



RUPRECHT-KARLS-  
UNIVERSITÄT  
HEIDELBERG



# BACHELOR/MASTER

## ARBEIT

**Steht noch nicht ganz fest**

Autor	Lukas Schmidt
Studiengang	Medizinische Informatik Universtität Heidelberg / Hochschule Heilbronn
Matrikelnummer	182706
Abgabe	22. Januar 2016
Referent	Prof. Dr.-Ing. Gerrit Meixner
Korreferent	Prof. Dr. Martin Haag



# Inhaltsverzeichnis

<b>Abbildungsverzeichnis</b>	<b>iii</b>
<b>Tabellenverzeichnis</b>	<b>v</b>
<b>Glossar</b>	<b>vii</b>
<b>1. Einleitung</b>	<b>1</b>
1.1. Motivation . . . . .	1
1.2. Zielsetzung . . . . .	1
1.3. Aufbau der Arbeit . . . . .	1
<b>2. Stand der Wissenschaft</b>	<b>3</b>
2.1. Einleitende Worte . . . . .	3
2.2. Arbeiten im Forschungsumfeld . . . . .	3
2.3. Apple's Programmiersprache Swift . . . . .	4
2.3.1. Objective-C und Swift . . . . .	4
2.3.2. Überblick der Neuerung . . . . .	4
2.3.3. Type Inference . . . . .	4
2.3.4. Closures - Functions as First Class Types . . . . .	5
2.3.5. Optional Types . . . . .	6
2.4. Wearables . . . . .	7
<b>3. Analyse und Entwurf</b>	<b>9</b>
3.1. Entwicklungsmethodik . . . . .	9
3.2. Anforderungsanalyse . . . . .	9
3.3. Usecases . . . . .	9
3.4. Apple Watch . . . . .	13
3.4.1. Hardware . . . . .	13
3.4.2. Abhängigkeit zum iPhone . . . . .	14
3.4.3. Software . . . . .	14
3.4.4. Schnittstellen . . . . .	15
3.4.5. Sensoren . . . . .	15
3.4.6. Eingabe Interfaces . . . . .	15
3.4.7. Armband . . . . .	16
3.4.8. Prototyping . . . . .	16
3.4.9. Evaluation . . . . .	16

<b>4. Umsetzung</b>	<b>17</b>
4.1. Benutzeroberfläche . . . . .	17
4.2. WatchConnectivity . . . . .	17
4.3. Anwendung . . . . .	19
<b>5. Zusammenfassung, Evaluation und Ausblick</b>	<b>23</b>
<b>Literaturverzeichnis</b>	<b>25</b>
<b>A. Appendix</b>	<b>27</b>
<b>B. Dokumentation</b>	<b>29</b>

# Abbildungsverzeichnis

4.1.	Interface Elemente zu Erstellen von Benutzeroberflächen . . . . .	18
4.2.	Interface Elemente mit Quellcode verknüpfen . . . . .	19
4.3.	Notification für Medikament . . . . .	20
4.4.	Interface zu wählen von eines Medikaments (oben links). Medikament als genommen bestätigen (oben rechts). Medikament ist als genommen markiert (unten links). Medikament ist in der Übersicht als genommen markiert(unten rechts) . . . . .	21



# Tabellenverzeichnis





# Glossar

**Git** Werkzeug. 9

**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. 14



# 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 pflegen, zu überwachen und auszuwerten.

## 1.2. Zielsetzung

Die Ziele der Arbeit sind wie folged.

Entwickeln eines interaktives Prototypen, welcher auf der Apple Watch ausgeführt werden kann. Diese Prototyp soll dann mit geeigneten 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-Praktikum 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 als Einhalten des Medikamentenplans, schon jahrzehntelang erforscht wurde, ist diese Arbeit Teil eines noch sehr jungen Forschungsgebietes. Smartwatches existieren noch nicht lange im Consumer Bereich. Im folgenden ist eine Zusammenfassung 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" beschäftigen. Dies ist auf das noch junge Forschungsfeld zurückzuführen. Sailer Arbeit [12] bietet einen Einstieg für diese Arbeit. Hier wurde auch die Thematik des PITA Praktikums abgeleitet, deren Erkenntnisse hier fortgeführt werden.

Weiter gibt es sehr spannende Forschungen im Bereich der Smartwatchanwendung, 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" [8]. Wenn man nicht nur die Zeit, sondern auch den Ort oder das Zimmer für eine Medikamenteneinnahme definieren kann, könnte der Patient 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 [7]. Die Arbeit beschreibt das Erkennen von Gegenständen an ihrem elektromagnetischen Feld. Der Kontextgewinn, zu Erkennen welche Gegenstände der Träger der Uhr berührt, könnte zur Fehler bei Medikamenteneinnahme verhindern.

Orientiert man sich an Arbeiten, deren Ziel die smartphonegestützte "Medication Adherence" zu sein hatten, findet man unter anderem die aktuelle nationale Umfrage [6] aus den USA, bei der 1604 Smartphone-Nutzer zur ihrer Nutzung von Gesundheitsanwendungen 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 [1]. 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 [1].

### 2.3.1. Objective-C und Swift

In den frühen 80er Jahren entwickelte Brad Cox die Sprache Objective-C [2]. Weiter führt Dalrymple aus, dass die Sprache die Vorteile einer schnellen C-Sprache mit den Vorteilen der objektorientierten Sprache SmallTalk verbinden sollte. Die Firma NextSTEP nutzte Objective-C und als NEXTStep von Apple aufgekauft wurde, integrierte Apple Objective-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 [15]. Da jedoch alle highlevel APIs in Objective-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 Objective-C APIs [15].

### 2.3.2. Überblick der Neuerung

las

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 instanziiert wurde. Es ist nicht nötig für die Variable eine Typendefinition zu definieren [1]. 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 inferred 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 Referenz in eine Variablen gespeichert werden. Folgende Code Beispiele sollen dies veranschaulichen. Wir nutzen hierfür einen Asynchrones Netzwerk-Request [1].

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.

Generics oder Protocol  
Extension

### 2.3.5. 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 [1]. Im folgenden Beispiel wird eine Medikation in Java ohne Optional Types und in Swift mit Optional Types modelliert.

Listing 2.4: 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 2.4 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 2.5: 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 2.5 sind die Instanzvariablen nun mit Optional Types modelliert. 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.

## 2.4. Wearables

Übersetzt man es ins deutsche, bedeutet es "Tragbares oder Anziehbares im 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 [3]. 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 [14]. Diese Daten zeigen immer nur einen Teilausschnitt. 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 [14].

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

Swatch Quelle

später die Apple Watch vor, die über einen 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, Sprachsteuerung, sowie Internetverbindung und weitere Sensoren verfügt [9]. Auch ein Ring, der mit Hilfe eines Sensoren den Puls misst existiert als Prototyp [14].

Während im Smartphone-Markt noch einen Fokus auf Leistung und Funktion der Geräte legte, darf man beim Wearables-Markt den Faktor der Ästhetik nicht vergessen. Hier wird ein Bereich betreten, der starke Einflüsse von Mode aufweist. Anwender eines Gerätes achten also nicht mehr nur auf die Funktion, sondern auch auf Form, Farbe und Lifestyle, den das Produkt verkörpert. Apple bietet unter dem Namen "Watch Edition" eine 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 luxuriöse modische Uhren spezialisiert hat, betritt nun auch den Markt der Wearables [13].

## 3. Analyse und Entwurf

In diesem Kapitel Ergebnisse der Analyse und des Entwurfes erläutert. 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
<i>Primärer Akteur</i>	Patient

<i>Beschreibung</i>	Ein Erinnerungs Pop-Up erscheint auf dem Display und es wird ein Signal/eine Vibration ausgelöst. Das Pop-Up zeigt ein Abbild des Medikaments, dessen Namen und die Uhrzeit, zu der es eingenommen werden soll.
<i>Vorbedingung</i>	Es wurde ein Medikationsplan aus der DB auf die Uhr geladen.
<i>Ablauf</i>	<ul style="list-style-type: none"> <li>• Das Pop-Up mit der Erinnerung erscheint auf dem Smartwatch-Display. Die Erinnerung beinhaltet Informationen zur Uhrzeit, Menge und Art der Medikation</li> <li>• Der Patient bestätigt, dass er das Medikament genommen hat</li> <li>• Gerät bestätigt visuell dass der Patient das Medikament als genommen markiert hat</li> </ul>
<i>Alternativablauf</i>	<ul style="list-style-type: none"> <li>• Das Pop-Up mit der Erinnerung erscheint auf dem Smartwatch-Display</li> <li>• Der Patient wählt Option zum Verschieben der Medikation aus</li> <li>• Auf einem zusätzlichen Dialog kann er aus einer Auswahl eine Zeitdauer wählen, um das die Medikation verschoben wird</li> </ul>
<i>Alternativablauf 2</i>	<ul style="list-style-type: none"> <li>• Das Pop-Up mit der Erinnerung erscheint auf dem Smartwatch-Display</li> <li>• Der Patient reagiert nicht auf die Erinnerung</li> <li>• Die Erinnerung wird alle x Minuten wiederholt, solange der Patient nicht reagiert.</li> <li>• Das Fehlen einer Reaktion des Patienten innerhalb einer Zeit von x Minuten wird vermerkt</li> </ul>
<i>Ergebnis</i>	Der Patient hat die Einnahme des Medikaments bestätigt und dieses auch eingenommen
<i>Alternativergebnis 1</i>	Der Patient hat die Erinnerung an die Medikamenteneinnahme verschoben
<i>Alternativergebnis 2</i>	Der Patient hat die Erinnerung an das Medikament ausgeschaltet

<b>Usecase 2</b>	<b>Der Patient wird an mehrere Medikamente erinnert</b>
<i>Primärer Akteur</i>	Patient
<i>Beschreibung</i>	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.
<i>Vorbedingung</i>	Es wurde ein Medikationsplan aus der DB auf die Uhr geladen.
<i>Ablauf</i>	<ul style="list-style-type: none"> <li>• Das Pop-Up mit der Erinnerung erscheint auf dem Smartwatch-Display. Die Erinnerung beinhaltet Informationen zur Uhrzeit, Menge und Art der Medikationen</li> <li>• Der Patient bestätigt, dass er die Medikamente alle genommen hat</li> <li>• Gerät bestätigt visuell, dass der Patient die Medikamente als genommen markiert hat</li> </ul>
<i>Alternativablauf</i>	<ul style="list-style-type: none"> <li>• Das Pop-Up mit der Erinnerung erscheint auf dem Smartwatch-Display</li> <li>• Der Patient wählt Option zum Verschieben der Medikationen aus</li> <li>• Auf einem zusätzlichen Dialog kann er aus einer Auswahl eine Zeitdauer wählen, um das die Medikationen verschoben werden</li> </ul>
<i>Alternativablauf 2</i>	<ul style="list-style-type: none"> <li>• Das Pop-Up mit der Erinnerung erscheint auf dem Smartwatch-Display</li> <li>• Der Patient wählt ein Medikament von der Liste aus</li> <li>• Patient befindet sich nun im Usecase 1</li> </ul>

<i>Alternativablauf 3</i>	<ul style="list-style-type: none"> <li>• Das Pop-Up mit der Erinnerung erscheint auf dem Smartwatch-Display</li> <li>• Der Patient reagiert nicht auf die Erinnerung</li> <li>• Die Erinnerung wird alle x Minuten wiederholt, solange der Patient nicht reagiert.</li> <li>• Das Fehlen einer Reaktion des Patienten innerhalb einer Zeit von x Minuten wird vermerkt</li> </ul>
<i>Ergebnis</i>	Der Patient hat die Einnahme der MEDikamente bestätigt und dieses auch eingenommen
<i>Alternativergebnis 1</i>	Der Patient hat die Erinnerung an die Medikamenteneinnahme verschoben
<i>Alternativergebnis 2</i>	Der Patient hat die Erinnerung an das Medikament ausgeschaltet
<i>Alternativergebnis 3</i>	Der Patient hat bestimmte Medikamente ausgewählt und mit dem weiterführenden Usecase 1 bearbeitet

<b>Usecase 3</b>	<b>Einzelne Einnahmebestätigung zurücknehmen</b>
<i>Primärer Akteur</i>	Patient
<i>Vorbedingung</i>	Es wurde ein Medikationsplan aus der DB auf die Uhr geladen. Eine Medikation wurde als genommen markiert
<i>Ablauf</i>	<ul style="list-style-type: none"> <li>• Das System zeigt das genommene Medikament an</li> <li>• Der Benutzer drückt auf den Button mit der Aufschrift "Rücknahme"</li> <li>• Das System wechselt zur Darstellung eines einzelnen Medikaments, beschrieben im Usecase 1</li> </ul>
<i>Ergebnis</i>	Die Einnahmebestätigung ist zurückgenommen. Die Erinnerung ist erneut zu bestätigen oder zu verschieben.

<b>Usecase 4</b>	<b>Mehrere Einnahmebestätigungen zurücknehmen</b>
<i>Primärer Akteur</i>	Patient
<i>Vorbedingung</i>	Es wurde ein Medikationsplan aus der DB auf die Uhr geladen. Mehrere Medikationen, welche zur gleichen Zeit genommen wurden, wurde als genommen markiert
<i>Ablauf</i>	<ul style="list-style-type: none"> <li>• Das System zeigt die genommenen Medikamente an</li> <li>• Der Benutzer drückt auf den Button mit der Aufschrift "Rücknahme"</li> <li>• Das System wechselt zur Darstellung mehrere Medikamente, beschrieben im UseCase 2</li> </ul>
<i>Ergebnis</i>	Die Einnahmebestätigung ist zurückgenommen. Die Erinnerung ist erneut zu bestätigen oder zu verschieben.

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

## 3.4. Apple Watch

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

### 3.4.1. Hardware

Die Apple Watch existiert in zwei Versionen. Eine 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 [10].

### 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 dem Erscheinen der Uhr wurde auch das Betriebssystem in Version 1 ausgeliefert und dazu das Extensible Markup Language (XML) namens WatchKit. Dies erlaubte es Entwicklern Anwendungen zu entwickeln, welche auf dem verbundenen iPhone ausgeführt wurden. Dies führte zu schlechter Performance der Anwendungen und zu voller Abhängigkeit zum iPhone.

Im Juni 2015 veröffentlichte Apple die erste Vorabversion von watchOS 2.0, welches später im September 2015 für Endnutzer bereitgestellt wurde. watchOS bietet mehr Unabhängigkeit für Anwendungen, da diese direkt auf der Uhr ausgeführt werden. Für Anwendungsentwickler gibt es vier Arten Informationen auf der Uhr darzustellen. Es handelt sich um native Anwendungen (Apps), Glances, Complications, und Actionable Notifications [4].

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 (3.4.6 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 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 [4].



Es gibt zwei verschiedene 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 [5].

### 3.4.4. Schniststellen

Bluetooth 4.0 und Wi-Fi 802.11b/g sind die Netzwerkschnittstellen 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 Bezahl dienst, vorgesehen ist [11].

### 3.4.5. Sensoren

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

Tapie Engine

### 3.4.6. Eingabe Interfaces

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

Accessability

### 3.4.7. Armband

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

Das in der günstigsten Version mitgelieferte 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 bieten eine einfache 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.

### 3.4.8. Prototyping

Zum Erstellen des frühen Prototyps wurden Papier-Prototypen erstellt. Durch die Nutzung dieser Werkzeuge konnte schnell und iterativ gearbeitet werden. Da 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.

### 3.4.9. Evaluation

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

Hierfür werden 2 Fragebögen genutzt. Einmal AttrakDiff, welcher die subjektive Wahrnehmung der Bedienbarkeit und Aussehen des Prototypen erfragt. Es handelt sich um einen standardisierten Fragebogen und eine standardisierte Auswertungsmethode. Da dieser Fragebogen keine Antworten über Funktionen des Prototyps 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

## 4. Umsetzung

Die in Kapitel 3 aufgeführten Analyse Ergebnisse wurden für die erste Iteration 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 zwischen 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 Interface-Elemente mit dem Quellcode verbunden 4.2.

Die Auswahl an Interface-Elemente ist sehr begrenzt und bietet kaum Möglichkeiten eigene Interface-Elemente 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 im Listing xx zu sehen, wie genau eine Verbindung aufgebaut werden kann. Wichtig ist, dass diese Verbindung zum richtigen Zeitpunkt im Application-Lifecycle aufgebaut wird, da es sonst zu Datenverlusten kommen kann.

Mit der `sendMessage:` 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

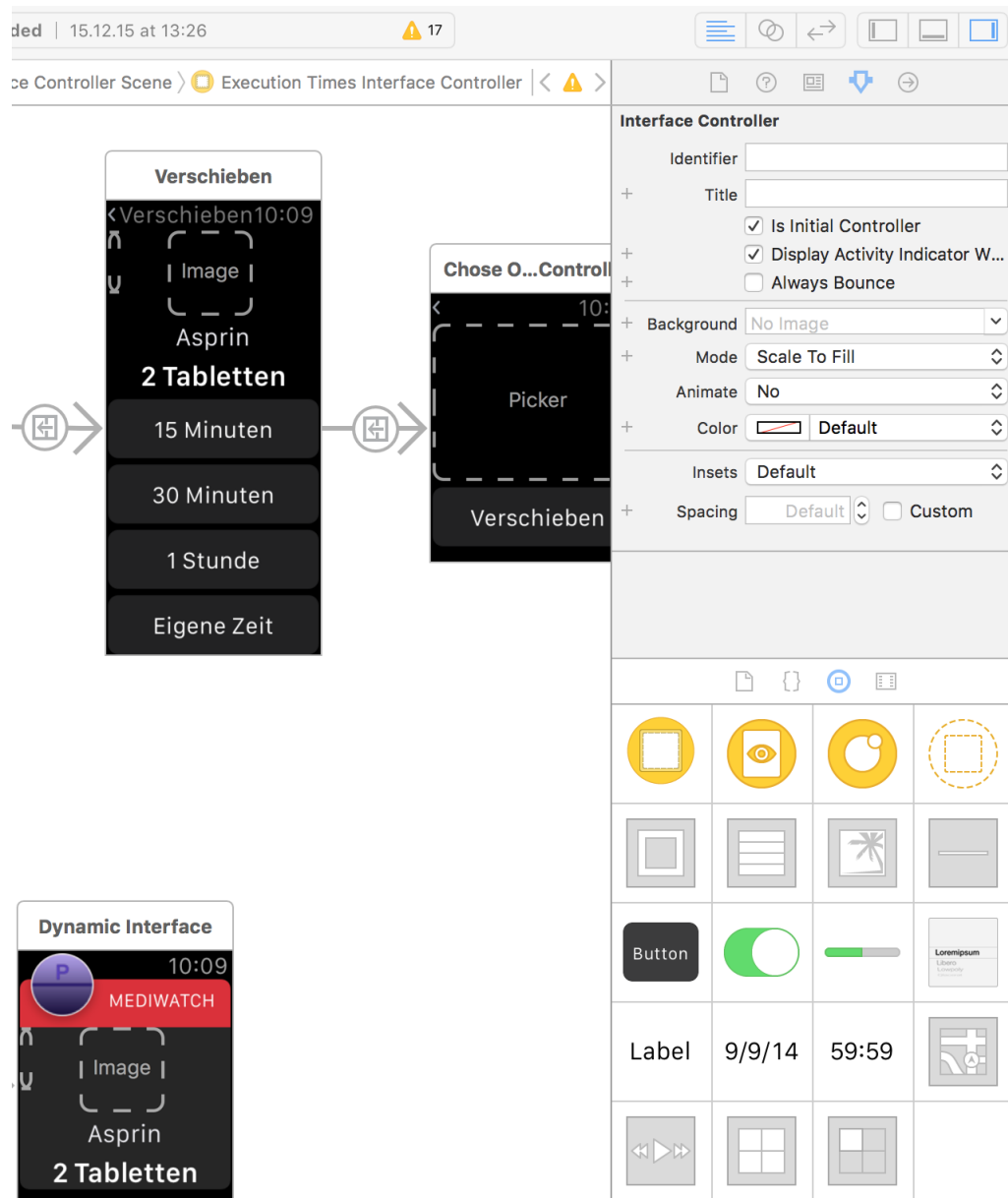
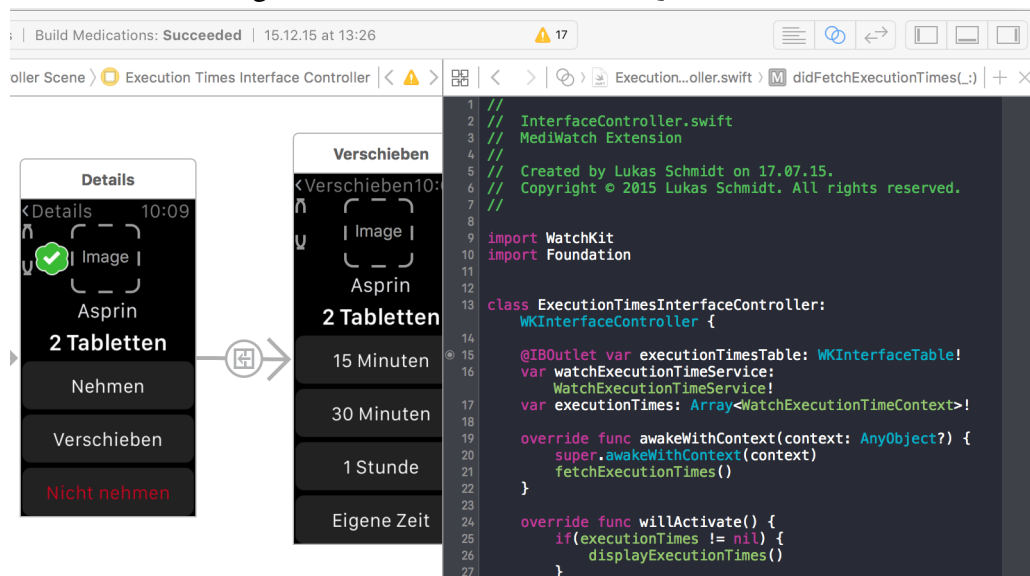


Abbildung 4.2.: Interface Elemente mit Quellcode verknüpfen



eine Datenänderung auf dem iPhone, die relevante Daten für die Uhr enthält, so sollten diese mit der Methode `transferUserInfo:` an 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.

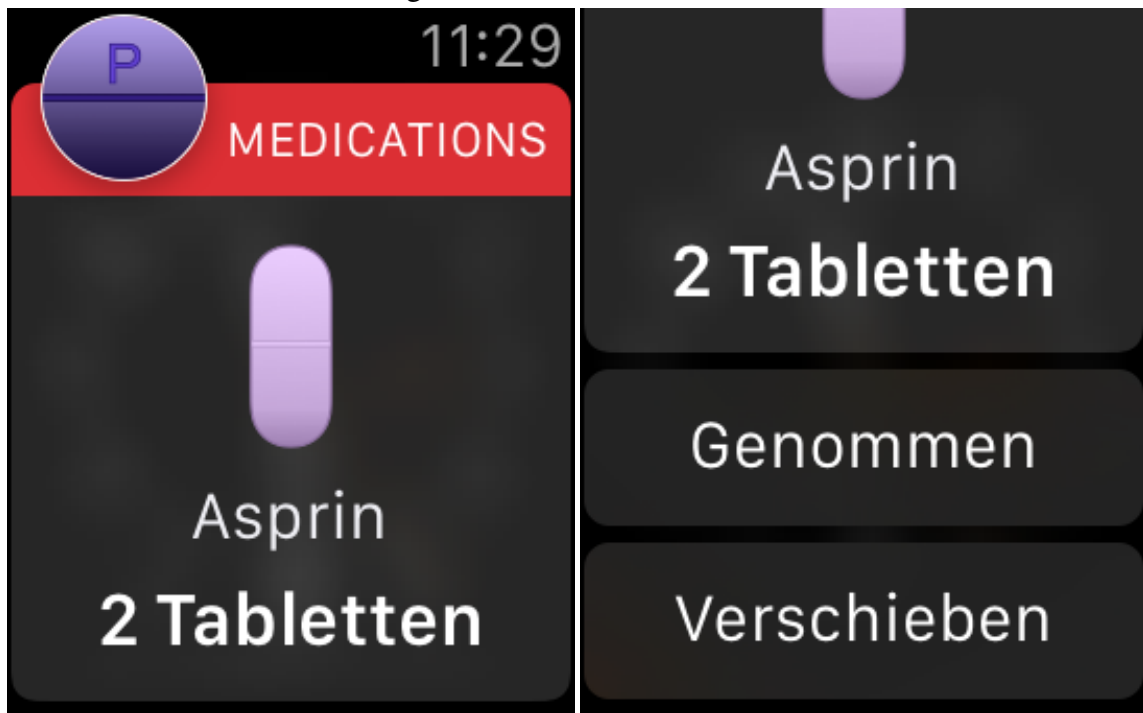
Zum Übertragen von größeren Daten steht die Methode `transferFile:` zur Verfügung. Die Methode verhält sich äquivalent zu `transferUserInfo:`, wird jedoch in der Realisierung nicht verwendet.

### 4.3. Anwendung

Die Anwendung besteht aus 2 Komponenten auf der Apple Watch. Eine native Notification (3.4.3). Diese wurde darauf hin optimiert, dass eine große graphische Repräsentation des Medikaments angezeigt wird. Dem Nutzer wird so die Wiedererkennung des Medikaments 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.

Neben der Notification gibt es noch eine native App. Diese läuft auf der Uhr und stellt eine Liste der heutigen Medikamente dar. Die App muss aktiv vom Nutzer gestartet werden. In 4.4 wird ein Medikament aus der List ausgewählt. Nach dem man ein Medikament

Abbildung 4.3.: Notification für Medikament



ausgewählt hat, kommt man in eine Detailansicht des Medikaments. Dort kann man nun das Medikament als genommen markieren. Ein visuelles Feedback zur Einnahme zeigt dem Nutzer seine Aktion. Von der Detailansicht ist es auch möglich ein Medikament zur späteren Einnahme zu markieren

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







## **5. Zusammenfassung, Evaluation und Ausblick**



# Literaturverzeichnis

- [1] *The Swift Programming Language*. Apple, Inc., 2014.
- [2] DALRYMPLE, M. Learn objective-c on the mac. 1–20.
- [3] DVORAK, J. L. Moving wearables into the mainstream.
- [4] INC., A. Developing for apple watch.
- [5] INC., A. Notification essentials.
- [6] KREBS P, D. D. Health app use among us mobile phone owners: A national survey. *MIR mHealth uHealth* (2015).
- [7] 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.
- [8] 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.
- [9] 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.
- [10] RICHES, G. Apple watch for developers.
- [11] RITCHIE, R. Apple watch specs, 2015.
- [12] SAILER, F., POBIRUCHIN, M., WIESNER, M., AND MEIXNER, G. An approach to improve medication adherence by smart watches, 2015.
- [13] TAGHEUER. Tagheuer connected.
- [14] 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.
- [15] WELLS, G. The future of ios development: Evaluating the swift programming languag. Master's thesis, Claremont McKenna College, 2015.



## **A. Appendix**



## **B. Dokumentation**