RUPRECHT-KARLS-
UNIVERSITÄT
HEIDELBERG

HOCHSCHULE HEILBRONN
TECHNIK WIRTSCHAFT INFORMATIK

BACHELOR/MASTER

ARBEIT

Steht noch nicht ganz fest

Autor	Lukas Schmidt
Studiengang	Medizinische Informatik Universtität Heidelberg / Hochschule Heilbronn
Matrikelnummer	182706
Abgabe	1. März 2016
Referent	Prof. Dr.-Ing. Gerrit Meixner
Korreferent	Prof. Dr. Martin Haag

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Glossar

Git Werkzeug. 11

SDK A markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable. It is defined in the XML 1.0 Specification produced by the W3C, and several other related specifications, all free open standards. 16

1. Einleitung

1.1. Motivation

Durch steigenden Anzahl an Medikamenten, die Patienten über den Tag nehmen müssen, kann dies zu einer großen Mentalen Aufgabe für den Patienten werden. Durch Hilfsmittel wie Medikationsplänen oder nach Zeit vorsortierten Medikamente kann die Planung der Einnahme erleichtert werden.

Zur Erleichterung der Patienten soll der Medikationsplan nun per Smartwatch einsehbar gemacht werden und somit dem Patient eine Interaktion mit dem Plan ermöglichen. Daraus bieten sich auch Vorteile für den behandelten Arzt, der Einblick in die Einnahmegewohnheiten seines Patienten bekommt. Initiale Ideen stammen auf dem PITA-Praktikum an der HS-Heilbronn. Hier wird auch in enger Zusammenarbeit mit dieser Arbeit ein Komponente für Ärzte entwickelt, die es ihnen ermöglicht die Medikationen der Patienten zu plegen, zu überwachen und auszuwerten.

1.2. Zielsezung

Die Ziele der Arbeit sind wie folged.

Entwickeln eines interaktives Prototypen, welcher auf der Apple Watch ausgeführt werden kann. Diese Prototyp soll dann mit gegeigten Personen, die zur passenden Zielgruppe gehören, evaluiert werden. Die Ergebnisse der Evaluierung sollen im Rahmen der technischen Möglichkeiten umgesetzt werden. Zum Schluss soll noch die Machbarkeit überprüft werden, die Anwendung an das Backend-System vom PITA-Praktuikum anzuschließen Anbinden des Prototypen an das Backend-System vom PITA-Praktikum

1.3. Aufbau der Arbeit

In Kapitel 2 wird auf das Forschungsumfeld der Arbeit Bezug genommen. Weiter werden technologische Grundlagen beschrieben, wie Apple‘s Swift Programmiersprache beschrieben. Des Weiteren werden Grundlagen in der Evaluierung der Gebrauchstauglichkeit und von Wearables beschrieben Im drittel Kapitel werden Anforderung beschrieben,

die teilweise aus vorherigem PITA-Praktikum stammen. Weiter werden Anforderungen geschildert, die sich aus den Möglichkeiten der Apple Watch ergeben. Abschnitt 4 betrachtet die Kernpunkte der Implementierung. Weiter wird hier der Prototyp beschrieben In Kapitel 5 wird die Planung und Durchführung der Evaluierung beschrieben. Die Ergebnisse der Evaluierung an der Zielgruppe sind hier zu finden. In der Diskussion ,in Kapitel 6, wird aufgezeigt, welche Probleme während der Arbeit entstanden sind. Es wird gezeigt wie der Prototyp im Gesamtbild einzuordnen ist und es wird ein Ausblick gegeben, welche Ziele mit dem System weiter verfolgt werden können

2. Stand der Wissenschaft

2.1. Einleitende Worte

Während das Forschungsfeld der ”Medication Adherence”, also as Einhalten des Medikamentenplans, schon jahrzehntelang erforscht wurde, ist diese Arbeit ist Teil eines noch sehr jungen Forschungsgebietes. Smartwatches existieren noch nicht lange im Consumer Bereich. Im folgenden ist eine Zusammenfassungen der relevanten Forschungen zu finden. Apple‘s Programmiersprache Swift, sowie die Technologie der Wearables wird beschrieben.

2.2. Arbeiten im Forschungsumfeld

Leider gibt es kaum Arbeiten, welche sich mit dem Thema Smartwatch und ”Medication Adherence”beschäftigten. Dies ist auf das noch junge Forschungsfeld zurückzuführen. Sailer Arbeit [15] bietet eine Einstieg für diese Arbeit. Hier wurde auch die Thematik des PITA Praktikum abgeleitet, deren Erkenntnisse hier fortgeführt werden.

Weiter gibt es sehr spannende Forschungen im Bereich der Smartwatchanwednung, die in Fortschreitender Entwicklung auch im Bereich ”Medication Adherence”vorstellbar sind. Mit Ambient Assisted Living, also der technischen Unterstützung älterer oder eingeschränkter Personen im Haushalt, beschäftigt sich die Arbeit ”Non-obstructive Room-level Locating System in Home Environments Using Activity Fingerprints from Smartwatch” [11]. Wenn man nicht nur die Zeit, sondern auch den Ort oder das Zimmer für eine Medikamenteneinnahme definieren kann, könnte der Patienten das Medikament direkt einnehmen und die Gefahr des Vergessens verringert sich. Ebenso gliedert sich die Arbeit von Laput in den Bereich ein, der einen Kontextgewinn zur Folge hat [10]. Die Arbeit beschreibt das Erkennen von Gegenständen an ihrem Elektromagnetischem Feld. Der Kontext Gewinn, zu Erkennen welche Gegenstände der Träger der Uhr berührt, könnte zur Fehler bei Medikamenteneinnahme verhindern.

7 Sekunden

Orientiert man sich an Arbeiten, deren Ziel die smartphonegestütze ”Medication Adherenceßum Ziel hatten findet man unter anderem die aktuelle nationale Umfrage [9] aus den USA, bei der 1604 Smartphone Nutzer zur ihrer Nutzung von Gesundheitsanwednungen befragt wurden. Mehr dazu wird im Kapitel 3 zur Problemstellung aufgegriffen

2.3. Apple‘s Programmiersprache Swift

Swift wurde im Juni 2014 von Apple vorgestellt und genießt seit dem steigendes Interesse. Im Juni 2015 wurde Version 2.0 veröffentlicht [2]. Mit Version 2.0 wurde ebenfalls der Plan vorgestellt, Swift Open Source zu machen und somit auch anderen Plattformen die Entwicklung mit Swift zu ermöglichen [2]. Diesen Plan setzte Apple Ende 2015 in die Tat um. Swift ist nun völlig Open Source. Es können Vorschläge für neue Sprachfunktionen gemacht werden. Auch Apples Entwicklungsteam diskutiert seine Pläne für die Sprache öffentlich [8].

2.3.1. Objective-C und Swift

In den frühen 80er Jahren entwickelte Brad Cox die Sprache Objective-C [3]. Weiter führt Dalrymple aus, dass die Sprache die Vorteile einer schnellen C-Sprach mit den Vorteilen der objektorientierten Sprache SmallTalk verbinden sollte. Die Firma NextSTEP nutze Objectiv-C und als NEXTStep von Apple aufgekauft wurde, integrierte Apple Objectiv-C und ermöglichte Mac-Entwicklern die Nutzung.

Als Apple nun 2008 seine iOS Plattform öffnete und Entwickler eigene Anwendungen für das System schreiben konnten, bekam Objective-C neue Aufmerksamkeit. Viele Entwickler sahen Objective-C als ein Überbleibsel alter Zeiten und waren der Sprache negativ eingestellt. Apple stand nun unter Zugzwang um seine Plattform, mit der große wirtschaftliche Interessen Verbunden sind, für Entwickler attraktiv zu halten [19]. Da jedoch alle highlevel APIs in Objectiv-C vorhanden sind, ist es technisch nicht Möglich auf eine bekannte Sprache wie Java für iOS und Mac Entwicklung umzusteigen. Man entschied sich für eine Neuentwicklung, mit Hinblick auf neue Programmierparadigmen und sehr guter Kompatibilität zu alten Objectiv-C APIs [19].

2.3.2. Überblick der Neuerung

1. Type Inference
2. Closures - Functions as First Class Types
3. Generics
4. Optional Types

2.3.3. Type Inference

Bei Type Inference erkennt der Compiler, welcher Typ in eine Variable instanziert wurde. Es ist nicht nötig für die Vairable eine Typendefinition zu definieren [2]. Hierdurch wird

der Quellcode leichter zu lesen.

Listing 2.1: Beispiel zu Type Inference in Swift label

```
//Variable with type Definition
let medication: Medication = Medication()

//Variable with infered type
let medication = Medication()
```

2.3.4. Closures - Functions as First Class Types

Während bei strikt objektorientierten Sprachen nur Objekte und primitive Datentypen existieren, gibt es Sprachen, bei denen Funktionen als Typen existieren. Diese Funktionen können auch als Refernz in eine Variablen gespeichert werden. Folgende Code Beispiele sollen dies veranschaulichen. Wir nutzen hierfür einen Asynchrones Netzwerk-Request [2].

Listing 2.2: Java Beispiel zu First Class Objects

```
public interface NetworkRequestResponseHandler
    public handleNetworkResponse(Error error, Response response);
}

public interface NetworkRequest
    public performNetworkRequest(NetworkRequestResponseHandler
        handler);
}

public class ExampleClass implements NetworkRequestResponseHandler {
    public handleNetworkResponse(Error error, Response response) {
        //Use Response
    }

    public static main(String[] args) {
       NetworkRequestImpl().performNetworkRequest(this)
    }
}
```

In Beispiel 2.2 wird Java als Repräsentant für eine strikt objektorientierte Sprache verwendet. Hier wird ein Interface definiert, welches der Nutzer des Netzwerk-Requests implementieren muss, um den Request zu empfangen. Beim Aufrufen des Request, muss nun der Aufrufer als Referenz übergeben werden, damit bei Abschließen des Request, der Aufrufer benachrichtigt werden kann (handleNetworkResponse).

Listing 2.3: Swift Beispiel zu First Class Objects

```
protocol NetworkRequest {
    func performNetworkRequest(handleNetworkRequest:(Error, Response)
       ->())
}

class ExampleClass {
    func main() {
        //Store a function in a variable
        let handleRequest = {(error, response) in
            //Handle Response
        }
        NetworkRequestImpl.performNetworkRequest(handleRequest)
    }
}
```

Im Codebeispiel 2.3 benötigt es kein Interface für den Nutzer des Netzwerk-Requests. Es ist nun in Swift möglich eine Funktion zu definieren und diese gleichzeitig in einer lokale Variable zu speichern. Nun kann diese Funktion als Referenz zum Netzwerk-Requests übergeben werden. Die Funktion wird aufgerufen, wenn der Netzwerk-Requests beendet ist.

2.4. Generics

Generics sind schon längere Zeit Teil moderner Programmiersprachen. Sie Unterstützen der Entwickler um Fehler zu Compilezeit zu entdecken. Ein sehr gutes Beispiel Colletions (Arrays, Listen, Set, etc). Eine Collection Typ wie Array muss nicht für jeden Typ den er hält neu implementiert werden.

Listing 2.4: Swift Beispiel zu Generic Array Collection

```
class Medication: NSObject {
    //...
}
//Could only store type Medication
class CustomMedicationArray {
    //...
    func add(medication: Medication) {
        //...
    }

    func objectAtIndex(index: Int) -> Medication {
```

```
    }
}
//Could only store any type which inherits from NSObject
//loses all type information
class CustomArray {
    //...
    func add(medication: NSObject) {
        //...
    }

    func objectAtIndex(index: Int) -> NSObject {
        //...
        //return object
    }
}
//Could store any type
//Does not lose type information
class GenericArray<T> {
    //...
    func add(medication: T) {
        //...
    }

    func objectAtIndex(index: Int) -> T {
        //...
        //return object
    }
}
let genericArray = GenericArray<Medication>()
```

Im Codebeispiel 2.4 werden die Vorteile von Generics aufgezeigt. Der erste Versuch einen Array zu implementieren (CustomMedicationArray) zeigt, dass dieser Array zwar eine sehr gute Typen Deklarierung enthält. Der Compiler kann den Entwickler also maximal Unterstützen, jedoch ist diese Implementierung minimal Wiederverwendbar. Der zweite Ansatz (CustomArray) gibt als Typ-Einschränkung den globalen Supertyp an. Dies führt zu einer Maximalen Wiederverwendbarkeit, da jede Klasse von diesem globalen Supertyp erbt (funktioniert nur theoretisch, da in Swift kein Zwang besteht von einem Globalen Supertyp zu erben). Die letzte Implementierung (GenericArray) nutzt nun Generics. So wird bei der Definition der Klasse ein generische Typ "Tëingeführt. Diese Typ ist eine Art Platzhalten für einen konkreten Typ, der später vom Array gehalten wird. So muss bei der Initialisierung der generischen Klasse die Typ-Information für T mit übergeben werden (siehe letze Zeile in 2.4).

Das Oben beschriebene Beispiel dient nur zur Verdeutlichung den Konzeptes. Swift bietet

eine Reihe an Collection Types, darunter auch eine generische Array Implementierung.

2.5. Protocol Extension

Protocol Extension helfen dem Entwickler Abstraktionen, die über ein Interface (in Swift Protocol) definiert werden zu implementieren und dadurch Duplizierungen zu minimieren. So kann es sein, dass eine Methode eines Inerfaces in jeder Klasse, die das Interface implementiert gleich umgesetzt ist. Dies führt zu einer Code-Duplizierung.

Listing 2.5: Swift Beispiel zu Generic Array Collection

```
protocol List {
    typealias T
    var count: Int { get }
    func objectAtIndex(index: Int) -> T

    var last: T? { get }
    var first: T? { get }
}

extension List {
    var last: T {
        return objectAtIndex(count - 1)
    }

    var first: T {
        return objectAtIndex(0)
    }
}

class ArrayList<Element>: List {
    typealias T = Element
    var count: Int = 0

    func objectAtIndex(index: Int) -> T {
        //some Implementation
    }
}
```

Im Codebeispiel 2.5 wird ein List Interface definiert. Die Extension zu diesem Interface ermöglicht die Implmentierung von ”lastünd ”first”. Diese Implementierung teilen sich nun alle Klassen, die List implementieren.

2.5.1. Optional Types

Optional Types eröffnen neue Möglichkeiten beim Modellieren von Datenmodellen und erstellen von APIs. Es ist so möglich Argumente in einer Methode als Optional zu kennzeichnen und somit dem Nutzer der Methode zu erlauben eine null-Referenz zu übergeben. Ist kein Optional-Type gekennzeichnet, so verbietet der Compiler ein null-Referenz übergabe [2]. Im folgenden Beispiel wird eine Medikation in Java ohne Optional Types und in Swift mit Optional Types modelliert.

Listing 2.6: Java Beispiel mit fehlenden Optional Types

```
public class Medication {
    Date creationDate = new Date();
    Date executionDate;
}

Medication medication = new Medication()
medication.creationDate // null?
medication.executionDate // null?
```

Im Codebeispiel 2.6 ist zu erkennen, dass zur Compilezeit keine Aussage über den Zustand der Instanz-Variablen getroffen werden kann. Der Entwickler muss also aus dem logischen Kontext erkennen, welche Variablen eine null Referenz enthalten könnte. Dies kann zu Laufzeitfehlern führen.

Listing 2.7: Swift Beispiel mit Optional Types

```
class Medication {
    let creationDate = Date()
    var executionDate: Date?
}

let medication = Medication()

medication.creationDate // compiler promises to be not null
medication.executionDate // could be null
```

In 2.7 sind die Instanzvariablen nun mit Optional Types modeliert. Nun kann zur Compilezeit zugesichert werden, welche Variablen eine null Referenz enthalten können und welche Variablen sicher mit einem Wert belegt sind. Dies führt zu weniger Fehler während der Laufzeit.

Auch APIs können so modelliert werden. So darf ein Parameter nicht null sein, wenn er nicht als Optional definiert wurde. Dies macht eine API strikter und führt ebenfalls zu weniger Laufzeitfehlern, da Fehler schon zur Compilezeit erkannt werden.

2.6. Wearables

Übersetzt man es ins deutsche, bedeutet es ”Tragbaresöder Änziehbaresïm Sinne von einem Kleidungsstück tragen. Wenn man nun den Begriff Wearables mit Kleidung assoziiert, liegt man nicht falsch. Anstatt Computer auf dem Schreibtisch oder in der Hosentasche zu haben, trägt man sie am Körper wie Kleidungs -oder Schmuckstücke [4]. Die Grenzen zwischen Kleidung und Computer verschmelzen und es ist manchmal nicht klar, was man schon als Wearable bezeichnen kann oder auch nicht.

Wearables sind meist mit Sensoren ausgestattet, die Daten aus der Körperregion sammeln, an der sie getragen werden [18]. Diese Daten zeigen immer nur einen Teilauschnitt. Durch Tragen von mehreren Geräten, am Körper verteilt, können mehr Daten gesammelt werden. So ist es möglich einen noch genaueren Überblick über den Zustand des Körpers zu erhalten [18].

Erst mit der Miniaturisierung der Computertechnik und Sensoren, war es möglich Computer mit integriertem Akku in Größe einer Streichholzschachtel herzustellen. Während Swatch 1995/1996 eine Uhr vorstellte, die als Skipass funktionierte stellte Apple 18 Jahre später die Apple Watch vor, die über eien Mikroprozessor, WLAN und eine Vielzahl an Sensoren verfügt (mehr zur Apple Watch in 3.4)

Doch nicht nur Uhren zählen zu den Wearables. Google präsentierte mit der Google Glass eine Datenbrille, die über eine Kamera, ein Heads ”Prismatic head-mount”Display, Sprachstuerung, sowie Internetverbindung und weitere Sensoren verfügt [12]. Auch ein Ring, der mit Hilfe eines Sensoren den Puls misst existiert als Prototyp [18].

Während im Smartphone-Markt noch einen Focus auf Leistung und Funktion der Geräte legte, darf man beim Wearables-Markt des Faktor der Ästhetik nicht vergessen. Hier wird ein Bereich betreten, der starke Einflüsse von Mode aufzeigt. Anwender einer Gerätes achten also nicht mehr nur auch die Funktion, sondern auch auf Form, Farbe und Lifestyle, den das Produkt verkörpert. Apple bietet unter dem Namen ”Watch Editionëine Apple Watch aus echtem Gold an, deren Preis über 10.000 Euro beträgt und sich an den luxusmarkt richtet. Auch TAGHeuer, eine Firma die sich auf luxuriöses modische Uhren spezialisiert hat, betritt nun auch den Markt der Wearables [17].

3. Analyse und Entwurf

In diesem Kapitel Ergebnisse der Analyse und des Entwurfes erläuteret. große Teile der Anforderungsanalyse stammen aus dem PITA Praktikum.

3.1. Entwicklungsmethodik

Da das Projekt nur durch eine Person durchgeführt wurde, kann man nicht von einem definierbaren Methodik sprechen. Es handelte sich um ein iteratives Vorgehen. Funktionen, die bei der Evaluation erarbeitet wurden, sind direkt in eine neue Version der Anforderungen integriert worden. Der Quellcode wurde mit Park Distance Control/Park Pilot (PDC) versioniert verwaltet

3.2. Anforderungsanalyse

Die Anforderungen wurden von eine Gerspräch mit Monika Pobiruchin am Anfang des Pita-Praktikum erhoben. Kernaussage dieser Anforderungen, dass das System von alten Menschen genutzt werden kann, um Medikamente regelmäßiger einzunehmen. Dies beinhaltet eine geringe Auseinandersetzung mit der Technik des Systems. Die Uhr soll möglichst autonom sein und nicht zwingen an ein Smartphone gekoppelt sein.

3.3. Usecases

Von dem Gespräch mit Monika Pobiruchin wurde eine Persona für die Zielgruppe abgeleitet. Diese Persona ist im Anhang zu finden. Mit Hilfe der Persona, wurden die Kern-Usacases abgeleitet. Die folgenden Usecase 1 bis 4 wurden im Praktikum erarbeitet und wurden von dort übernommen.

Usecase 1	**Der Patient wird an ein Medikament erinnert**
Primärer Akteur	Patient

Beschreibung	Ein Erinnerungs Pop-Up erscheint auf dem Display und es wird ein Signal/eine Vibration ausgelost. Das Pop-Up zeigt ein Abbild des Medikaments, dessen Namen und die Uhrzeit, zu der es eingenommen werden soll.
Vorbedingung	Es wurde ein Medikationsplan aus der DB auf die Uhr geladen.
Ablauf	• Das Pop-Up mit der Erinnerung erscheint auf dem Smartwatch-Display. Die Erinnerung beinhaltet Informationen zur Uhrzeit, Menge und Art der Medikation • Der Patient bestätigt, dass er das Medikament genommen hat • Gerät bestätigt visuell dass der Patient das Medikament als genommen markiert hat
Alternativablauf	• Das Pop-Up mit der Erinnerung erscheint auf dem Smartwatch-Display • Der Patient wählt Option zum Verschieben der Medikation aus • Auf einem zusätzlichem Dialog kann er aus einer Auswahl eine Zeitdauer wählen, um das die Medikation verschoben wird
Alternativablauf 2	• Das Pop-Up mit der Erinnerung erscheint auf dem Smartwatch-Display • Der Patient reagiert nicht auf die Erinnerung • Die Erinnerung wird alle x Minuten wiederholt, solange der Patient nicht reagiert. • Das Fehlen einer Reaktion des Patientens innerhalb einer Zeit von x Minuten wird vermerkt
Ergebnis	Der Patient hat die Einnahme des Medikaments bestätigt und dieses auch eingenommen
Alternativergebnis 1	Der Patient hat die Erinnerung an die Medikamenteneinnahme verschoben
Alternativergebnis 2	Der Patient hat die Erinnerung an das Medikament ausgeschaltet

Usecase 2	**Der Patient wird an mehrere Medikamente erinnert**
Primärer Akteur	Patient
Beschreibung	Ein Erinnerungs Pop-Up erscheint auf dem Display und es wird ein Signal/eine Vibration ausgelöst. Das Pop-Up zeigt eine Liste der Medikamente, die eingenommen werden müssen.
Vorbedingung	Es wurde ein Medikationsplan aus der DB auf die Uhr geladen.
Ablauf	• Das Pop-Up mit der Erinnerung erscheint auf dem Smartwatch-Display. Die Erinnerung beinhaltet Informationen zur Uhrzeit, Menge und Art der Medikationen • Der Patient bestätigt, dass er die Medikamente alle genommen hat • Gerät bestätigt visuell, dass der Patient die Medikamente als genommen markiert hat
Alternativablauf	• Das Pop-Up mit der Erinnerung erscheint auf dem Smartwatch-Display • Der Patient wählt Option zum Verschieben der Medikationen aus • Auf einem zusätzlichem Dialog kann er aus einer Auswahl eine Zeitdauer wählen, um das die Medikationen verschoben werden
Alternativablauf 2	• Das Pop-Up mit der Erinnerung erscheint auf dem Smartwatch-Display • Der Patient wählt ein Medikament von der Liste aus • Patient befindet sich nun im Usecase 1

Alternativablauf 3	• Das Pop-Up mit der Erinnerung erscheint auf dem Smartwatch-Display • Der Patient reagiert nicht auf die Erinnerung • Die Erinnerung wird alle x Minuten wiederholt, solange der Patient nicht reagiert. • Das Fehlen einer Reaktion des Patientens innerhalb einer Zeit von x Minuten wird vermerkt
Ergebnis	Der Patient hat die Einnahme der MEdikamente bestätigt und dieses auch eingenommen
Alternativergebnis 1	Der Patient hat die Erinnerung an die Medikamenteneinnahme verschoben
Alternativergebnis 2	Der Patient hat die Erinnerung an das Medikament ausgeschaltet
Alternativergebnis 3	Der Patient hat bestimmte Medikamente ausgewählt und mit dem weiterführenden Usecase 1 bearbeitet

Usecase 3	**Einzelne Einnahmebestätigung zurücknehmen**
Primärer Akteur	Patient
Vorbedingung	Es wurde ein Medikationsplan aus der DB auf die Uhr geladen. Eine Medikation wurde als genommen makiert
Ablauf	• Das System zeigt das genommene Medikament an • Der Benutzer drückt auf den Button mit der Aufschrift “Rücknahme” • Das System wechselt zur Darstellung eines einzelnen Medikaments, beschrieben im Use-Case 1
Ergebnis	Die Einnahmebestätigung ist zurückgenommen. Die Erinnerung ist erneut zu bestätigen oder zu verschieben.

Usecase 4	**Mehrere Einnahmebestätigungen zurücknehmen**
Primärer Akteur	Patient
Vorbedingung	Es wurde ein Medikationsplan aus der DB auf die Uhr geladen. Mehrere Medikationen, welche zur gleichen Zeit genommen wurden, wurde als genommen makiert
Ablauf	• Das System zeigt die genommenen Medikamente an • Der Benutzer drückt auf den Button mit der Aufschrift “Rücknahme” • Das System wechselt zur Darstellung mehrere Medikamente, beschrieben im UseCase 2
Ergebnis	Die Einnahmebestätigung ist zurückgenommen. Die Erinnerung ist erneut zu bestätigen oder zu verschieben.

Diese vier Usecases sind auch Grundlage für dies Arbeit. Wie die Usecases umgesetzt sind wird in Kapitel 4 beschrieben. In Kapitel 5 finden sich überarbeitet Usecases, die Verbesserungen enthalten, welche aus der Evaluation mit der Zielgruppe hervorgehen.

3.4. Apple Watch

Eine Andorderung, welche sich aus dem Kontext dieser Arbeit entnehmen lässt, ist die Nutzung der Apple Watch als Zielplatform. Die Apple Watch wurde im September 2014 vorgestellt und staret im April 2015 mit dem Verkauf.

3.4.1. Hardware

Die Apple Watch existiert in zwei Versionen. Einen Uhr mit 38mm (272x340) und eine mit 42mm (312x390) großem Display. Die bietet einen 8GB großen internen Speicher. Mit einer Akkulaufzeit von 18h unter durchschnittlicher Nutzung, hält die Uhr einen Tag durch [13].

3.4.2. Abhängigkeit zum iPhone

Die Apple Watch wurde als Erweiterung zum iPhone Entwickelt. Und so sind auch viele integrale Funktionen nur vom iPhone aus steuerbar. Die Uhr kann ohne ein iPhone nicht durch den Setup-Prozess geleitet werden. Auch native Anwendungen (siehe 3.4.3) können nur über das iPhone installiert werden. Sind diese Schritte getan, also die Uhr Betriebsbereit und Anwendungen installiert, kann die Uhr teilweise auch Autonom agieren. So kann sie auch ohne iPhone, über WLAN, mit dem Internet kommunizieren.

3.4.3. Software

Mit erscheinen der Uhr wurde auch das Betriebsystem in Version 1 ausgeliefert und dazu das Extensible Markup Language (XML) Names WatchKit. Dies erlaubte es Entwicklern Anwendungen zu entwickeln, welche auf dem verbunden iPhone ausgeführt wurden. Diese führte zu schlechte Performance der Anwendungen und zu vollen Abhängigkeit zum iPhone.

Im Juni 2015 veröffentlichte Apple die erste Vorabversion von watchOS 2.0, welches später im September 2015 für Endnutzer bereit gestellt wurde. watchOS biete mehr Unabhängigkeit für Anwendungen, da diese direkt auf der Uhr ausgeführt werden. Für Anwendugsentwickler gibt es vier Arten Informationen auf der Uhr dazustellen. Es Handelt sich um native Anwendungen (Apps), Glances, Complications, und Actionable Notifications [6].

Native Anwendungen sind fest installiert auf der Uhr. Sie können unabhängig auf der Uhr gestartet werden. In einer native Anwendungen lassen sich komplexere Anwendungen realisieren, da der Nutzer durch viele Möglichkeiten hat der Eingaben und Interaktionen (3.4.7 zu tätigen. Installiert werden die Apps vom iPhone aus. Eine Watch-App benötigt immer eine iPhone App, die jeweils auf dem iPhone installiert ist.

Complications sind kleine Interface Elemente, die sich auf dem Zifferblatt der Uhr platzieren lassen. So kann mit einem Blick auf die Uhrzeit auch Informationen aus der App abgelesen werden.

Ein Glance ist ein Interface, auf dem die wichtigsten Informationen einer App übersichtlich dargestellt werden. Der Nutzer soll mit einem Blick die Informationen erkennen. Es ist keine Interaktion mit einem Glance möglich. Tippt der der Nutzer auf einen Glance öffnet sich die Zugehörige App. Glances sind optional zu einer App zu entwickeln.

Notifications oder auch auf deutsch Benachrichtigungen Informieren den Benutzer über Ereignisse. Diese Ereignisse können entweder zeitlich geplant sein, vom Betreten einer Ortskoordinate ausgelöst werden oder von einem Server auf das Gerät gepusht (Server sendet ein Ereignis, wie z.B. eine Nachricht) werden. Notification können Aktionen beinhalten. So kann eine Bestätigung einer Notification direkt geschehen, ohne dafür die dazugehörige Anwendung zu starten. Dies nennt man eine Actionable Notification [6].

Es gibt zwei verschiede Arten von Notification, die beide die gleiche Funktion haben, jedoch mit unterschiedlichen Informationsgehalt für die Uhr angereichert werden. Zum einen die Standard Notification. Diese Notification wird vom iPhone gespiegelt, bietet also nicht mehr Informationen gegenüber der iPhone Notification. Es ist jedoch möglich, mit einer nativen Anwendung auch eine optimale Darstellung der Notification zu entwickeln. Diese Darstellung kann detaillierte Informationen beinhalten wie Bilder, Karten oder genauer Beschreibung der Information [7].

3.4.4. Schniststellen

Bluetooth 4.0 und Wi-Fi 802.11b/g sind die Netzwerkschnistellen der Apple Watch. Dazu kommt noch ein NFC Chip, der jedoch nicht über eine API nutzbar ist und vorerst nur für Apple Pay, dem Apple eigenen Bezahldienst, vorgesehen ist [14].

3.4.5. Sensoren

Die Apple Watch besitzt einen Beschleunigungssensor und Gyroskop welche genaue Bewegungsdaten liefern. Ebenso ist ein optischer Herzschagsensor verbaut, der an der Unterseite der Uhr auf der Haut anliegt. Ein Mikrofon, welches für Spracheingabe genutzt werden kann ist auch vorhanden.

3.4.6. Taptic Engine

Tapic Engine

3.4.7. Eingabe Interfaces

Neben eines normalen Touchescreen führte Apple in der Uhr auch eine Eingabeart Names Force Touch ein. Diese Technologie erlaubt es der Uhr zu erkennen, wie fest der Nutzer auf das Display drückt. Dies ermöglicht eine Neue Art der Eingabe. Besonders bei einer kleinen Eingabefläche, wie die der Apple Watch, ist eine neue Interaktionsdimesion interessant, da es eine differenziertere Interaktion erlaubt. Leider ist eine mögliche Interaktion mit Force Touch optisch nicht zu erkennen, was eine klare Nutzerführung schwer macht.

Von Analogen Uhren ist die Krone, also das Rad an der Seite einer Uhr, an dem die Uhr eingestellt oder aufgezogen werden kann bekannt. An der Apple Watch ist die Krone auch zu finden. Apple nennt sie ”Digital Crown”, also digitale Krone. Sie befindet sich ebenfalls an der Seite der Uhr. Die Krone ist drehbar und lässt sich ebenfalls als Druckknopf nutzen. Sie dient zum scrollen von Inhalten, sowie präzisen Auswählen von Elementen

Accesability

aus einer Auswahlliste. Durch nutzen der Krone wird der Bildschirm nicht durch einen Finger verdeckt, was bei einem kleinen Display von Vorteil ist.

3.4.8. Armband

Apple biete 6 verschiede Armbänder für die Apple Watch an (Stand Nov. 2015). Zusätzlich ist es möglich, Armbänder von Drittherstellern zu erwerben.

Das in der günstigsten Version mitgelieferter Version (Sportarmband) verfügt über einen sehr komplizierten Verschlussmechanismus und ist deswegen weniger für Menschen geeignet, die über schwache sensomotorische Fähigkeiten verfügen. Es gibt auch Armbänder, die einen magnetischen Verschluss bieten, diese biete eine einfachen Handhabung, sind jedoch 3 mal so teuer, wie das Sportarmband.

An der Verbindung zwischen Armband und Uhr ist eine nicht weiter spezifizierter Wartungsport verbaut. Dieser Anschluss könnte in Zukunft Armbänder ermöglichen, welche Informationen aus dem Armband an die Uhr weiterleiten.

3.4.9. Prototyping

Zum erstellen des frühen Prototyps wurden Papier-Prototypen erstellt. Durch die Nutzung dieser Werkzeug konnte schnell und iterativ gearbeitet werden. Da as diese Arbeit nur eine Person gearbeitet hat, bietet sich diese Methode an, da sie schnell zu erlernen ist, bzw. keine Vorkenntnisse voraussetzt und schnell Ergebnisse erzielt.

4. Umsetzung

Die in Kapitel 3 aufgeführten Analyse Ergebnisse wurden für die erste Interation des Prototyps umgesetzt. Es handelt sich hierbei um ein funktionsfähigen Prototypen, der nativ auf der Apple Watch ausführbar ist. Im folgenden werden Schritte der Umsetzung genauer beschrieben. Hierbei wird genauer auf die Benutzeroberflächenerstellung, sowie auch die Verbindung zischen Uhr und iPhone eingegangen.

4.1. Benutzeroberfläche

Xcode bietet für visuelle Erstellung von Benutzeroberflächen ein eigenen integrierten Editor namens InterfaceBuilder bereit. Hiermit können graphische Elemente per DragAnd-Drop zu einer Benutzeroberfläche zusammengestellt werden 4.1. Ebenfalls per DragAnd-Drop werden diese Inerface-Elemete mit dem Quellcode verbunden4.2.

Die Auswahl an Interface-Elemenete ist sehr begrenzt und bietet kaum Möglichkeiten eigene Interface-Elemenete zu erstellen. Trotz dieser Begrenztheit lassen sich Anwendungen komplexere Anwendungen bauen.

4.2. WatchConnectivity

Wichtig für die Kommunikation zwischen Uhr und iPhone ist das WatchConnectivity Framework. Hierbei ist ist im Listing xx zu sehen, wie genau eine Verbindung aufgebaut werden kann. Wichtig ist, das diese Verbindung zum richtigen Zeitpunkt im Application-Lifecycle aufgebaut wird, da es sonst zu Datenverlusten kommen kann.

Mit der ßendMessage:"Methode können Nachrichten und Datenpakete gesendet werden, welche sich durch eine ID identifizieren lassen. Auch ist möglich eine direkte Antwort auf eine solche Nachricht über einen Replay-Handler zurück zu senden. So kann eine Art Request/Response Datentransfer realisiert werden. Zu beachten ist jedoch, dass die Kommunikation relativ langsam ist. Sie sollte nur verwendet werden um kleine Informationen zu senden.

Werden erst beim Watch-App Start benötigte Daten vom der iPhone angefordert, kann dies zu einer großen Verzögerung kommen. Daher sollten Informationen, die auf der Uhr angezeigt werden, nicht erst zum Zeitpunkt der Darstellung angefordert werden. Gibt es

Abbildung 4.1.: Interface Elemente zu Erstellen von Benutzeroberflächen

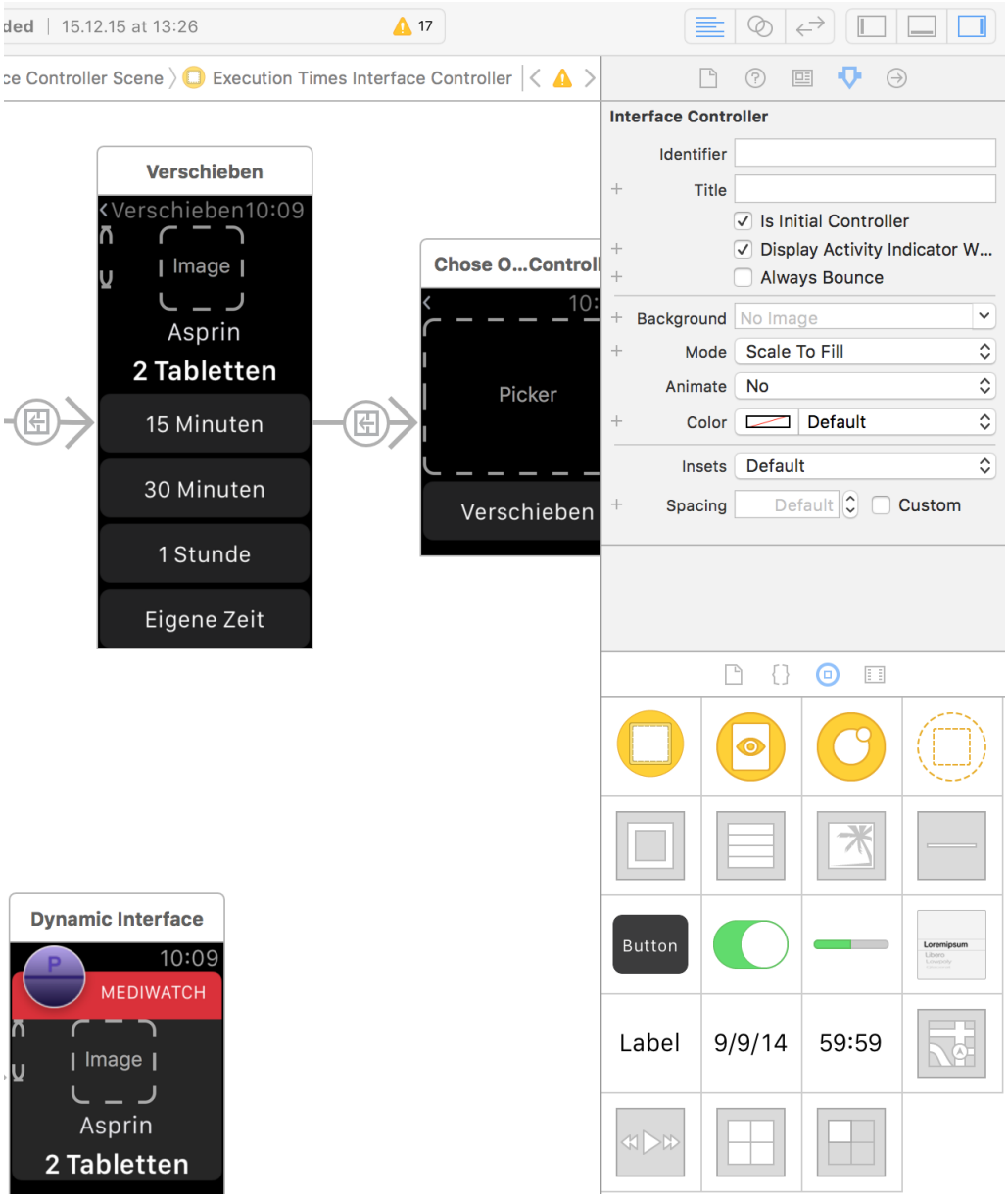
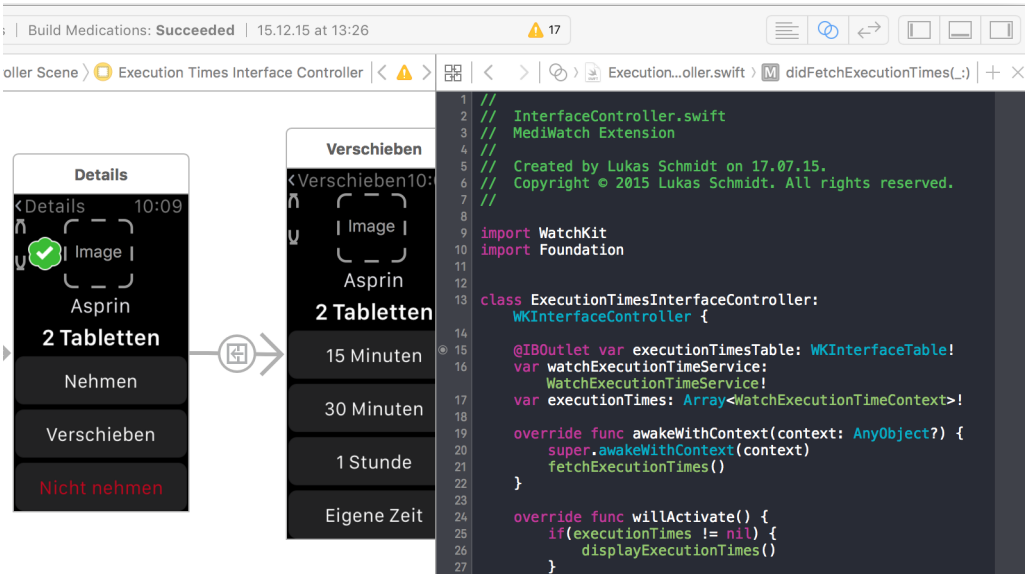


Abbildung 4.2.: Interface Elemente mit Quellcode verknüfen



eine Datenänderung auf dem iPhone, die relevante Daten für die Uhr enthält, so sollten diese mit der Methode ”transferUserInfo:än die Uhr gesendet werden. Mit dem Aufrufen dieser Methode wird die übergebene Information nicht direkt zu Uhr geschickt. Das System sendet die Daten zu einem optimalen Zeitpunkt (starke Verbindung zur Uhr, mögliche WLAN Verbindung) an die Uhr. Wenn nun die Watch-App startet, so sind die Daten bereit und können direkt dargestellt werden.

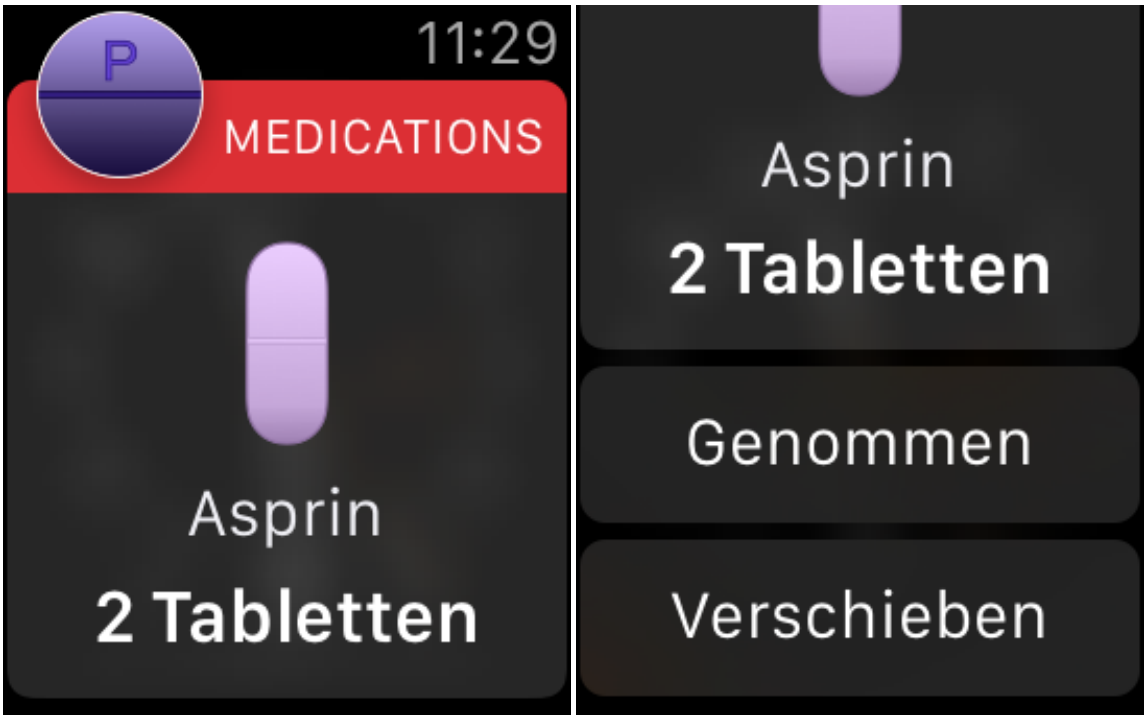
Zum übertragen von größeren Daten steht die Methode ”transferFile:ßur Verfügung. Die Methode verhält sich equivalent zu ”transferUserInfo:”, wird jedoch in der Realisierung nicht verwendet.

4.3. Notification Management

Notifications für die Medikationen werden mit der Notification API registriert werden [7]. Auch kann das Verhalten der Notifications angepasst werden. So werden vordefinierte Aktionen wie Ëinnehmenöder ”Verschiebenïn die Notifications konfiguriert.

Bei der Notification handelt sich um eine UILocalNotification [7]. Die Art von Notification benötigt keinen Remote Server , sondern wird vom verbunden iPhone verwaltet. remote S. erklären
Zum jetzigen Zeitpunkt ist es nicht möglich UILocalNotification von der Uhr zu verwalten. Dies würde die Uhr im Falle der Mediwatch-Anwendung noch unabhängiger machen. Sind die Notifications einmal auf dem iPhone registriert, werden sie zum Ausführungszeitpunkt auf der Uhr angezeigt. Dazu ist keine Verbindung zum iPhone mehr nötig.

Abbildung 4.3.: Notification für Medikament



4.4. Anwendung

Die Anwendung besteht aus 2 Komponenten auf der Apple Watch. Eine native Notification und eine auf der Uhr installierte und ausgeführte Anwendung (siehe 3.4.3).

4.4.1. Notification

Die Notification bringt die Uhr zum Zeitpunkt der geplanten Medikamenteneinnahme zum vibrieren und lässt optional einen Ton erklingen. Dies führt dazu, dass der Anwender auf die Uhr Blick und die bevorstehende Medikamenteneinnahme bemerkt. Die Notification wurde darauf hin optimiert, dass eine große graphische Repräsentation des Medikaments angezeigt wird. Dem Nutzer wird so die Wiedererkennung des Medikaments zu erleichtert (siehe 4.3).Neben dem Medikamentennamen wird auch die Dosierung des Medikaments dargestellt. Es wird dem Nutzer ermöglicht mit nur einer Aktion das Medikament zu Bestätigen oder es zu Verschieben. Die Notification ist darauf ausgelegt eine sehr schnelle Interaktion zu ermöglichen, damit der Nutzer nicht länger als 10 Sekunden mit der Uhr interagieren muss (siehe 4.3).

4.4.2. Native Anwendung

Neben der Notification gibt es noch eine native App. Diese läuft auf der Uhr und stellt eine Liste der Medikamente des aktuellen Tages dar. Die App muss aktiv vom Nutzer gestartet werden. In 4.4 wird ein Medikament aus der List ausgewählt. Nach dem man ein Medikament ausgewählt hat, bekommt man in einen Detailansicht des Medikaments zu sehen. Dort kann man nun das Medikament als genommen oder nicht genommen markieren. Ein visuelles Feedback zur Einnahme zeigt dem Nutzer seine Aktionen an. Von der Detailansicht ist es auch Möglich ein Medikament zur späteren Einnahme zu markieren. Hier hat der Anwender die Möglichkeit aus vordefinierten Zeiten auszuwählen oder eine eigene Zeit zu wählen (siehe 4.5).

Abbildung 4.4.: Native Anwendung: Interface zu wählen von eines Medikaments (oben links). Medikament als genommen bestätigen (oben rechts). Eigene Zeitdauer auswählen (unten links). Medikament ist in der Übersicht als verschoben markiert (unten rechts)

Abbildung 4.5.: Native Anwendung: Interface zu verschieben von eines Medikaments (oben links). Zeitdauer für das Verschiebenwählen (oben rechts). Medikament ist als genommen markiert (unten links). Medikament ist in der Übersicht als genommen markiert(unten rechts)

5. Evaluation, Zusammenfassung und Ausblick

5.1. Evaluation

Die Evaluierung des Prototypen soll eine quantifiziertes Ergebnis liefern, anhand dessen Schwachstellen aufgezeigt werden sollen. Diese Schwachstellen sollen mit der nächsten Interation der Software ausgebessert werden.

5.1.1. Zielgruppe

Die Zielgruppe sollten Menschen sein, deren Alltag von Medikamenteneinnahmen geprägt ist. Es sollte sich um Menschen mit wenig oder keinen Vorkenntnissen mit Touchscreen-basierten Geräten handeln. Patienten fortgeschrittenen Altern, die sich stationär im Krankenhaus aufhalten eignen sich gut, Medikamente nehmen und viel freie Zeit während des Aufenthaltes im Krankenhaus für eine Befragung haben.

5.1.2. Vorgehen

Im ersten Schritt wird der Zielgruppe die Apple Watch gezeigt und die dahinterliegende Technologie erklärt, um Neugier bei den Patienten zu wecken. Nun können die Patienten die Uhr anlegen. Im zweiten Schritt wird ein Notification, die eine Vibration auslöst gestartet. Die Patienten müssen die Vibration spüren und dann auf die Uhr schauen. Dort sollen sie den Bildschirminhalt wiedergeben. Haben sie den Bildschirminhalt wiedergegeben, so müssen sie die Benachrichtigung bestätigen, also auf den Button mit der Aufschrift ”Genommen”tippen.

In Schritt Drei müssen die Patienten durch drücken auf die digital Crown den App Bildschirm aufrufen um dort die Mediwatch App zu öffnen. Wenn die App geöffnet ist, soll ein Medikament ausgewählt werden und dieses um eine definierte Zeit verschoben werden. Ist das Medikament verschoben, so ist die Aktion beendet.

5.1.3. Auswertung

Für die Evaluation werden 2 Fragebögen genutzt. Einmal AttrakDiff [5], welcher die subjektiv Wahrnehmung der Bedienbarkeit und Aussehen des Prototypen erfragt. Es handelt sich um einen standardisierten Fragebogen und eine standardisierten Auswertungsmethode. Da dieser Frageboden keine Antworten über Funktionen des Prototyp gibt, ist es nötig einen zweiten Fragebogen zu erstellen. Dieser Fragebogen erfragt die Situation, also den Kontext in dem sich der Anwender befindet und die daraus Folgenden Ansprüche an den Prototypen. So sollen fehlende Funktionen oder Fehler in der Analyse aufdecken werde. Die Fragen sind im Anhang zu finden. Die Erfassung der Fragen wird mit Limesurvey [1] in Version 2.05 durchgeführt. Da sich Limesurvey selbst hosten lässt, bleiben die erfragten Daten auf einem sicheren Hochschulserver und werden nicht bei Drittanbietern gespeichert.

5.2. Durchführung der Evaluation

5.2.1. Beschreibung der Zielgruppe

Es haben insgesamt 11 Patienten an der Befragung teilgenommen. Davon waren 7 weiblich und 4 männlich. Alle waren im Alten zwischen 70 und 85.

5.2.2. Befragung der Patienten

Die Befragung in der Zielgruppe verlief sehr schwierig und nicht wie geplant. Der erste Schritt der Befragung klappte bei fast allen Probanden sehr gut. So erkannten alle die Notification mit der einhergehenden Vibration wurde von allen Patienten als verständlich geschildert. Das darauf folgende Bestätigen einer Notification verlief zu großen Teilen schlecht. Die Patienten konnten die Buttons zum Bestätigen nicht treffen, da ihnen die Feinmotorik fehlte. Öffnen einer App ist auf Grund der fehlenden Feinmotorik ebenfalls nicht möglich. Die App-Icons sind zu klein und werden nicht getroffen. Das öffnen der Anwendung wurde bei allen Patienten nicht erreicht.

5.2.3. Auswertung

Der Fragebogen AttrakDiff ist in dieser Zielgruppe nicht praktikabel. Eine sehr genauer Unterscheidung zwischen den Begrifflichkeit, die der AttrakDiff abfragt ist für die Zielgruppe nicht möglich. Dies fällt auf, wenn man mit den Probanden spricht. Die Probanden schweifen oft ab und es ist nicht möglich ihre Aussagen zu erfassen. Direkte Fragen, die den Alltag der Probanden betreffen werden zuverlässig beantwortet. Schwierigkeiten, die

Tabelle 5.1.: My caption

Problem	Auswirkungen
fehlender Internetanschluss	keine Aktualisierungen und keine Benachrichtigungen für den Arzt
fehlende Feinmotorik der Probanden	
Vibration zu leicht	
Nur mit Brille lesbar	
Armband schwer anlegbar	
Uhr ist unästhethisch	

bei der Medikamenten Einnahme werden nicht zugegeben. Die Befragung wurden deswegen auf eine Thinking Alound Methode [16] umgestellt. Leider sind die Ergebnisse so nicht mehr quantifizierter, jedoch sind aus des Aussagen der Patienten und den Beobachtungen gute Schlussfolgerungen möglich. Auch der der zweite Fragebogen zu den Funktionen und deren Nutzungskontext wurde nicht genutzt. Hier wurde versucht Während er Gesprächs mehr Einblick in das Leben der Probanden zu erhalten. Probanden erzählen gerne über ihr Leben und suchen eher das persönliche Gespräch. Aus diesen Gesprächen geht hervor, dass die Patienten mit großer Übereinstimmung keinen Internetanachluss besitzen. Dies schließt entfernte Aktualisierungen des Mediktionsplans und Einnahme Bestätigungen für den Arzt aus. Bedenken äußern die Weiblichen Patientinnen über die Größe und das Aussehen der Uhr. Eine Uhr muss dem Geschmack der Patientin gefallen. Hierbei spielt die Größe der Uhr, wie auch das Aussehen eine Rolle. Bei Beschreibung von anderen modischen Farben der Uhr oder der Armbänder, zeigen sie sich interessiert. Ein Großteil der Patienten sieht jedoch ein, dass sie auf Grund Ihrer Sehschwäche die große Uhr (42mm) brauchen. Die große Uhr wird in Gegenzug als zu Groß beschrieben. Die meisten Patienten haben keine Probleme die Schrift zu lesen. Meist nehmen Sie Ihre Brille zur Hilfe, die im Krankenhaus natürlich immer griffbereit ist. Nur bei einem kleine Teil kommt es vor, dass sie den Text auf dem Display auch mit Brille nicht lesen können. Das größte Problem ist jedoch die schon genannte fehlende Feinmotorik. So können die Probanden die Uhr nicht wirklich aktiv bedienen, sondern reagieren nur auf die Vibration am Handgelenk. Dies führt zu einer Hilflosigkeit gegenüber der Technik der Uhr.

5.2.4. Probleme tabellarisch

Literaturverzeichnis

[1] Limesurvey.

[2] *The Swift Programming Language*. Apple, Inc., 2014.

[3] DALRYMPLE, M. Learn objective-c on the mac. 1–20.

[4] DVORAK, J. L. Moving wearables into the mainstream.

[5] GMBH, U. I. D. Attrakdiff fragebogen.

[6] INC., A. Developing for apple watch.

[7] INC., A. Notification essentials.

[8] INC., A. Swift open source.

[9] KREBS P, D. D. Health app use among us mobile phone owners: A national survey. *MIR mHealth uHealth* (2015).

[10] LAPUT, G., YANG, C., XIAO, R., SAMPLE, A., AND HARRISON, C. Em-sense: Touch recognition of uninstrumented, electrical and electromechanical objects. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology* (New York, NY, USA, 2015), UIST '15, ACM, pp. 157–166.

[11] LEE, S., KIM, Y., AHN, D., HA, R., LEE, K., AND CHA, H. Non-obstructive room-level locating system in home environments using activity fingerprints from smartwatch. In *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing* (New York, NY, USA, 2015), UbiComp '15, ACM, pp. 939–950.

[12] MUENSTERER, O. J., LACHER, M., ZOELLER, C., BRONSTEIN, M., AND KÜBLER, J. Google glass in pediatric surgery: An exploratory study. *International Journal of Surgery 12*, 4 (2014), 281 – 289.

[13] RICHES, G. Apple watch for developers.

[14] RITCHIE, R. Apple watch specs, 2015.

[15] SAILER, F., POBIRUCHIN, M., WIESNER, M., AND MEIXNER, G. An approach to improve medication adherence by smart watches, 2015.

[16] SOMMERVILLE, I. *Software engineering*. Pearson, Boston, Munich [u.a.], 2016.

[17] TAGHEUER. Tagheuer connected.

[18] TENG, X.-F., ZHANG, Y.-T., POON, C., AND BONATO, P. Wearable medical systems for p-health. *Biomedical Engineering, IEEE Reviews in 1* (2008), 62–74.

[19] WELLS, G. The future of ios development: Evaluating the swift programming languag. Master's thesis, Claremont McKenna College, 2015.

A. Appendix

B. Dokumentation