeit einzuräumen, die Anwendung nach Bedarf konfigurieren zu können. Hierbei soll den Nutzern die Möglichkeit gewährt werden, den Marker und auch die Steuerung per Hotkeys, je nach Eigenbedarf konfigurieren zu können. Zusätzlich soll den Nutzern die Möglichkeit gewährt werden, sich auch das geplante Passwort selber einstellen zu können. Dies entspricht in etwa den Verbesserungsvorschlägen der Nutzer, womit sowohl eine Möglichkeit gegeben ist, den Marker vergrößern zu können, als auch die Hotkeykombinationen entsprechend anpassen zu können.

8.2.2 Kombination der Prototypen

Da die Nutzer bei beiden Prototypen jeweils unterschiedliche Aspekte (der geringe Aufwand beim passiven Marker, die Kontrollmöglichkeit beim Interaktiven Marker) bevorzugen bzw. ablehnen (zu kleine Größe des passiven Markers bzw. zu hohe Komplexität einer rein Keyboardbasierten Steuerung), sollen im folgenden beide Prototypen in ihrer Funktionalität kombiniert werden. Der nachfolgende Prototyp soll folglich den Marker in die Desktopumgebung permanent einblenden, jedoch sollen die Nutzer jetzt die Möglichkeit haben, den Marker entsprechend anpassen zu können, bezüglich Größe und Position. Den Nutzern soll dann jetzt auch zusätz-

lich mithilfe der Maus die Möglichkeit eingeräumt werden, den Marker anpassen zu können, was jenen Nutzern entgegen kommt, für welche eine rein Hotkeybasierte Steuerung einen zu großen Aufwand darstellt. Um den Marker anzupassen, soll hierfür ein Konfigurationsmenü eingerichtet werden, mit welchen die entsprechenden Eigenschaften eingestellt werden können.

9 Zweiter Prototyp

Ausgehend von der Ergebnissen der Evaluation im Kapitel8 wurde ein weiterentwickelter Prototyp implementiert. Dieser Prototyp verwendet das Backend der vorherigen Iteration, kombiniert jetzt aber Funktionalitäten der beiden vorherigen Prototypen-Varianten. Dies bedeutet, dass der Marker jetzt permanent in der Umgebung integriert ist und die Nutzer die Möglichkeit haben, den Prototypen mit der Maus zu bewegen, durch Hotkeys zu vergrößern und auch bei Bedarf ein- und auszublenden. Die Anwendung wurde um ein Konfigurationsmenü und ein Kontextmenü für den Marker erweitert. Der lokale Server wurde durch einen neuen Server ersetzt, welcher grundlegende Sicherheitsmaßnahmen wie Passwörter und SSL-Verschlüsselungen implementiert. Im nachfolgenden sollen die neuen bzw.überarbeiteten Features des Prototypen vorgestellt werden.

9.1 Backend

Das Backend der Anwendung wurde um einen neuen Webserver erweitert. Die Erfassung der Dateien funktioniert nach demselben Prinzip wie es bereits im 7 vorgestellt wurde. Der vorher verwendete *SimpleHttpServer* wurde in dieser Iteration durch einen Twisted-Webserver ersetzt, welcher durch das Python-Package Twisted[77] bereitgestellt wird.

9.1.1 Server

Mithilfe eines Twisted[78] Webservers können die Sicherheitsprobleme des vorher verwendeten *simplehttpServer* eliminiert werden, wie in den Kapiteln 5 und 7 angesprochen wurde. *Twisted* erlaubt die Übertragung beliebig großer Dateien und

[77] `https://pypi.org/project/Twisted/`
[78] `https://twisted.org/`

stellt Möglichkeiten zur Verfügung Sicherheitsmaßnahmen zu implementieren. Der Server wurde basierend auf der Dokumentation mit einer Authentifikation ausgestattet[79], welche einen Usernamen und ein Passwort erfordert. Der Username und das Passwort werden in einem separaten File gespeichert, wenn der Nutzer eine Anfrage an den Webserver stellt, indem er eine Datei herunterlädt, werden die entsprechenden Credentials (Username und Passwort) vom Nutzer gefordert. Wenn eine Übereinstimmung gegeben ist, kann die Datei heruntergeladen werden. Der Webserver wurde ebenfalls mit der Möglichkeit ausgestattet ein SSL (Secure Socket Layer) bzw. ein TLS (Transport Socket Layer) - Zertifikat einzubinden.

9.1.2 Verschlüsselung per SSL/TLS

Mithilfe eines SSL/TLS - Zertifikates[80] kann die Verbindung zwischen dem Server und dem Nutzer verschlüsselt werden, wodurch eine gesicherte Übertragung über *https* möglich ist. Basierend auf der Dokumentation[81]und dem Python-Package pyOpenSSL[82] wurde der Server mit der Möglichkeit ausgestattet ein *SSL/TLS*-Zertifikat einzubinden. Das Zertifikat wurde mit der Anwendung *mkcert*[83] erstellt und selbst signiert, weswegen es zu Problemen bei Browsern kommen kann, da es sich bei diesem Zertifikat um kein offizielles Zertifikat handelt, welches von einer Vertrauenswürdigen Drittstelle ausgestellt wurde. Die Grundlegenden Funktionen und der Aufbau eine Übertragung über HTTPS ist aber trotzdem möglich.

9.2 Frontend

Die visuelle Gestaltung des Markers aus der vorhergehenden Iteration wurde beibehalten, die Standardgröße für den Marker wird mithilfe der *GetSystemMetrics*-

[79] `https://docs.twisted.org/en/stable/core/howto/cred.html`
[80] SSL ist hierbei eine ältere Version und der Ursprüngliche Name des Protokolls, welches in TLS umbenannt wurde und eine Weiterentwicklung des SSL-Protokolls darstelltTurner (2014). Auf die genauen Unterschiede soll nicht eingegangen werden, da es nicht im Fokus der Arbeit steht, SSL hat sich aber als eine Bezeichnung überlebt, weswegen man sich nicht von SSL und TLS verwirren lassen sollte – es ist dasselbe Protokoll
[81] `https://docs.twisted.org/en/stable/core/howto/ssl.html`
[82] `https://pypi.org/project/pyOpenSSL/`, dies ist laut der Dokumentation ein erforderliches Package, um eine solche Verschlüsselung einzurichten
[83] https://github.com/FiloSottile/mkcert

Funktion[84] ermittelt, welche es der Anwendung erlaubt, auf Systemmetriken zuzugreifen. Als Metrik wird hier die *SMCYSIZE*[85] verwendet, um den Marker möglichst passend[86] in die Titelleise zu integrieren, eine Einscanbarkeit ist mit dieser Größe aber nicht automatisch gegeben[87]. Die Nutzer können jetzt mit einem Rechtsklick auf den Marker ein Kontextmenü aufrufen, welches ihnen weitere Interaktionen mit dem Markern erlaubt und dieselben Funktionen bietet, wie die Hotkeys der Anwendung. Die Anwendung wurde ebenfalls um ein Konfigurationsmenü erweitert, welches über ein Icon im Systemtray aufgerufen werden kann. Der Marker behält seine bisherigen Eigenschaften, wie z.B ein Verschieben mit der Maus oder die Reaktion auf Systemevents bei. Die folgenden Menüs wurden unter Zuhilfenahme des bereits im 5 vorgestellten *pyqt-Toolkits* implementiert

9.2.1 Kontextmenü für den Marker

Um den Nutzern die Möglichkeit zu bieten, die Markergröße auch ohne Hotkeys verändern zu können, wurde dem Marker ein Kontextmenü hinzugefügt, welches mit einem Rechtsklick auf den Marker aufgerufen werden kann. Dieses Kontextmenü erlaubt es den Nutzer, die gleichen Aktionen auszuführen, welche der Hotkey Variante der vorherigen Iteration zur Verfügung standen. Als Optionen können die Nutzer den Marker *ausblenden*[88], *vergrößern*[89], *auf die Standardgröße setzen* und einen *Kompaktmodus aktivieren*[90]. Das Menü wurde so gestaltet, dass die zugewiesenen Hotkeys für die jeweilige Aktion ebenfalls angezeigt werden, basierend auf der empfehlung von (Shneiderman et al. (2018),S.279) dem Nutzer wird somit die Alternative geboten, die Steuerung entweder per Menü vorzunehmen oder

[84] `https://learn.microsoft.com/en-us/windows/win32/api/winuser/nf-winuser-getsystemmetrics`

[85] Diese Metrik gibt die Höhe eines Buttons in der Titelleiste an, also z.B den Minimieren oder Schließen-Button

[86] Bei Fenstern, die eine selbst gestaltete Fensterdekorationen haben, kann es dazu kommen, dass der Marker entweder zu klein oder zu groß für die Titelleiste ist

[87] Die Höhe kann von System zu System abweichen, das System, auf dem diese Anwendung im laufe der Arbeit implementiert wurde, gibt eine Größe von 22 Pixel aus

[88] Dafür muss man sich dann die Kombination zum Einblenden merken oder es alternativ mithilfe des Konfigurationsmenüs wieder einblenden

[89] Auf das doppelte,fünffache oder Zehnfache der aktuellen Größe

[90] Diese Option erlaubt es dem Nutzer, den Marker auf eine Größe von 15*15Pixel zu minimieren, eine Option die für jene Nutzer gedacht ist, die eine permanente Präsenz eines großen Markers als zu störend empfinden

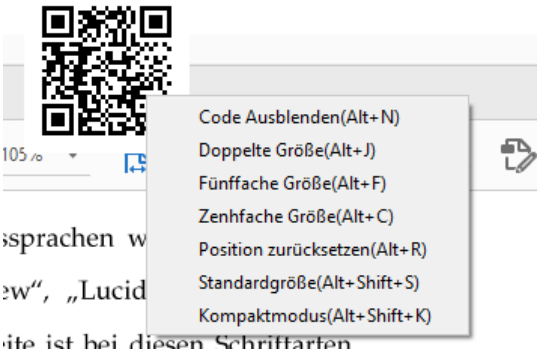

Abbildung 14: Das Kontextmenü, welches mit einem Rechtsklick auf den Marker angezeigt wird(eig.Abb)

Abbildung 15: Das Icon im Systemtray in Form eines QR-Codes, womit der Nutzer auf das Konfigurationsmenü zugreifen kann(eig.Abb)

die Aktionen durch Hotkeys auszuführen. Die Anzeige der Hotkeys wurde basierend auf dem Feedback der Nutzer eingefügt, um die Erlernbarkeit der Hotkey-Kombinationen zu erleichtern.

9.2.2 Konfigurationsmenü

Als ein neues Feature wurde ein Konfigurationsmenü implementiert, welches den Nutzern erlaubt, die Standardgröße des Markers, die Hotkey-Kombinationen und das Passwort der Anwendung zu konfigurieren. Um keine zusätzlichen Konfigurationselemente am Marker oder im Kontextmenü einblenden zu müssen, wurde ein Icon im Systemtray implementiert, mit dem das Konfigurationsmenü aufgerufen werden kann. Mit einem Rechtsklick auf dieses Icon kann das Konfigurationsmenü aufgerufen, die Anwendung beendet oder der Marker ein- und ausbgeblendet werden. Die folgenden Konfigurationsmöglichkeiten sind im Menü möglich. Aufgrund der Vertrautheit der Nutzer mit dieser Art von Menüs wurde das Konfigurationsmenü als ein *Two-Panel Selector* designt (TIDWELL (2011),S.198 ff.)

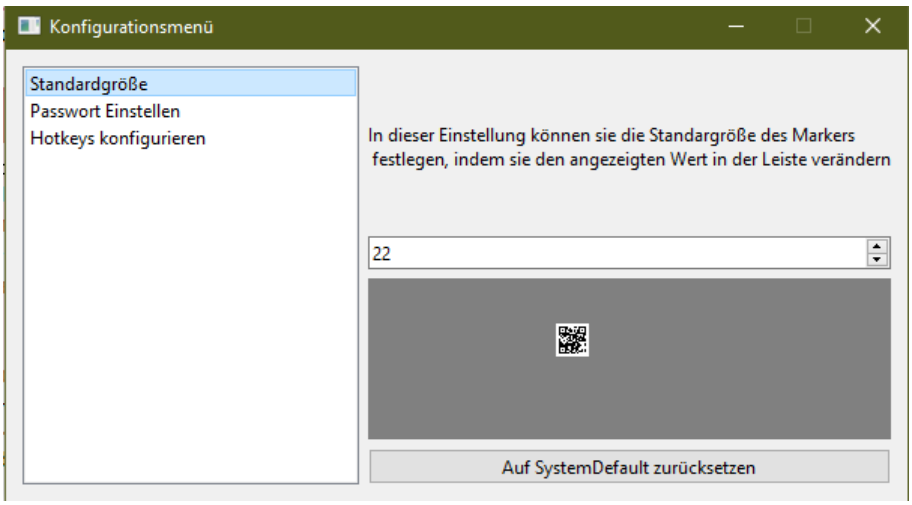

Abbildung 16: Das Konfigurationsmenü für die Markergröße mit Previewfenster(eig.Abb)

Standardgröße für Marker

In diesem Untermenü lässt sich die Standardgröße des Markers einstellen, in welcher der Marker bei der Auswahl eines Fensters eingeblendet wird. Das Menü bietet die Möglichkeit, die Standardgröße des Markers anzupassen, die Änderungen werden sofort übernommen und erfordern keine zusätzlichen Bestätigungen. Der Nutzer hat bei der Anwendung die Möglichkeit, die zukünftige Größe in einem Preview-Fenster betrachten zu können. Der auf dem Desktop bereits eingeblendete Marker wird nicht verändert, dafür ist ein erneutes Einblenden erforderlich. Wenn der Nutzer die vom System vorgegebene Default-Größe (Die CMYSIZE-Metrik) wiederherstellen möchte, so kann dafür ein Button genutzt werden, welcher die den Marker auf die Größe der bereits im Abschnitt erwähnten Systemmetrik setzt.

Passwort verändern und anpassen

Dieses Menü bietet die Option sich ein neues Passwort für die Anwendung zu setzen. In der Aktuellen Version dieser Anwendung wird nur die Möglichkeit geboten, das Passwort zu setzten der Username[91] bleibt hierbei unverändert. Um ein neues Passwort zu setzen, muss der Nutzer ein neues Passwort eingeben und dieses bestätigen (sowohl durch eine Wiederholung, als auch per Button). Beim Betätigen des

[91] üser"

Buttons "neues Passwort setzenërfolgt ein Check, welcher die beiden Eingaben miteinander abgleicht, bei einem Fehler wird eine Fehlermeldung generiert. Sollte der Prozess ordnungsgemäß ablaufen, wird im "Passwort-File", in welchem die Daten gespeichert sind, entsprechend aktualisiert (user:neues Passwort).

Hotkeys konfigurieren

Diese Kategorie bietet die Möglichkeit, die Tastenbelegung für die jeweiligen Aktionen einsehen und ändern zu können. Die Tastenkombinationen sind in einer Konfigurationsdatei abgespeichert, welche die jeweiligen Kombinationen für die jeweilige Aktion enthält. Beim Start des Programms werden die initial vorhandenen Hotkey Kombinationen aus der Datei ausgelesen und mithilfe der Anwendung Keyboard als globale Kombinationen gesetzt[92]. Wenn ein Nutzer die Kombination ändern möchte, muss im Menü der entsprechende Button für die Aktion betätigt werden. Dies öffnet daraufhin ein ein neues Fenster, wo der Nutzer eine beliebige Tastenkombination setzen kann, diese Kombination muss dabei aus mindestens 2 Tasten bestehen. Wenn eine neue Kombination erfolgreich gesetzt wurde, wird diese Kombination in der Konfigurationsdatei gespeichert (die alte wird dabei überschrieben) und mithilfe eines *PYQT-Signals*[93] weitergeleitet. Die Weiterleitung ist erforderlich, da die entsprechenden Keyboard-Methoden aktualisiert werden müssen. Das Signal erlaubt es, Events von einem PYQT-Komponenten (in diesem Fall das Menü wo die Kombination gesetzt wird) an eine andere Komponente weiterzuleiten (an die zuständige Methode, welche den Marker und die Hotkeys initialisiert). Dieses Signal enthält die neue Kombination und die nötigen Parameter, mit denen die alte Kombination im File ausgelesen werden kann. Mithilfe einer Methode des Keyboard-Packages wird die alte Kombination entfernt und durch die neue ersetzt.

[92] Es kann somit, je nach Kombination, zu einer Überschneidung mit anderen, globalen Tastenkombinationen kommen. Dieser globale Ansatz wurde deswegen gewählt, weil die nativen Methoden von PYQT nur funktionieren würden, wenn das Fenster (der Marker) aktuell im Fokus des Systems steht, also aktiv ausgewählt ist

[93] `https://doc.qt.io/qtforpython/PySide6/QtCore/Signal.html`

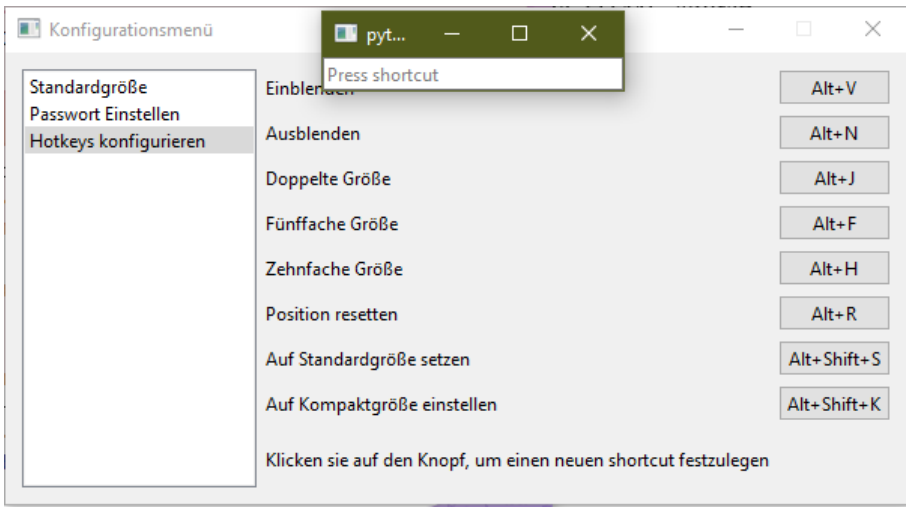

Abbildung 17: Konfigurationsfenster für Hotkeys, das kleinere Fenster gibt den Prompt an und zeigt auch die betätigte Tastenkombination an(eig.Abb)

9.2.3 Ergebnisse und weitere Entwicklung

Basierend auf Ergebnissen der Anforderungserhebung und der ersten Evaluation, wurde ein fortgeschrittener Prototyp entwickelt, welcher alle relevanten Funktionen aus den vorhergegangenen Schritten implementiert. Dieser Prototyp blendet den Marker permanent in der Desktopumgebung ein, bietet den Nutzern aber die Möglichkeit, den Marker bezüglich der Größe und die Anwendung bezüglich der Steuerung konfigurieren zu können. Der Anwendung wurden neue Menüs zugefügt, mit welchen die Nutzer die Marker, Hotkeys und Passwörter nach ihrem eigenen Bedarf konfigurieren können. Der Prototyp wurde im folgenden in einer Nutzerstudie getestet und evaluiert.

10 Evaluation des zweiten Prototypen

Nach der Implementierung des zweiten Prototypen wurde eine erneute Nutzerstudie zur Evaluation angesetzt. Um eine Vergleichbarkeit mit den vorangegangenen Prototypen herstellen zu können, wurde die Evaluation aus 8 wiederholt. Ziel der Studie war wieder eine Erfassung von Qualitativen und Quantitativen Daten für zukünftige Weiterentwicklungen der Anwendung sowie eine Erprobung der neu Implementierten Features und Funktionen. In dieser Evaluation testeten die Probanden den Prototypen, erfüllten eine getimte Aufgabe und wurden in einem Anschließenden Interview nach ihren Eindrücken befragt.

10.0.1 Probanden

Für die Evaluation wurden fünf Probanden rekrutiert, zwei aus dem privaten Umfeld und drei aus dem universitären Umfeld mithilfe von Aufrufen. Zwei der Probanden sind weiblich und drei Probanden männlich. Alle fünf Probanden waren Studenten verschiedener Fachrichtungen, wovon drei der Probanden aus der Fachrichtung der Medieninformatik waren. Die Probanden P1 und P4 waren in vorherigen Schritten mitbeteiligt, P1 bei der Anforderungserhebung, P4 in der Anforderungserhebung und der Evaluation des ersten Prototypen.

10.0.2 Studiendesign

Die Studie wurde nach dem gleichen Prinzip wie die Evaluation in Kapitel 10 durchgeführt und war in zwei Blöcke unterteilt. Die erste Phase bestand aus einer Einführungsphase, in welcher sich die Probanden mit dem Prototypen vertraut machen konnten und die Möglichkeit hatten, den Prototyp bezüglich der eigenen Präferenzen zwecks Hotkey-Kombinationen, Passworts und Standardgröße des Markers zu konfigurieren. Im Anschluss an diese Phase wurde der Task aus der vorherigen Eva-

luation wiederholt, in welchem die Probanden so schnell wie möglich Dateien einscannen mussten, die Anzahl und die Art der Dateien blieb unverändert (12 Dateien, in den Formaten .txt,.jpg,.mov und URL's). Als letzter Block folgte erneut ein semistrukturiertes Interview, hierbei wurden die Probanden nach ihren *Eindrücken zu dem Prototypen, bevorzugten Interaktions und Konfigurationsmethoden, bevorzugten Vorgehensweisen zur Übergabe sensibler Dateien und Vorschlägen zur Anpassung* befragt.

10.0.3 Ursprüngliche Planung

Ursprünglich war geplant, die Evaluation mithilfe einer Tagebuchstudie durchzuführen, ums sowohl die Performance, als auch die Häufigkeit der der Nutzung der Anwendung zu ermitteln. Dieses Vorgehen wurde aufgrund des zeitlichen Umfangs der Arbeit durch eine Nutzerstudie ersetzt. Idealerweise wäre für die Evaluation dieser Anwendung eine Tagebuchstudie mit einer längeren Dauer empfohlen (so ca. ab einem Monat aufwärts), um die Daten über eine längere Zeit zu sammeln. Hierbei soll eine längere Dauer nicht nur mehr Messungen generieren, sondern auch gleichzeitig zur Beobachtung eines Trends dienen, ob die Anwendung das Potential besitzt, auch tatsächlich als eine alternative von den Nutzern zur Datenübertragung genutzt zu werden, im Vergleich zu etablierten Methoden, wie eine Übertragung per Kabel, E-Mail oder Cloud-Dienst.

10.0.4 Aufbau und Durchführung

Die Studie wurde vor Ort im Labor auf *Acer Aspire V3-772G* mit einer Auflösung von 1920x1080 durchgeführt. Nach einer Begrüßung und Einweisung der Probanden erfolgte eine Einführungsphase, in welcher die Probanden diesmal nur einen Prototypen ausprobieren konnten. Den Probanden wurde auch die Möglichkeit gegeben, die Menüs auszuprobieren und den Prototypen entsprechend ihrer Eigenen Bedürfnisse zu konfigurieren. Während dieser Phase wurden die Aktionen der Probanden beobachtet und ihre Eindrücke bzw. bereits auftauchende Vorschläge, sowie ihre Aktionen notiert. Nach Abschluss der Einführung wurde erneut der zeitlich getimte Task eingeleitet. Die Probanden hatten erneut die Aufgabe, die verschiedenen

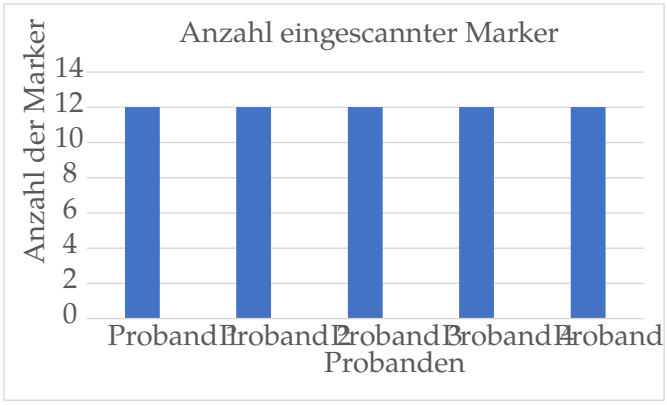

Abbildung 18: Die Erfolgsrate beim der zweiten Iteration des Prototypen, insgesamt konnten bei dieser Aufgabe 12 Marker maximal eingescannt werden(eig.Abb)

Files so schnell wie möglich einzuscannen. Im darauf folgenden Interview wurden die Probanden nach ihren Eindrücken bezüglich des Prototypen befragt. Zusätzlich wurden in diesem Interview die Probanden nach ihren *bevorzugen Interaktionsmethoden* (*Hotkey oder Maus*) und *Konfigurationsmethoden*(*Optionsmenü vs. Konfigurationsdatei*) und *möglicher Verbesserungen* befragt. Als letzte volgte die Befragung bezüglich der Art und Weise wie die Probanden sensible bzw. wichtige Dateien üblicherweise übertragen und wie sie mit diesem Prototypen das unbefugte einscannen durch Dritte verhindern würden. Nach Abschluss des Interviews wurde den Probanden für ihre Teilnahme gedankt und sie wurden verabschiedet

10.0.5 Ergebnisse der zeitlich getimten Aufgaben

Wie bereits im Kapitel 8 beschrieben, wurde für die Evaluation in dieser Studie derselbe zeitlich getimte Task mit denselben Dateien wiederholt, dabei wurden wieder die Zeit und Erfolgsrate gemessen.

Bei diesem Durchlauf konnten die Probanden sämtliche Marker erfolgreich einscannen, aufgrund der Konfigurationsmöglichkeiten, welche die Anwendung bietet. Durch die Bereitstellung der Konfigurationsmöglichkeiten konnten Faktoren wie z.B die Kameraqualität des verwendeten Mobilgeräts eliminiert werden, da die Nutzer jetzt die Möglichkeit haben, die Größe des Markers beliebig anpassen zu können. Zeitlich gesehen (Geometrisches Mittel = 03:19:52) sind die Probanden hier, im Vergleich zu den vorherigen Varianten, langsamer als bei der ursprünglichen

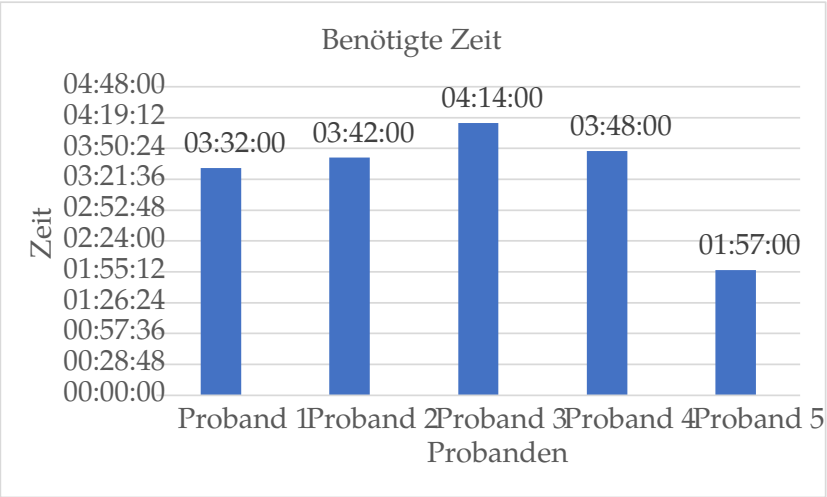

Abbildung 19: Die Erforderliche Zeit pro Proband(eig.Abb)

Hotkeyvariante8 und der permanenten Variante10. Weitere Faktoren, welche in zukünftigen Evaluationen untersucht werden sollten, sind die Fragen, inwieweit die individuelle Reaktionszeiten und auch die Verarbeitungsgeschwindigkeit der Scans auf dem Mobilgerät (also, wie lange es dauert, bis der Marker erfasst und dessen Inhalt ausgelesen wird) Einfluss auf die benötigte Zeit nehmen. Da die Stichprobe hier nur 5 Probanden betrug wurde wieder das Geometrische Mittel bei der Zeit verwendet (Sauro & Lewis (2012),S.30ff.)

10.1 Ergebnisse

Die aufgezeichneten Interviews wurden sinngemäß transkribiert, geglättet und mit den Notizen aus der Einführungsphase einer thematischen Kodierung unterzogen. Als übergeordnete Themen konnten *Problemfelder, Meinung zu Features, Interaktionsmethoden, vorgeschlagene Anpassungen und bevorzuge Konfigurationsmethode, visuelle Gestaltung, Sicherheitsaspekte* ermittelt werden. Die Themen werden im folgenden zusammengefasst vorgestellt.

Problemfelder der Anwendung

Ein wiederkehrendes Problem waren Bugs beim Aufrufen der Menüs innerhalb der Anwendung, so wurde das Konfigurationsmenü in mehreren Fällen entweder nach

einer merklichen Verzögerung oder gar nicht aufgerufen, sodass mehrfaches Klicken bzw. ein Neustart der Anwendung erforderlich war. Bei der Probandin P5 ergab sich ebenfalls ein Fehler bei der Erfassung durch die Kamera, da die interne Anwendung die Marker nicht erfassen konnte, nach dem Download einer externen Anwendung funktionierte das Einscannen jedoch ohne weitere Probleme. Die Nutzer bewerteten die Auswahl an Optionen unterschiedlich, Probanden wie P3 beurteilten die Fülle negativ und sprachen sich für eine Verringerung der Optionen aus. Die Positionierung und die Sichtbarkeit des Markers waren ebenfalls weitere Kritikpunkte an der Anwendung. Von der Positionierung her kritisierten die Nutzer die Tendenz zum Ümherspringen"des Markers zwischen den Fenstern, was sich auf den Arbeitsablauf störend auswirken könnte, weswegen eine durchgehend konstante Position bevorzugt wird. Von der Problematik der Sichtbarkeit ergibt sich, wie in der vorhergegangenen Evaluation, eine Aufteilung der Nutzer in jene, welche sich an einer permanenten Präsenz stören und jene, die es ignorieren.

Meinung zu Features

Die Probanden beurteilen die Anwendung als eine einfachere, weniger aufwändige Alternative für eine Dateiübertragung vom PC auf das Smartphone. Die Performance bei der Dateiübertragung, sowie das Prinzip, dass lediglich das Dateifenster angeklickt werden muss, wird ebenfalls positiv aufgefasst. Die Authentifikation per Passwort und die Verschlüsselung durch ein SSL/TLS-Zertifikat werden als weitere positive Aspekte betrachtet, wodurch auch ein Sicherheitsgefühl gegeben ist und durch die Passwortabfrage auch eine Sicherheit gegenüber dem unbefugten Einscannen durch Dritte besteht. Ein Nebeneffekt, welcher sich im Laufe der Evaluation ergeben hat, war die Tatsache, dass die User sich nur einmalig Authentifizieren mussten. Sofern das Passwort nicht im weiteren Verlauf geändert wurde, hat es ausgereicht, wenn sich die User einmalig eingeloggt haben, da jeder darauf folgende Download keine weitere Authentifizierung gebraucht wurde, ein Fakt, welcher positiv mit aufgenommen wurde.

Bevorzuge Konfigurationsmethoden

Bei der bevorzugten Konfigurationsmethode wurden die Nuzer befragt, ob sie diese (und andere Anwendungen) bevorzugt entweder über ein eingebautes Konfigurationsmenü oder unter Zuhilfenahme von Konfigurationsdateien einstellen. Die bevorzugte Konfigurationsmethode der Nutzer ist die Verwendung eines Optionsmenüs, diese Möglichkeit wird als bequemer und weniger Fehleranfällig wahrgenommen. Konfigurationsdateien haben laut den Nutzern den Nachteil, dass diese mit mehr Aufwand verbunden sind (wie z.B die erstmalige Suche nach der entsprechend zuständigen Datei, vor allem wenn es mehrere sind) und mehr Möglichkeiten für Fehler bieten (z.B wenn der Nutzer eine falsche Syntax verwendet).

Interaktionsmethoden

Die bevorzugte Interaktionsmethode ist abhängig von der individuellen Präferenz und der Erfahrung, welche die Nutzer mit dem jeweiligen Programm haben. So bevorzugen die Nutzer, wenn sie Erfahrung mit dem Programm haben, die Verwendung von Hotkeys. Hotkeys besitzen den Vorteil, dass diese sowohl bequemer, als auch schneller sind und man somit auch eine Zeitersparnis erreichen kann. Wenn ein Nutzer das Programm allerdings zum ersten Mal bedient oder wenn das Programm bestimmte Arten von Aufgaben erfordert (wie z.B eine Navigation), dann wird auch die Maus als eine bevorzugte Alternative verwendet. Sollte eine Funktion häufig genutzte Anwendungen haben, so neigen die Nutzer eher dazu, Hotkeys zu verwenden, letzten Endes ist es aber bei dem jeweiligen Programm auch immer eine Frage der persönlichen Präferenz.

Vorgeschlagene Anpassungen

Vorgeschlagene Verbesserungen am Prototypen beinhalten eine Erweiterung der Interaktionsmöglichkeiten und eine Anpassung des Markerdesigns. So wird empfohlen, eine Option einzufügen, den Marker ohne das direkte Öffnen der Datei (z.B durch eine direkte Auswahl oder per Rechtsklick) einblenden zu können, sodass die Nutzer sich den Schritt zum Öffnen der Datei sparen. Als ein weiteres Feature

wurde vorgeschlagen, die Übertragungsfunktion auch auf Ordner auszuweiten, sodass nicht nur einzelne Dateien, sondern auch bei Bedarf ganze Ordner oder Teile des Ordners auf einmal übertragen werden können. Weiterhin empfehlen die Probanden, die Erfassung des aktiven Fensters soweit zu modifizieren, dass es nicht erforderlich ist, das Fenster manuell anzuklicken, sondern dass diese Funktion eine neu geöffnete Datei sofort und automatisch erkennt. Weitere Empfehlungen sind eine Erweiterung der Interaktionsmöglichkeiten, sodass der Marker auch mithilfe der Maus eingeblendet werden kann. Diese Erweiterung beinhaltet auch die potentielle Idee, einige Optionen im Kontextmenü zu entfernen und stattdessen den Marker durch einen einfachen Mausklick vergrößern oder verkleinern. Damit die Nutzer den Marker dem betreffenden Fenster besser zuordnen können, empfehlen diese eine erweiterte Feedbackfunktion, welche dem Nutzer anzeigt, welcher Inhalt im Marker aktiv ist und zu welchem Fenster dieser zugeordnet ist, um vor allem bei mehreren offenen Fenstern einen Überblick zu behalten. Ebenfalls wird angedeutet eine Tutorialfunktion zu implementieren, um vor allem Erstnutzern den Einstieg zu erleichtern. Als Alternative um die Sicherheit zu verbessern, wurde eine Authentifikation per Whitelist vorgeschlagen, um den Nutzern Aufwand durch die Authentifikation per Passwort zu ersparen.

10.1.1 Sicherheitsaspekte

Um weitere Möglichkeiten zu finden die Verbindungen und Dateien zu sichern, wurden die Nutzer auch nach ihren Vorgehensweisen bezüglich der sicheren Übertragung ihrer Dateien befragt. Hierbei verwenden die Nutzer entweder private Kommunikationskanäle (wie E-Mail oder Messenger), oder physikalische Medien wie USB-Sticks und andere Datenträger. Eine weitere Möglichkeit besteht auch in der Verschlüsselung der betreffenden Dateien.

10.2 Endergebnisse und empfohlenes Vorgehen für die weitere Entwicklung

Basierend auf der Evaluation konnten folgende Features spezifiziert und das weitere Vorgehen ermittelt werden. Diese Features wurden nicht in einer nachfolgenden Interation verwendet, bieten sich aber als mögliche Punkte an, die man bei einer zukünftigen Entwicklung der Anwendung einbeziehen sollte.

Sicherheit

Die aktuellen Sicherheitsmaßnahmen werden von den Nutzern positiv aufgenommen und schaffen es einen Kompromiss zwischen Sicherheit und Aufwand herzustellen, sodass die Nutzer ihre Übertragung gesichert haben, aber keinen übermäßigen Aufwand betreiben müssen[94]. Eine alternative Vorgehensweise würde sich möglicherweise im Einrichten einer Whitelist bieten, wie sie von P3 vorgeschlagen wurde, wodurch die Möglichkeit bestehen würde, die Passwortabfrage zu eliminieren und eine genauere Kontrolle darüber auszuüben, welche Geräte auf die Anwendung zugreifen dürfen.

10.2.1 Interaktionsmethoden

Bezüglich der Interaktionsmethoden sind keine Anpassungen erforderlich, da die Nutzer sich je nach eigener Präferenz ihre eigene, naheliegende Interaktionsmethode aussuchen, es könnte sich allerdings anbieten, die bestehenden Interaktionsmethoden um weitere Möglichkeiten zu erweitern, wie eine Vergrößerung des Markers per Mausklick oder die Möglichkeit, die Anwendung komplett per Hotkey zu bewegen und zu konfigurieren.

10.2.2 Erweiterung der bestehenden Features

Basierend auf den Verbesserungsvorschlägen der Nutzer ist eine umfangreiche Erweiterung der Features empfohlen. So beinhalten die vorgeschlagenen Verbesserungen unter anderem die Möglichkeit, eine Datei ohne das vorherige Öffnen zu

[94] Da hilft es wahrscheinlich, dass die Nutzer sich auch nur einmalig Authentifizieren müssen

übertragen oder auch einen Ordner komplett oder teilweise mithilfe des Markers zu übertragen. Weiterhin bietet es sich an, für die Nutzer ein grundlegendes Feedback für das aktuell ausgewählte Fenster zur Verfügung zu stellen, damit die Nutzer Bescheid wissen, auf welches Fenster sich der aktuelle Marker bezieht. Beim permanenten Einblenden des Markers muss noch eine Lösung gefunden werden, um dieses Design auch für Probanden ansprechend zu machen, welche sich an einem permanent eingeblendeten Marker stören.

11 Technische Evaluation

Im Anschluss an die Evaluation durch die Nutzer wurde eine technische Evaluation durchgeführt. Diese Evaluation hatte hierbei das Ziel, die scanbarkeit von Markern im Größenbereich 5x5 bis 100x100 Pixel in Abhängigkeit von verschiedenen Bildschirmen, Abständen und Mobilgeräten zu prüfen.

11.1 Versuchsaufbau

Bei den verwendeten Bildschirmen handelt es sich um einen *Dell UltraSharp 27 4K Monitor* (27 Zoll, 3840 x 2160 Auflösung)[95], einen *Dell P2211H* (21,5 Zoll, 1920 x 1080 Auflösung)[96] und einen *ROG Strix XG248Q* (23.8 Zoll, 1920x1080 Auflösung)[97]. Als Mobilgeräte wurden hierbei ein *Huawei Mate 20 Lite*[98], ein *Xiaomi Redmi A9*[99] und ein *Yotopt*[100]-Tablet mit Modellnummer Y103 EEA verwendet. Für die Evaluation sind die Kameras der Mobilgeräte relevant, wobei die Kameras der Geräte jeweils 20 Megapixel(Huawei), 13 Megapixel (Xiaomi) und 5 Megapixel(Yotop) aufweisen.

11.2 Versuchsablauf

Die Mobilgeräte wurden am Anfang mit einem Stativ befestigt und 10cm vom Bildschirm entfernt platziert. Es wurde ein Marker eingeblendet, die QPixmap auf welcher der Marker angezeigt wird, wurde für den Anfang auf eine Größe von 5x5 Pixel skaliert, um mit der kleinsten sichtbaren Markergröße zu beginnen. Dieser wurde

[95] Technische Daten einsehbar unter:https://www.prad.de/technische-daten/monitor-datenblatt/dell-u2720q/

[96] Technische Daten einsehbar unterhttps://www.prad.de/technische-daten/monitor-datenblatt/dell-p2211h/

[97] Technische Daten Einsehbar unter:https://rog.asus.com/de/monitors/23-to-24-5-inches/rog-strix-xg248q-model/wtb/

[98] Technische Daten einsehbar unterhttps://www.devicespecifications.com/de/model/23d04be1

[99] Technische Daten einsehbar unterhttps://www.mi.com/de/redmi-9a/specs/

[100] Technische Daten einsehbar unter: http://www.yotopt.com/products/14.html

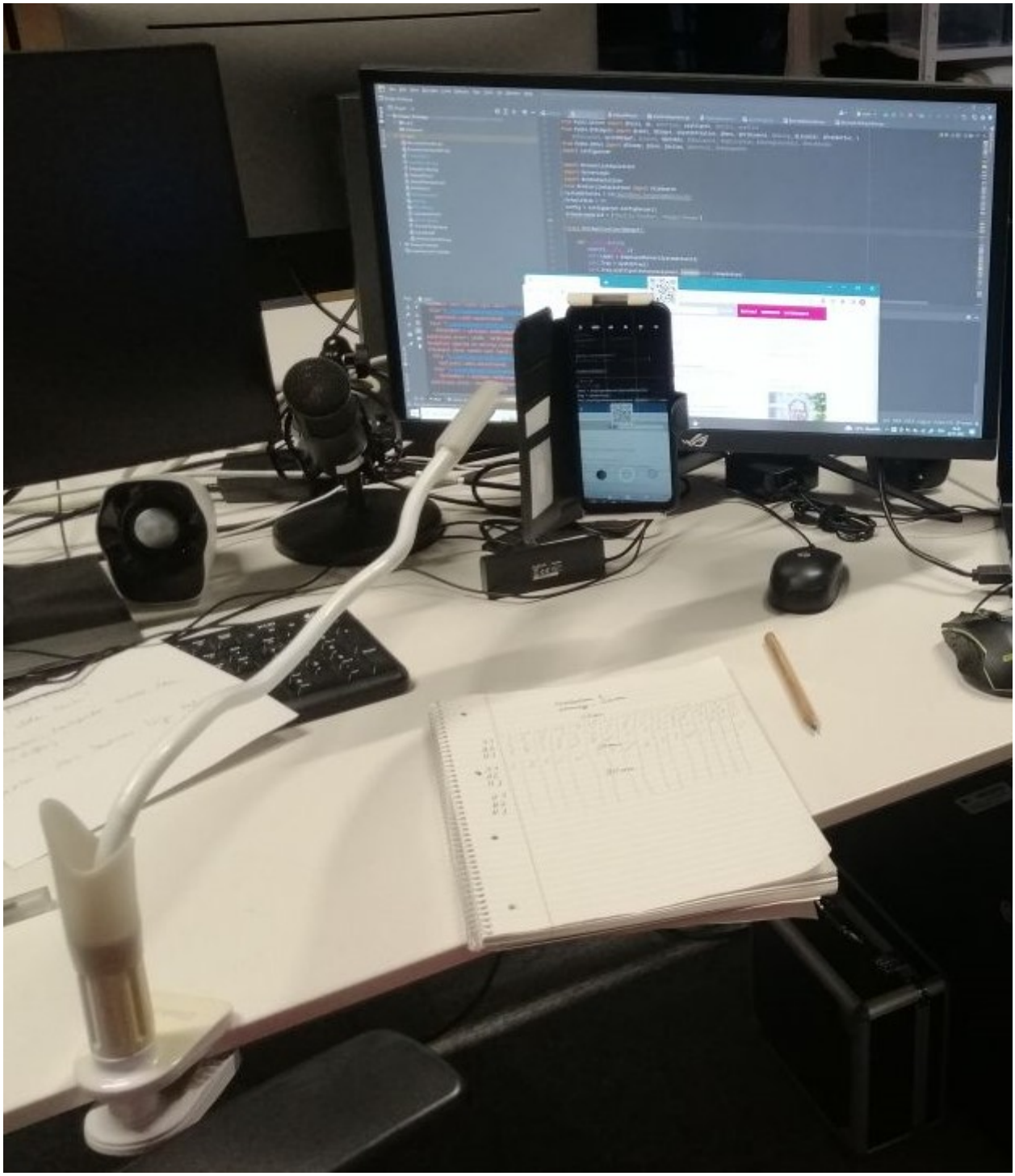

Abbildung 20: Der Versuchsaufbau mit dem Asus-Bildschirm und dem Xiaomi, zum Zeitpunkt der Aufnahme betrug die Distanz zwischen Gerät und Bildschirm 20 cm(eig.Abb)

in das Sichtfeld der Kamera bewegt, sodass dieser immer in der Mitte der Kamera-Anwendung sichtbar war. Bei den Mobilgeräten wurde der jeweils eingebaute QR-Scanner im Kameramenü aktiviert, es wurden keine externen Anwendungen verwendet oder installiert. Der Marker wurde für einen Zeitraum von maximal 15 Sekunden im Sichtfeld der Kamera belassen, um auch langsameren Geräten eine Möglichkeit zu bieten, den Marker zu erfassen. Nach Ablauf der 15 Sekunden wurde der Erfolg vermerkt (Erfassung erfolgt vs. nicht-erfolgt), ein Scan galt als erfolgt, wenn die Kamera den QR-Code erfasste und auf den Inhalt weiterleitete. Der Inhalt war dabei stets derselbe und war ein regulärer Link zur Webseite des Fachbereichs, um einen Marker mit einem "mittleren"[101] Inhalt zu simulieren (`https://www.uni-regensburg.de/sprache-literatur-kultur/medieninformatik/aktuelles/index.html`.

Nachdem der Erfolg notiert wurde, wurde schrittweise die Größe um jeweils 5x5 Pixel erhöht, bis zum Maximalwert von 100x100 Pixel, diese Prozedur wurde für jedes Gerät am jeweiligen Bildschirm wiederholt[102]. Dieser Wertebereich wurde gewählt, um die kleinste, noch einscannbare Markergröße zu ermitteln, um damit einen potentiellen Richtwert für einen permanent eingeblendeten Marker zu ermitteln, ab welchen die Erfassung weitestgehend funktioniert, ohne dass die Markergröße auf die Probanden negativ einwirkt (z.B. indem dieser Teile des Fensters verdeckt). Es wurde eine Schrittgröße von 5x5 gewählt, da bei solchen Ausmaßen eine Veränderung des Markers besser sichtbar wird (man könnte wenn man detailliert vorgehen will auch in 1x1 Schritten erhöhen, um möglichst exakt die noch kleinste Größe zu ermitteln). Nachdem für die Distanz alle Messungen erfolgten, wurde die Distanz auf 20cm erhöht und der Prozess wiederholt. So kamen insgesamt pro Bildschirm auf allen drei Distanzen 180 Messungen zustande, mit 60 Messungen pro Gerät, bei drei Bildschirmen konnten 540 Ergebnisse generiert werden.

[101] Es ging darum, einen Inhalt zu finden, welcher länger als eine URL ist, aber kürzer als ein Dokumentenname

[102] Insgesamt 20 Scans pro Gerät und Distanz

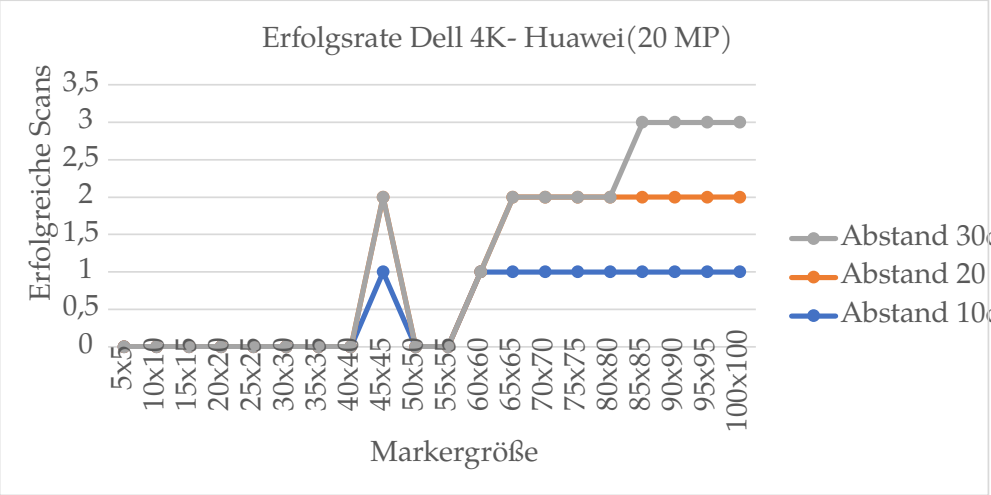

Abbildung 21: Die Erfolgsrate vom Huawei mit einer 20 Megapixel Kamera auf dem Dell 4K Monitor, die Erfolgswerte werden hierbei zusammengezählt, wenn ein Punkt bei 3 liegt bedeutet das folglich, dass die betreffende Markergröße auf allen drei Distanzen erfasst werden konnte(eig.Abb)

11.3 Ergebnisse der Evaluation

Die Erfolgsraten wurden in Tabellen für jeden Bildschirm binär zusammengetragen (0 für Misserfolg, 1 für Erfolg).

Die Messungen ergaben Erfolgsraten, welche von der Qualität der Kamera, der Distanz und der Qualität des Bildschirms abhingen. So wies die Kombination aus B1 (mit einer 4K Auflösung) und M1 (20 Megapixel) die höchste Erfolgsrate auf, wo auf eine Distanz von 10cm bis zu 50 Prozent der Marker erfasst werden konnten, einen ähnlichen Erfolg konnte man beim *Asus* erkennen. Bei allen drei Mobilgeräten und Bildschirmen wird deutlich, dass die Erfolgsrate mit einer höheren Distanz abnimmt, ein Trend, welcher auch von der Qualität der verwendeten Kamera abhängt. Der Erfolg des jeweiligen Scanvorgangs ist auch von der Größe des verwendeten Markers abhängig. Es konnte, bis auf eine Ausnahme[103], keine Markergröße unter 45x45 Pixel erfasst werden. Auffällig war, dass die Größe 45x45 erfasst werden konnte, aber nicht die sukzessiven Größen von 50x50 - 60x60 unter manchen Konstellationen. Die Erfolgsrate wird ebenfalls von der Qualität und Distanz der jeweiligen Kamera beeinflusst, Yotop konnte trotz der 5 Megapixel-Kamera immer

[103] Bei einer Größe von 30x30 Pixeln konnte bei der Kombination Xiaomi und Dell P221H und bei einer Distanz von 10cm der Marker vom Xiaomi erfasst werden

noch mindestens 25 Prozent der Marker, bei einer Distanz von 10cm erfassen, anschließende Scans aus höheren Distanz verliefen erfolglos. Eine Auffälligkeit ergab sich bei Xiaomi, mit einer 13 Megapixel Kamera, welche beim *Asus* keinen Marker aus der näheren Distanz erfassen konnte.

11.4 Schlussfolgerungen aus der Evaluation

Im Anschluss an eine Evaluation durch die Nutzer wurde auch eine technische Evaluation durchgeführt, welche das Ziel hatte, die Scanbarkeit eines Markers anhand verschiedener Mobilgeräte, Bildschirme und Abstände zu prüfen. Aus den Ergebnissen der Evaluation wird ersichtlich, dass für eine hardwareunabhängige Nutzung des Prototypen, (automatische) Mechanismen zur Anpassung der Markergröße notwendig sind, um die Anwendung für eine möglichst breite Auswahl an Geräten und Distanzen nutzbar machen zu können. Es wird ersichtlich, dass die Verwendung einer kleinen, unauffälligen Markergröße von dem jeweiligen Bildschirm und Mobilgerät abhängig ist, wodurch es schwer ist dem Marker eine für alle Systeme gültige Mindestgröße zuzuordnen, da die Einscanbarkeit dieser von der Umgebung abhängt. Für die Anwendung und deren zukünftige Iterationen bedeutet dies folglich, dass um komplett automatisch agieren zu können, die Anwendung entsprechend die Bildschirmauflösung, Bildschirmdiagonale und auch die Kameraqualität als Parameter in die automatische Berechnung seiner Größe einbeziehen muss, wobei sich hier die Verwendung allzu großer Markergrößen negativ auf die User-Experience auswirken könnte. Alternativ muss dann die Anwendung gezwungenermaßen von den Nutzern selbst eingestellt und konfiguriert werden.

12 Diskussion und Ansätze für eine weitere Entwicklung

In einem zweimaligen Durchlauf des User Centered Design - Prozesses konnte eine Anwendung entwickelt werden, welche es den Nutzern erlaubt durch das einscannen eines, in der Desktopumgebung einblendeten Markers die Datei oder URL im aktiven Fenster mithilfe eines Webservers zu übertragen. Die Nutzer können hierbei mit dem eingeblendeten Marker auch per Maus oder Keyboard interagieren und besitzen die Möglichkeit den Code nach dem jeweils eigenen Bedarf konfigurieren zu können. Die Nuzer können hierbei die Standardmäßige Größe des eingeblendeten Markers konfigurieren, sowie das verwendete Passwort und die Hotkey-Kombinationen. Die Übertragung der Daten erfolgt über einen Twisted-Webserver, welcher von den Nutzern die Eingabe eines Nutzernamens und Passwortes fordert und mit einer durch SSL/TLS verschlüsselten Verbindung ausgestattet ist.

Im Laufe der Evaluation im Kapitel 10 ergaben sich für die Anwendung noch diverse Performance-Probleme, welche in einer zukünftigen Iteration noch behoben werden müssen. Generell müssen noch Wege gefunden werden, mit denen beispielsweise die Erfassung der Systemevents und der Erstellung des Markers noch performanter gestaltet werden kann. Im Laufe der Nutzerstudien ergab sich weiterhin ein wiederkehrender Gegensatz zwischen einer permanenten Einblendung des Markers und einer Einblendung nach Bedarf. So wurde von einem Teil der Probanden die permanente Einblendung eines Markers unabhängig der Größe als ein Störfaktor eingestuft, während sich ein anderer Teil der Probanden an so einer Einblendung nicht störten und diese sogar als praktisch empfanden, aufgrund des geringen Aufwandes der mit der Bedienung verbunden ist. Im Verlauf der technischen Evaluation im Kapitel11 ergab sich auch, dass um einen Marker erfolgreich einzuscannen der Bildschirm in seiner Größe und Auflösung, die Kameraqualität des Mobilgerätes (Megapixel) und die Distanz vom Bildschirm selber beachtet werden

müssen. Bei einem permanent eingeblendeten Marker muss hier ein Kompromiss getroffen werden zwischen der Größe des permanent eingeblendeten Markers und der User Experience, da ein zu großer (oder zu kleiner) Marker für die Nutzer ein negatives Element darstellen kann. Mögliche Ansätze für die (reibungslosere) Intergration eines permanenten Markers können sowohl eine veränderung des Designs sein, als auch ein potentieller Ansatz, wo der Marker für den Nutzer nicht sichtbar in die Bildschirmumgebung implementiert wird, aber immer noch durch die Handykamera erfasst werden kann.

Die bisherigen Interaktionsmöglichkeiten sind insoweit ausreichend, es würde sich aber noch anbieten, eine Möglichkeit zu finden die Anwendung jeweils durch jede Interaktionsmöglichkeit komplett steuerbar und konfigurierbar zu machen (Also die alleinige Steuerung und Konfiguration sowohl mit Maus als auch mit Keyboard erlauben). Von den Features her gibt es noch Möglichkeiten zur Erweiterung, eine weitere Möglichkeit wäre, Ordner ganz oder teilweise zu übertragen, sowie die Möglichkeit die Datei auch ohne vorheriges Öffnen übertragen zu können. Um die Sicherheit für die Nutzer potentiell zu erhöhen kann es sich anbieten, das bisherige System durch eine Whitelist zu ersetzen, wo anhand der MAC-Adresse der betreffenden Geräte gefiltert wird.

Für eine weitere Entwicklung ist ein weiterer Durchlauf des *UCD*-Zyklus empfohlen, dieser Zyklus sollte hierbei den Fokus auf einen längeren Zeitraum legen (wie z.B einen Monat),um auch die Beobachtung anstellen zu können, ob solch eine Art von Anwendung das Potenzial besitzt, sich als eine Alternative zu gängigen Übertragungsmethoden zu etablieren.

Literaturverzeichnis

Abe, S., Hiraki, T., Fukushima, S. & Naemura, T. (2018). Screen-camera communication via matrix barcode utilizing imperceptible color vibration. In *The 31st annual acm symposium on user interface software and technology adjunct proceedings* (S. 166–168).

Alapetite, A. (2010). Dynamic 2d-barcodes for multi-device web session migration including mobile phones. *Personal and Ubiquitous Computing*, *14* (1), 45–52.

Alsaiari, A., Johnson, A. & Nishimoto, A. (2019). Polyvis: Cross-device framework for collaborative visual data analysis. In *2019 ieee international conference on systems, man and cybernetics (smc)* (S. 2870–2876).

Badam, S. K. & Elmqvist, N. (2014). Polychrome: A cross-device framework for collaborative web visualization. In *Proceedings of the ninth acm international conference on interactive tabletops and surfaces* (S. 109–118).

Badam, S. K. & Elmqvist, N. (2019). Visfer: Camera-based visual data transfer for cross-device visualization. *Information Visualization*, *18* (1), 68–93.

Blandford, A., Furniss, D. & Makri, S. (2016). Qualitative hci research: Going behind the scenes. *Synthesis lectures on human-centered informatics*, *9* (1), 1–115.

Boring, S., Altendorfer, M., Broll, G., Hilliges, O. & Butz, A. (2007). Shoot & copy: phonecam-based information transfer from public displays onto mobile phones. In *Proceedings of the 4th international conference on mobile technology, applications, and systems and the 1st international symposium on computer human interaction in mobile technology* (S. 24–31).

Boubezari, R., Le Minh, H., Ghassemlooy, Z. & Bouridane, A. (2016). Smartphone camera based visible light communication. *Journal of Lightwave Technology*, *34* (17), 4121–4127.

Braun, V. & Clarke, V. (2006). Using thematic analysis in psychology. *Qualitative research in psychology*, *3* (2), 77–101.

Bröhl, C., Rasche, P., Jablonski, J., Theis, S., Wille, M. & Mertens, A. (2018). Desktop pc, tablet pc, or smartphone? an analysis of use preferences in daily activities for different technology generations of a worldwide sample. In *International conference on human aspects of it for the aged population* (S. 3–20).

Brudy, F., Holz, C., Rädle, R., Wu, C.-J., Houben, S., Klokmose, C. N. & Marquardt, N. (2019). Cross-device taxonomy: Survey, opportunities and challenges of interactions spanning across multiple devices. In *Proceedings of the 2019 chi conference on human factors in computing systems* (S. 1–28).

Bufalino, J., Freire, M. L. M., Kannala, J. & Di Francesco, M. (2020). Mamba: Adaptive and bi-directional data transfer for reliable camera-display communication. In *2020 ieee 21st international symposium on ä world of wireless, mobile and multimedia networks"(wowmom)* (S. 307-316). doi: 10.1109/WoWMoM49955.2020.00059

Chang, T.-H. & Li, Y. (2011). Deep shot: a framework for migrating tasks across devices using mobile phone cameras. In *Proceedings of the sigchi conference on human factors in computing systems* (S. 2163–2172).

Collomosse, J. P. & Kindberg, T. (2008). Screen codes: visual hyperlinks for displays. In *Proceedings of the 9th workshop on mobile computing systems and applications* (S. 86–90).

Cui, H., Bian, H., Zhang, W. & Yu, N. (2019). Unseencode: Invisible on-screen barcode with image-based extraction. In *Ieee infocom 2019-ieee conference on computer communications* (S. 1315–1323).

Dearman, D. & Pierce, J. S. (2008). It's on my other computer! computing with multiple devices. In *Proceedings of the sigchi conference on human factors in computing systems* (S. 767–776).

Fath, T., Schubert, F. & Haas, H. (2014). Wireless data transmission using visual codes. *Photonics Research*, *2* (5), 150–160.

Freire, M. L. M. & Di Francesco, M. (2016). Reliable and bidirectional camera-display communications with smartphones. In *2016 ieee 17th international symposium on a world of wireless, mobile and multimedia networks (wowmom)* (S. 1-7). doi: 10.1109/WoWMoM.2016.7523499

Gao, Z., Zhai, G. & Hu, C. (2015). The invisible qr code. In *Proceedings of the 23rd acm international conference on multimedia* (S. 1047–1050).

Hagiwara, T., Takashima, K., Fjeld, M. & Kitamura, Y. (2019). Camcutter: Impromptu vision-based cross-device application sharing. *Interacting with Computers*, *31* (6), 539–554.

Hao, T., Zhou, R. & Xing, G. (2012). Cobra: Color barcode streaming for smartphone systems. In *Proceedings of the 10th international conference on mobile systems, applications, and services* (S. 85–98).

Hocaoğlu, K., Adar, A., Arıkök, Y. A. & Rodoplu, V. (2019). Design of a low-cost visible light communication (vlc) system for music and video streaming. In *2019 innovations in intelligent systems and applications conference (asyu)* (S. 1–6).

Hu, W., Lian, S., Song, X. & Li, T. (2013). Mobile camera based cross-screen interaction by object matching and tracking. *IEEE Transactions on Consumer Electronics*, *59* (3), 452-459. doi: 10.1109/TCE.2013.6626224

ISO. (2020). *Ergonomie der mensch-system-interaktion - teil 210: Menschzentrierte gestaltung interaktiver systeme (iso 9241-210:2019); deutsche fassung en iso 9241-210:2019*. Beuth Verlag, Berlin.

Kajan, R., Szentandrási, I., Herout, A. & Zachariáš, M. (2013). On-screen marker fields for reliable screen-to-screen task migration. In *International conference on human factors in computing and informatics* (S. 692–710).

Kajan, R., Szentandrási, I., Herout, P. & Pavelková, A. (2014). Reliable and unobtrusive inter-device collaboration by continuous interaction.

Kays, R. (2015). Modulation concepts for visible light communication using video displays. In *2015 ieee 5th international conference on consumer electronics - berlin (icce-berlin)* (S. 388-392). doi: 10.1109/ICCE-Berlin.2015.7391288

Khan, L. U. (2017). Visible light communication: Applications, architecture, standardization and research challenges. *Digital Communications and Networks*, *3* (2), 78–88.

Kuraki, K., Kato, K., Nakagata, S. & Tanaka, R. (2016). File transfer system by a hidden id signaling using camera in smart devices: Easy file sharing between computers and smart devices. In *2016 ieee 12th international colloquium on signal processing its applications (cspa)* (S. 130-134). doi: 10.1109/CSPA.2016.7515818

Lazar, J., Feng, J. H. & Hochheiser, H. (2017). *Research methods in human-computer interaction*. Morgan Kaufmann.

Lee, C.-M., Chen, P.-H. & Chen, Z.-A. (2017). Visible light communication system with arq applied to smart mobile device. In *2017 international conference on applied system innovation (icasi)* (S. 280–283).

Lee, J. R., Ruan, S.-J. & Lin, C. H. (2016). Voicecode: A 2d barcode system for digital audio encoding. In *2016 ieee 5th global conference on consumer electronics* (S. 1–2).

Li, T., An, C., Xiao, X., Campbell, A. T. & Zhou, X. (2015). Real-time screen-camera communication behind any scene. In *Proceedings of the 13th annual international conference on mobile systems, applications, and services* (S. 197–211).

Liu, W., Wang, B., Li, Y. & Wu, M. (2020). Screen-camera communication system based on dynamic qr code. In *Iop conference series: Materials science and engineering* (Bd. 790, S. 012012).

Liu, X., Doermann, D. & Li, H. (2008). Vcode—pervasive data transfer using video barcode. *IEEE Transactions on Multimedia*, *10* (3), 361–371.

Matheus, L. E. M., Vieira, A. B., Vieira, L. F., Vieira, M. A. & Gnawali, O. (2019). Visible light communication: concepts, applications and challenges. *IEEE Communications Surveys & Tutorials*, *21* (4), 3204–3237.

Mondal, S. C., Prabhu, S., Pandey, V. K. & Kumar, A. (2018). Bidirectional communication system with screen based transmission and camera based reception of qr encoded information. In *2018 4th international conference for convergence in technology (i2ct)* (S. 1–5).

Norman, D. (2013). *The design of everyday things: Revised and expanded edition*. Basic books.

Oh, S., Yoo, H., Jeong, D. R., Bui, D. H. & Shin, I. (2017). Mobile plus: Multi-device mobile platform for cross-device functionality sharing. In *Proceedings of the 15th annual international conference on mobile systems, applications, and services* (S. 332–344).

Paul, S. R. & Sohag, M. H. A. (2016). A practical approach for transferring file within computer networks using visible light communication. In *2016 5th international conference on informatics, electronics and vision (iciev)* (S. 268-271). doi: 10.1109/ICIEV.2016.7760008

Pohlmann, C. (2010). Visible light communication. In *Seminar kommunikationsstandards in der medizintechnik* (S. 1–14).

Pyla, P. S., Tungare, M., Holman, J. & Pérez-Quiñones, M. A. (2009). Continuous user interfaces for seamless task migration. In *International conference on human-computer interaction* (S. 77–85).

Pyla, P. S., Tungare, M. & Pérez-Quinones, M. (2006). Multiple user interfaces: Why consistency is not everything, and seamless task migration is key. In *Proceedings of the chi 2006 workshop on the many faces of consistency in cross-platform design.*

Rehman, S. U., Ullah, S., Chong, P. H. J., Yongchareon, S. & Komosny, D. (2019). Visible light communication: a system perspective—overview and challenges. *Sensors*, *19* (5), 1153.

Santosa, S. & Wigdor, D. (2013). A field study of multi-device workflows in distributed workspaces. In *Proceedings of the 2013 acm international joint conference on pervasive and ubiquitous computing* (S. 63–72).

Sauro, J. & Lewis, J. R. (2012). *Quantifying the user experience: Practical statistics for user research*. Elsevier.

Scharf, F., Wolters, C., Cassens, J. & Herczeg, M. (2013). Cross-device interaction. In *Ambient 2013, the third international conference on ambient computing, applications, services and technologies* (S. 35–41).

Schmid, A., Fischer, T., Weichart, A., Hartmann, A. & Wimmer, R. (2021). Screenshotmatcher: Taking smartphone photos to capture screenshots. In *Proceedings of mensch und computer 2021* (S. 44–48). New York, NY, USA: Association for Computing Machinery. Zugriff auf `https://doi.org/10.1145/3473856.3474014` doi: 10.1145/3473856.3474014

Schmidt, D., Seifert, J., Rukzio, E. & Gellersen, H. (2012). A cross-device interaction style for mobiles and surfaces. In *Proceedings of the designing interactive systems conference* (S. 318–327).

Shi, S., Chen, L., Hu, W. & Gruteser, M. (2015). Reading between lines: High-rate, non-intrusive visual codes within regular videos via implicitcode. In *Proceedings of the 2015 acm international joint conference on pervasive and ubiquitous computing* (S. 157–168).

Shneiderman, B., Plaisant, C., Cohen, M., Jacobs, S. & Elmqvist, N. (2018). *Designing the user interface: Strategies for effective human-computer interaction* (Sixth edition, global edition Aufl.). Boston and Columbus and Indianapolis and New York and San Francisco and Hoboken and Amsterdam and Cape Town and Dubai and London and Madrid and Milan and Munich and Paris and Montréal and Toronto and Delhi and Mexico City and Sao Paulo and Sydney and Hong Kong and Seoul and Singapore and Taipei and Tokyo: Pearson.

Stafford, M., Rogers, A., Wu, S., Carver, C., Artan, N. S. & Dong, Z. (2017). Tetris: Smartphone-to-smartphone screen-based visible light communication. In *2017 ieee 14th international conference on mobile ad hoc and sensor systems (mass)* (S. 570–574).

Strohmeier, P. (2015). Displaypointers: seamless cross-device interactions. In *Proceedings of the 12th international conference on advances in computer entertainment technology* (S. 1–8).

TIDWELL, J. (2011). *Designing interfaces. 2nd.* Sebastopol, CA: O'Reilly.

Toh, S. R., Goh, W. & Yeo, C. K. (2016). Data exchange via multiplexed color qr codes on mobile devices. In *2016 wireless telecommunications symposium (wts)* (S. 1–6).

Turner, S. (2014). Transport layer security. *IEEE Internet Computing*, *18* (6), 60–63.

Wang, A., Li, Z., Peng, C., Shen, G., Fang, G. & Zeng, B. (2015). Inframe++ achieve simultaneous screen-human viewing and hidden screen-camera communication. In *Proceedings of the 13th annual international conference on mobile systems, applications, and services* (S. 181–195).

Wang, A., Ma, S., Hu, C., Huai, J., Peng, C. & Shen, G. (2014). a robust barcode system for data transmissions over screen-camera links. In *Proceedings of the 20th annual international conference on mobile computing and networking* (S. 321–324).

Xie, M., Hao, L., Yoshigoe, K. & Bian, J. (2013). Camtalk: A bidirectional light communications framework for secure communications on smartphones. In *International conference on security and privacy in communication systems* (S. 35–52).

Yamsang, P., Pongyart, W. & Vanichchanunt, P. (2017). Multilevel grey-scale visual coding for data transmission from lcd screens to android smartphone cameras. In *2017 8th international conference of information and communication technology for embedded systems (ic-ictes)* (S. 1–4).

Zeng, Y., Chen, Y., Feng, S., Zhou, K., Sun, X., Mao, T., … others (2014). Visible light communication system for mobile device. In *2014 sixth international conference on ubiquitous and future networks (icufn)* (S. 26–28).

Zhang, B., Ren, K., Xing, G., Fu, X. & Wang, C. (2015). Sbvlc: Secure barcode-based visible light communication for smartphones. *IEEE Transactions on Mobile Computing*, *15* (2), 432–446.

Zhang, K., Wu, C., Yang, C., Zhao, Y., Huang, K., Peng, C., … Yang, Z. (2018). Chromacode: A fully imperceptible screen-camera communication system. In *Proceedings of the 24th annual international conference on mobile computing and networking* (S. 575–590).

Erklärung zur Urheberschaft

Ich habe die Arbeit selbständig verfasst, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt, sowie alle Zitate und Übernahmen von fremden Aussagen kenntlich gemacht.

Die Arbeit wurde bisher keiner an keiner anderen Hochschule zur Erlangung eines akademischen Grades eingereicht.

Die vorgelegten Druckexemplare und die vorgelegte digitale Version sind identisch.

Regensburg, 1.12.2022 _____

Unterschrift

Erklärung zur Lizenzierung und Publikation dieser Arbeit

Name: Alexander Kugler

Titel der Arbeit: *Cross-Device Interaction mit markerbasiertem Window Manager*

Hiermit gestatte ich die Verwendung der schriftlichen Ausarbeitung zeitlich unbegrenzt und nicht-exklusiv unter folgenden Bedingungen:

- ☐ Nur zur Bewertung dieser Arbeit
- ☐ Nur innerhalb des Lehrstuhls im Rahmen von Forschung und Lehre
- ☒ Unter einer Creative-Commons-Lizenz mit den folgenden Einschränkungen:
  - ☒ BY – Namensnennung des Autors
  - ☐ NC – Nichtkommerziell
  - ☐ SA – Share-Alike, d.h. alle Änderungen müssen unter die gleiche Lizenz gestellt werden.

(An Zitaten und Abbildungen aus fremden Quellen werden keine weiteren Rechte eingeräumt.)

Außerdem gestatte ich die Verwendung des im Rahmen dieser Arbeit erstellten Quellcodes unter folgender Lizenz:

- ☐ Nur zur Bewertung dieser Arbeit
- ☐ Nur innerhalb des Lehrstuhls im Rahmen von Forschung und Lehre
- ☐ Unter der CC-0-Lizenz (= beliebige Nutzung)
- ☒ Unter der MIT-Lizenz (= Namensnennung)
- ☐ Unter der GPLv3-Lizenz (oder neuere Versionen)

(An explizit mit einer anderen Lizenz gekennzeichneten Bibliotheken und Daten werden keine weiteren Rechte eingeräumt.)

Ich willige ein, dass der Lehrstuhl für Medieninformatik diese Arbeit – falls sie besonders gut ausfällt - auf dem Publikationsserver der Universität Regensburg veröffentlichen lässt.

Ich übertrage deshalb der Universität Regensburg das Recht, die Arbeit elektronisch zu speichern und in Datennetzen öffentlich zugänglich zu machen. Ich übertrage der Universität Regensburg ferner das Recht zur Konvertierung zum Zwecke der Langzeitarchivierung unter Beachtung der Bewahrung des Inhalts (die Originalarchivierung bleibt erhalten).
Ich erkläre außerdem, dass von mir die urheber- und lizenzrechtliche Seite (Copyright) geklärt wurde und Rechte Dritter der Publikation nicht entgegenstehen.

- ☒ Ja, für die komplette Arbeit inklusive Anhang
- ☐ Ja, für eine um vertrauliche Informationen gekürzte Variante (auf dem Datenträger beigefügt)
- ☐ Nein
- ☐ Sperrvermerk bis (Datum):

Regensburg, 1.12.2022

Unterschrift

Inhalt des beigefügten Datenträgers

`/1_Ausarbeitung`	Bachelorarbeit im PDF-Format
`/2_Links zu den Prototypen`	Links für Repositories
`/3_Studien/Design`	Interviewskript für die Anforderungserhebung sowie die Nutzerstudien
`/4_Studien/Rohdaten`	Sinngemäße Transkripte der Anforderungserhebung, Studien und deren Codierungen, sowie die Rohdaten
`/5_Studien/Auswertung`	Auswertungen der Anforderungserhebung und der Studien
`/6_Einverständnisserklärungen`	Einverständnisserklärungen der per Aufruf rekrutierten Nutzer
`/7_Abbildungen`	Alle selbst erstellten und übernommenen Abbildungen
`/8_Quellcode`	Quellcode der Prototypen