Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorthesis

Björn Baltbardis

Eine Software-Architektur zur Benutzerinteraktion beim überwachten Lernen automatisch klassifizierender Systeme

Fakultät Technik und Informatik
Studiendepartment Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Björn Baltbardis

Eine Software-Architektur zur Benutzerinteraktion beim überwachten Lernen automatisch klassifizierender Systeme

Bachelorthesis eingereicht im Rahmen der Bachelorprüfung

im Studiengang Angewandte Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Ulrike Steffens
Zweitgutachter: Prof. Dr. Kai von Luck

Eingereicht am: 10. November 2014

Björn Baltbardis

Thema der Arbeit
Eine Software-Architektur
zur Benutzerinteraktion beim überwachten Lernen
automatisch klassifizierender Systeme

Stichworte
Interaktionsmuster, Smart Home, Wearable, Android Wear, Smartwatch, Maschinelles Lernen, Software Architektur

Kurzzusammenfassung
Lernende Systeme werden immer relevanter für unseren Alltag. Zunehmend gilt es deshalb die Frage zu klären, wie wir mit ihnen interagieren können. Ein aufstrebendes Anwendungsgebiet lernender Systeme sind Smart Home Umgebungen, das heißt Wohnraum, der die Lebensqualität seiner Bewohner durch den Einsatz von Technik erhöht. Ziel dieser Arbeit ist es, am Beispiel eines lernenden Smart Home, ein geeignetes Interaktionskonzept zu entwickeln und softwaretechnisch umzusetzen. Dabei soll auf Wearables und das Betriebssystem Android Wear zurückgegriffen werden.
Björn Baltbardis

Title of the paper
A software architecture
for user interaction in supervised learning of
automatically classifying systems

Keywords
Patterns of interaction, Smart home, Smart environment, Wearable, Android Wear, SmartWatch, Machine Learning, Software Architecture

Abstract
Learning systems are gaining growth in our daily life. It is increasingly important to clarify the question of how we could interact with them. One application of artificial intelligence are smart home environments. It is therefore the objective of this thesis to develop a concept with appropriate interaction patterns for such a system as well as implementing a suitable software architecture. Furthermore, wearable devices using Android Wear should be used to achieve this goal.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einführung

Maschinelles Lernen ist eine Disziplin, die sich in den letzten Jahren wachsender Popularität erfreut. Zuletzt etwa mündete dieser Trend in den milliardenschweren Käufen der Unternehmen *DeepMind Technologies Ltd.* und *Nest Labs Inc.* durch das Internet-Unternehmen *Google Inc.* (vgl. El-Sharif, 2014a,b).

Waren solche lernenden Systeme früher häufig nur zu Forschungszwecken in Verwendung, erleben sie heute eine Wiederauflage in ihrer alltäglichsten Form. Selbst Internet-Unternehmen beschäftigen mittlerweile ganze Abteilungen von *Data Scientists*, deren Kernthematik die automatische Auswertung von Daten ist.

Je alltäglicher aber unsere Kontaktpunkte mit künstlicher Intelligenz werden, desto mehr Aufmerksamkeit gilt es auch den Interaktionsmustern unserer Einflussnahme zu widmen. Sie beschreiben bei definiertem Verhalten des Nutzers das erwartete Kommunikationsverhalten des Systems. Durch sie ist die Frage zu klären, wie man den Nutzer sinnvoll und effektiv in den Lernprozess einbinden kann bzw. dessen Ergebnisse angemessen präsentiert oder nutzt.

1.1 Motivation und Zielsetzung

Ein aufstrebendes Anwendungsgebiet lernender Systeme mit erhöhtem Interaktionsbedarf sind Smart Home Umgebungen. Sowohl kommerzielle Anbieter, als auch die Forschung sind derzeit sehr aktiv in diesem Bereich (vgl. Kap. 3.1 und 3.4). Es gibt bereits viele Interaktionskonzepte, aber die wenigsten sind besonders praktikabel oder thematisieren die Interaktion mit lernenden Komponenten als Teil des Smart Home (vgl. Kap. 3.4).

Ziel dieser Arbeit ist es deshalb, am Beispiel eines lernenden Smart Home, ein Interaktionskonzept zu entwickeln und softwaretechnisch umzusetzen, das den Nutzer möglichst effektiv in den Prozess des Lernens mit einbindet. Aus diesem Grunde werden mehrere Interaktionsmuster entwickelt, die Wünsche des Bewohners mit einem unterschiedlichen Grad an Autonomie begegnen. Denn so lange die Vorhersagen künstlicher Intelligenz fehlbar sind, so lange müssen dem Nutzer auch manuelle oder teilautonome Konzepte der Steuerung geboten werden.

Übergeordnet ist dieser Ansatz unter dem Begriff *Ambient Intelligence* (Umgebungsintelligenz) einzuordnen. Ambient Intelligence ist nach Ricardo Costa ein relativ neues Pardigma der Informationstechnologie, nach dem Menschen durch ein System unterstützt werden, das sich deren Präsenz, Wünschen und Gewohnheiten bewusst ist (vgl. Costa u. a., 2009). Es soll *einfühlsam*, *anpassungsfähig* und *zugänglich* sein. Nach Jean-Baptiste Waldner handelt es sich bei Ambient Intelligence sogar um den evolutionären Nachfolger des *Ubiquitous computing* bzw. des ersten mobilen Trends (vgl. Waldner, 2013). Es geht darum, die uns umgebenden Systeme intelligenter zu machen und gleichzeitig einen geeigneten Zugang für Nutzer zu bewahren.

Ein geeignetes Konzept steht und fällt hier folglich mit einem ausgereiften und unmittelbarem Interaktionsmodell. Diese unmittelbare Interaktionsmöglichkeit mit dem lernenden System kann in der heutigen Zeit durch sog. Wearables gewährleistet werden. Ihr Absatz verzeichnete zuletzt ein starkes Wachstum (s. Kap. 2.1) und sie bieten eine geeignete Ausgangssituation den Anforderungen der Umgebungsintelligenz gerecht zu werden. Das Konzept ist zunächst allgemeingültig für Wearables. In der praktischen Umsetzung soll aber eine Smartwatch mit dem Betriebssystem *Android Wear* zum Einsatz kommen.

Weiterhin wird von dem Vorhandensein einer bestehenden Smart Home Infrastruktur ausgegangen. Ein Teil dieser Arbeit thematisiert deshalb die Einbindung in ein solches System, sowie die notwendige, technische Kommunikation.

1.2 Abgrenzung

Die Entwicklung eines selbstlernenden Systems, sowie die Beschreibung des eigentlichen maschinellen Lernprozesses, ist nur am Rande ein Teil dieser Arbeit. Das Gros der Aufmerksamkeit gilt dem zu entwerfenden Interaktionskonzept, sowie der Software- und Schnittstellenarchitektur.
Ohnehin liegt es nahe, dass eine abstrahierte Interaktion mit dem Benutzer Vorteile für diesen hat. Die Loslösung dieser Arbeit vom darunterliegenden Lernprozess verringert das Risiko einer zu technikennahen Orientierung der Interaktionsmuster an den Lernprozessen der jeweiligen Smart Home Umgebung. Auf diese Weise ist die Interaktion nicht gekoppelt an die Implementation des Lernprozesses und der Benutzer kann diesen natürlich vorantreiben.

1.3 Gliederung der Arbeit

Im Kapitel *Grundlagen* sollen zunächst die nötigen Voraussetzungen für die ganz unterschiedlichen, sich in dieser Arbeit befindlichen Kernthemen, geschaffen werden. Es wird unter anderem auch deren Aktualität und gegenseitige Eignung verdeutlicht werden.
Anschließend folgt das Kapitel *Analyse*. Hier sollen Anforderungen und Ziele der prototypischen Entwicklung herausgestellt werden. Es wird untersucht, welche Bedienkonzepte sinnvoll und welche weniger sinnvoll bzw. in dem gewählten Umfeld gar nicht umsetzungsfähig sind.
In dem nachfolgenden Kapitel *Entwurf* werden der Aufbau und die Besonderheiten des Systems erläutert. Die Anforderungen des Analyse Kapitels werden in ein Interaktionskonzept umgewandelt. Es wird dabei genau so auf die softwaretechnischen Designentscheidungen, wie auf die Schnittstellenwahl des verteilten Systems eingegangen.
Die abschließenden Kapitel *Auswertung* und *Zusammenfassung und Ausblick* gleichen die, in der Analyse entwickelten Anforderungen, mit der Umsetzung ab. Es wird untersucht, an welchen Stellen noch optimiert werden kann, und gleichermaßen gezeigt, wie ein optimiertes System in Zukunft aussehen könnte.

2 Grundlagen

Dieses Kapitel beschreibt alle, für das Verständnis dieser Arbeit wichtigen, Grundlagen und Lehrmeinungen. Es dient der Verdeutlichung der maßgebenden Technologien und Konzepte, die in dieser Arbeit eingesetzt werden.

2.1 Wearables

Ein entscheidender Teil dieser Thesis soll durch die Interaktion mit sog. Wearables (*eng. „tragbar“*) bestimmt sein. Ein Wearable ist ein am Körper getragenes Computersystem. Haupttätigkeit des Trägers ist i.d.R. nicht die Benutzung des Wearables, sondern das Wearable unterstützt den Nutzer bei anderen physischen Tätigkeiten. Denkbar sind alltägliche Situationen bis hin zu sehr speziellen Anwendungsszenarien. Häufig zeichnen sich Wearables zudem dadurch aus, dass sie auf den Nutzer oder seine Umwelt bezogene Daten aufzeichnen und verarbeiten.

Abb. 2.1: Datenbrille Google Glass, vorgestellt im Juni 2012 Torborg u. Simpson (2013)

Wearables sind derzeit so aktuell und gefragt wie nie zuvor. Einige vielversprechende Wearables, wie die intelligente Datenbrille *Google Glass* (s. Abb. 2.1), sind schon länger auf dem Markt. Zuletzt aber sorgten Google Inc. und Apple Inc. durch das Wearable-Betriebssystem *Android Wear* bzw. die *Apple Watch* für Aufsehen.

Zahlreiche Hardware- aber auch Uhrenhersteller erkennen offenbar ein großes Potential und treffen Vorbereitungen, sich im Wearable-Markt zu platzieren. Eine erste Übersicht eröffnet die Liste, der mit Google Inc. kooperierenden Unternehmen (s. Abb. 2.2).

Auch die Zahlen zeigen einen deutlichen Trend: Joshua Flood, von ABI Research, prognostizierte auf der *11. Wearable Technologies Conference* in München, dass allein im

Abb. 2.2: Hardware Kooperationspartner von Google Inc. bei der Nutzung und Verbreitung von Android Wear Google Inc. (2014a)

Jahr 2014 mit einem Verkauf von 90 Millionen Wearables zu rechnen sei (vgl. Pai, 2014). Nachfolgende Tabelle als Auszug derselben Studie zeigt, dass zwischen den Jahren 2013 und 2015 mit einer Steigerung von insgesamt über 200% auf ein Dreifaches des bisherigen Absatzes zu rechnen ist (s. Tbl. 2.1).

	2013	**2014**	**2015**
Wearable Kameras	6,64	13,61	15,81
Smart Glasses	0.01	2,13	10,57
Smart Watches	1,23	7,44	24,92
Healthcare	13,45	22,59	34,35
Sonstige	32,49	44,23	60,66
Summe	53,90	90,00	164,20

Tabelle 2.1: Prognostizierter weltweiter Wearable-Absatz (in Millionen)
Quelle: „ABI Research World Market Forecast 2013 to 2019“ Pai (2014)

Die Tabelle zeigt ebenfalls eine prognostizierte Absatzsteigerung von etwa 2000% für Smartwatches zwischen den Jahren 2013 und 2015.

Laut dem Zukunftsforscher und Unternehmensberater Dr. Ian Pearson sind Datenbrillen wie Google Glass bereits in fünf Jahren selbstverständlich für unser Leben und schon in weiteren fünf Jahren vermutet er dieselbe Technik in Kontaktlinsen. Pearson geht davon aus, dass Wearables, wie Smartwatches, schon in wenigen Jahren das Straßenbild beherrschen und aus unserem Alltag - ebenso wie heute Smartphones - kaum mehr wegzudenken sind (vgl. Pearson, 2013).

Eine weitere Studie der Bitkom zeigt zusätzlich eine wachsende Bereitschaft, Wearables wie Smartwatches zu tragen: Waren es 2013 noch 31% der Befragten, so sind es 2014 bereits 38%. Bei *Smart Glasses* ist diese Zahl im gleichen Zeitraum sogar um 11% auf

insgesamt 31% gestiegen.
Die wachsenden Zahlen hängen wahrscheinlich auch damit zusammen, dass immer noch rund die Hälfte der Befragten nicht weiß, was Smartwatches und *Smart Glasses* sind. Eine wachsende Bekanntheit lässt im Rahmen der genannten Zahlen also auch eine zukünftig wachsende Nutzungsbereitschaft erwarten (vgl. Bitkom, 2014).

Ob sich einzelne Produkte, wie die von Google Inc. oder Apple Inc. durchsetzen werden, bleibt offen. Als wahrscheinlich aber gilt, dass Art und Formfaktoren unser mobilen Geräte sich ändern werden. Näheres zu dieser evolutionären Entwicklung ist in dem Kapitel 2.2 *Smart Home und Ambient Intelligence* zu finden.

2.1.1 Mehrwert von Wearables

Wearables erzeugen durch einige Eigenheiten einen Mehrwert gegenüber herkömmlichen Geräten, wie Smartphones oder Tablets. Der erste dieser Vorteile mag trivial erscheinen und ergibt sich bereits aus der Definition eines Wearables: Das Gerät wird unmittelbar am Körper getragen. Smartwatches beispielsweise befinden sich am Handgelenk des Nutzers und sind viel präsenter als ein, sich in der Hosen- oder gar Handtasche befindliches Smartphone. Dem Anwender fällt die Interaktion leichter und die Nutzung lässt sich einfacher in dessen Alltag integrieren. Informationen können weiterhin unauffällig und sozialverträglich abgerufen werden, während die Hände des Nutzers frei bleiben. Durch die angesprochene, erhöhte Präsenz birgt die Kontextabhängigkeit von Wearables ein vielfach höheres Potential als bei Smartphones. Informationen und Hilfen können dem Nutzer dank Datenbrillen und Smartwatches in dem Moment offeriert werden, in dem er sie benötigt.
Diese *unmittelbare Kommunikation* soll sich im Rahmen dieser Arbeit zu Nutze gemacht werden, um eine möglichst direkte und dem Stand der Technik entsprechende Verständigung mit lernenden Systemen zu erzielen.

2.2 Smart Home und Ambient Intelligence

Dieser Abschnitt ist dem übergeordneten Kontext gewidmet, in dem diese Arbeit einzuordnen ist. *Smart Home*, sowie *Ambient Intelligence*, stellen dabei zwei zentrale Begriffe dar, die im Folgenden betrachtet werden sollen.

Smart Home ist ein Oberbegriff für technische Geräte und Verfahren mit dem Ziel, die Lebensqualität, Sicherheit oder Effizienz in Wohnumgebungen zu steigern. Dazu dienen in der Regel vernetzte Systeme und automatisierbare Abläufe. Praktisch drückt sich der Begriff häufig in dem Vorhandenensein von beispielsweise einer zentrale Licht-, Heizungs-, Tür- und Sicherheitssteuerung aus. Sind ältere, oder körperlich eingeschränkte Menschen Nutzer der Technik, spricht man meist von Ambient Assisted Living. Wann ein System in diesem Kontext als *smart*, also intelligent gilt, ist nicht klar umgrenzt.

Der zweite, eingangs genannte Begriff, lässt sich etwas präziser beschreiben. Ambient Intelligence (AmI) beschreibt Systeme, die sich der Präsenz, den Wünschen und den Gewohnheiten von Menschen bewusst werden können und dabei als *einfühlsam*, *anpassungsfähig* und *zugänglich* wahrgenommen werden (vgl. Costa u. a., 2009). Auch hier geht es um die Unterstützung von Menschen in ihrem Alltag. Die explizit notwendige Interaktion soll nach Möglichkeit auf ein Minimum reduziert werden (vgl. Augusto u. a., 2010).
AmI lässt sich nicht auf eine einzige Technologie zurückführen, sondern ist nach Augusto u. a. als gelungenes Gesamtkonzept von Vernetzung, Sensoren, Interaktion und intelligenten Systemen zu verstehen.

Wie die vorausgegangene Einführung zeigt, sind die Begriffe *Smart Home* und *Ambient Intelligence* eng verwandt miteinander und bedienen sich auch teilweise einander. Die verwendeten Gerätschaften werden in beiden Konzepten stets kleiner, besser vernetzt und integrieren sich zunehmend nahtloser in ihre Umgebung.
Waldner beschreibt diese Entwicklung in seiner Arbeit als Evolution des Computers. Einst gestartet als Komposition aus Hauptkomponenten und Userinterface, kamen Netzwerkkommunikation und ein effizienteres Energiemanagement hinzu (vgl. Abb. 2.3).
Der erste mobile Trend ist entstanden und findet seinen Nachfolger durch rasanten Zufluss an Devices im sog. *Ubiquitous computing*. Seit dem Jahr 2005 sind die Hardwarevoraussetzungen so gut, dass von der nächsten Evolutionsstufe, dem *Disappearing computing* bzw. Ambient Intelligence gesprochen werden kann. Heute gibt es viele Anwendungen, die sich nahezu unsichtbar in unsere Umgebung integriert haben und täglich von uns genutzt werden, ohne dass wir sie überhaupt wahrnehmen.

Auch das in dieser Arbeit entwickelte Konzept soll sich nahtlos in seine Umgebung integrieren und explizite Interaktion mit dem Nutzer auf das Nötigste reduzieren. Lernende Systeme sollen es einfühlsam machen, während durch Wearables Zugänglichkeit sichergestellt wird und die später vorgestellten Interaktionsmuster anpassungsfähig gegenüber

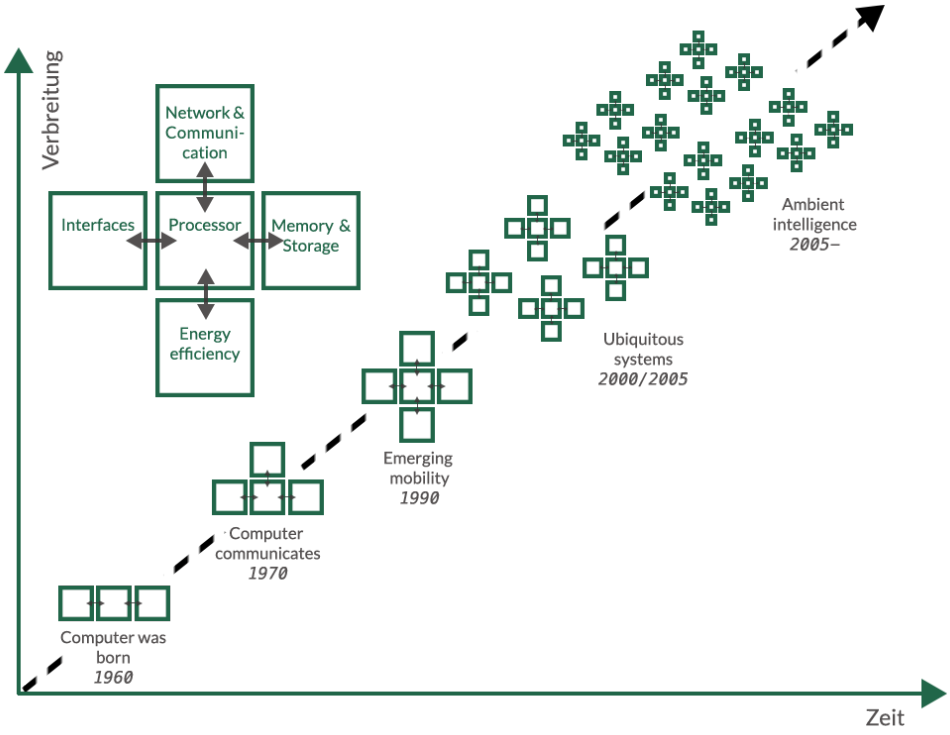


Abb. 2.3: Technologieevolution des Computers (in Anlehnung an Waldner (2013))

dem Nutzer sind. Die entwickelte Steuerung soll den Bewohner unterstützen, ohne selbst im Vordergrund zu stehen.

2.3 Maschinelles Lernen und Klassifikation

> *The structure of the intelligent behavior can become so complex that it is very difficult or even impossible to program close to optimally, even with modern high-level languages [..].*
>
> – Prof. Dr. Wolfgang Ertel, *Introduction to Artificial Intelligence* (Ertel, 2011, S.161)

Nach diesem Zitat aus dem Standardwerk *Introduction to Artificial Intelligence* lässt sich in der Softwareentwicklung zur Lösung gewisser Problemstellungen ein Grad von Komplexität erreichen, ab dem es nicht mehr sinnvoll oder sogar unmöglich ist, eine

zufriedenstellende Lösung auf herkömmlichen Wege zu gewährleisten. Letzteres meint die klassische Softwareentwicklung, verbreitete Hochsprachen und einfache, regelbasierte Systeme. Der logische, folgende und letztlich auch notwendige Schritt ist die Entwicklung von (selbst)lernenden Systemen, welche in ihrer Funktionsweise nicht selten dem menschlichen Lernen nachempfunden sind. Häufig findet man diese Systeme auch in kombinierter Form mit klassischen Systemen vor (vgl. Ertel, 2011, S.161).

Beim maschinellen Lernen wird das gewünschte System nicht im herkömmlichen Sinne programmiert. Es wird vielmehr so entwickelt, dass es aufgrund von gesammelten Erfahrungen daraus lernen und diese Lernerfahrungen folglich in Zukunft zu Rate ziehen kann.
Das maschinelle Lernen kann zusätzlich durch Vorgabe der zu erwartenden Ausgaben unterstützt werden. Die Vorgaben erfolgen durch einen sog. *Lehrer*. Man unterscheidet beim maschinellen Lernen je nach Notwendigkeit eines Lehrers grob zwischen den Verfahren des „überwachten Lernens“ (*supervised learning*), des „unüberwachten Lernens“ (*unsupervised learning*) und dem des „teilweise überwachtem Lernens“ (*semi-supervised learning*) (vgl. Ertel, 2011, S.214).

Für diese Arbeit soll sich des überwachten Lernens bedient werden, um Benutzergewohnheiten zu erkennen und auf deren Basis Vorhersagen treffen zu können. Grundlage dafür sind Kontextdaten wie beispielsweise Zeit oder Aufenthaltsort. Durch entsprechende Lernverfahren kann entschieden werden, wie die derzeitigen Kontextdaten interpretiert werden sollen, ob sie beispielsweise der einen oder anderen Klasse zuzuordnen sind. Diesen Vorgang nennt man Klassifikation, die berechnende Einheit Klassifikator. Auf Basis der erkannten Klasse können dann Entscheidungen getroffen werden.

Für die Klassifikation bieten sich in der Praxis diverse Lernverfahren an. Genannt seien an dieser Stelle *künstliche neuronale Netze* und sog. *Support Vector Machines*. Sie sollen nachfolgend an einem stark vereinfachten Beispiel betrachtet werden:

Man stelle sich eine Lampe vor, die sich stets etwa ab Mittag einschalten und bis abends brennen soll. Der Besitzer ist allerdings sehr geizig und schaltet die Lampe meist nur an, wenn er genug Geld für die Stromrechnung hat. Aus diesen Benutzergewohnheiten ergibt sich ein Bild, wie es Abbildung 2.4 veranschaulicht.
Die Lampe wird stets mit zunehmender Zeit durch den Nutzer eingeschaltet (dunkelgrün). Hat er viel Geld, ist er geneigt, sie früher anzuschalten, als bei einem geringen Kontostand. Die Punkte repräsentieren für die jeweiligen Lernverfahren die Trainings-

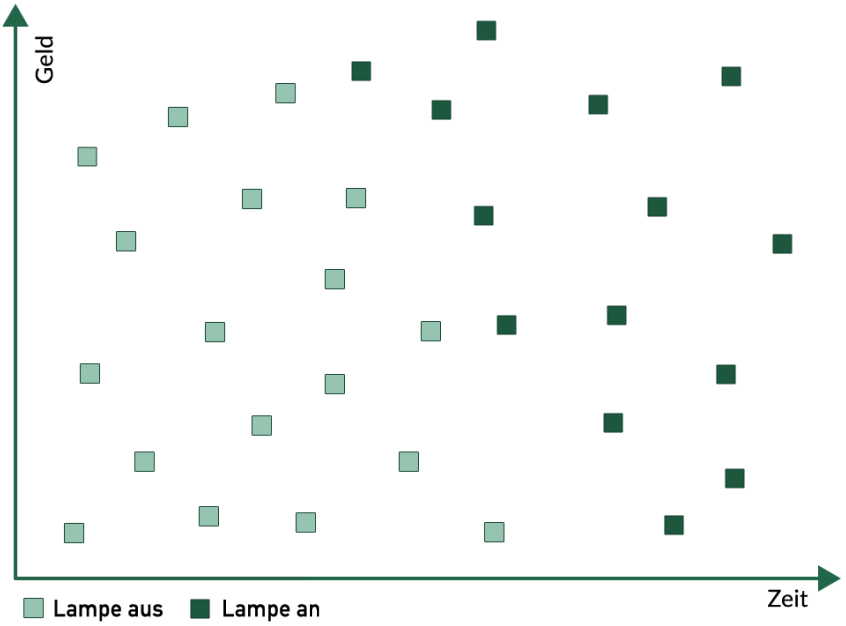


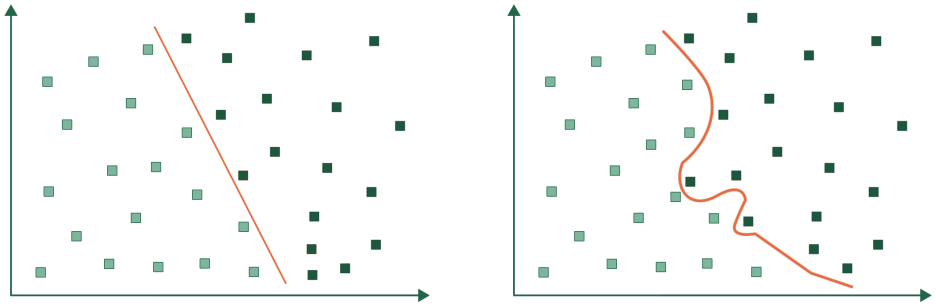
Abb. 2.4: Beispiel: Trainingsdaten zur Klassifikation

daten. Ziel ist es, dass das System auf Grundlage alter Trainingsdaten in die Lage zu versetzen, Situationen mit neuen Daten gleichwertig zu klassifizieren, also das Verhalten des Nutzers vorherzusagen.

2.3.1 Support Vector Machine

Eine *Support Vector Machine* (SVM) unterteilt eine Menge dieser Datensätze so in Klassen, dass um die Klassengrenzen herum ein möglichst breiter Bereich frei von Punkten bleibt. Eine solche Unterteilung ist als orangefarbene Linie in Abbildung 2.5a eingezeichnet. Für die Umsetzung der SVM wird mit Stützvektoren gerechnet, welche Namensgeber dieses Klassifikators sind. Das mathematische Verfahren soll jedoch an dieser Stelle nicht weiter betrachtet werden.

Wichtig ist dagegen, dass es unter realen Bedingungen deutlich mehr als zwei relevante Merkmale (hier: Geld und Zeit) gibt. Statt zweidimensional ist der sog. Merkmalsraum also meist mehrdimensional. Das Problem ist nicht mehr visualisierbar, aber trotzdem berechenbar.



(a) Linear separierbar (b) Nicht linear separierbar

Erschwerend ist darüber hinaus, dass die Daten meist nicht linear separierbar sind, sich also nicht durch eine einfache lineare Funktion in Klassen trennen lassen (vgl. Abb. 2.5b). Es gibt Ausreißer und Messfehler. Dies müssen die Support Vector Machines durch die Berücksichtigung von Toleranzen beachten.

2.3.2 Künstliche neuronale Netze

Ein künstliches neuronales Netz ist ein Wissensspeicher, der dem biologischen Vorbild des menschlichen Gehirns folgen soll. Es besteht aus einer Menge von Neuronen, die durch gerichtete und gewichtete Verbindungen miteinander verknüpft und in Schichten (sog. *Layer*) organisiert sind (vgl. Cleve u. Lämmel, 2014). Obligatorisch sind dabei die Eingabe- bzw. Ausgabeschicht. Optional gibt es eine oder mehrere Zwischenschichten (vgl. Abb. 2.6).
Um ein künstliches neuronales Netz für seinen vorgesehenen Einsatz vorzubereiten, muss es trainiert werden. Die bereits bekannten Daten aus Abbildung 2.4 könnten beispielsweise als Trainingsdaten dienen. Das eigentliche Training kann durch die sog. Backpropagation erfolgen. Dafür wird, für jeweils einen Satz an Trainingdaten, analysiert, wie die Ausgabe des Netzes ist. Die Ausgabe wird mit der erwarteten Ausgabe verglichen und die Abweichung als Fehler des Netzes betrachet (vgl. Cleve u. Lämmel, 2014). Der Fehler wird dann rückwärts bis zur Eingabeschicht zurück propagiert, d.h. die Gewichtungen zwischen den Neuronen durch ein mathematische Verfahren angepasst. Das Training endet zum Beispiel, wenn der durchschnittliche Fehler unter einen vorher festgelegten Wert fällt.
Die Eingabeschicht könnte im Beispiel aus zwei Neuronen bestehen, die Uhrzeit und Geld als kontinuierlichen Wert zwischen 0 und 1 als Eingabe erhalten. Die Ausgabe

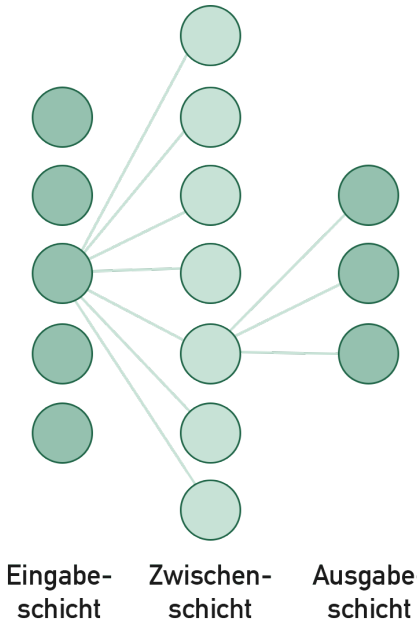


Abb. 2.6: Ein künstliches neuronales Netz

könnte auch durch einen kontinuierlich Wert oder in Form einer boolschen Ausgabe (Lampe an/aus) erfolgen.

2.4 Android

Android ist eines der am weitesten verbreiteten Betriebssysteme für Smartphones und Tablets. Nach aktuellen Statistiken wird es auf rund 84,6% aller Smartphones genutzt (vgl. Strategy Analytics, 2014). Android wurde im Jahr 2007 erstmals angekündigt und wird seitdem durch die *Open Handset Alliance* unter Federführung von Google kontinuierlich weiterenwickelt. Mittlerweile ist Android in Version 5.0 verfügbar. Wie Abbildung 2.7 veranschaulicht, basiert es auf einem Linux Kernel, auf dem die sog. *Android Runtime* und Kernbibliotheken (*Libraries*) aufbauen. Darauf basieren wiederum das *Application Framework* und die Applikationen.
Die einzelnen Komponenten sollen an dieser Stelle nicht alle im Detail betrachtet werden. Da es für das zukünftige System allerdings unabdingbar ist, das Smartphone jederzeit

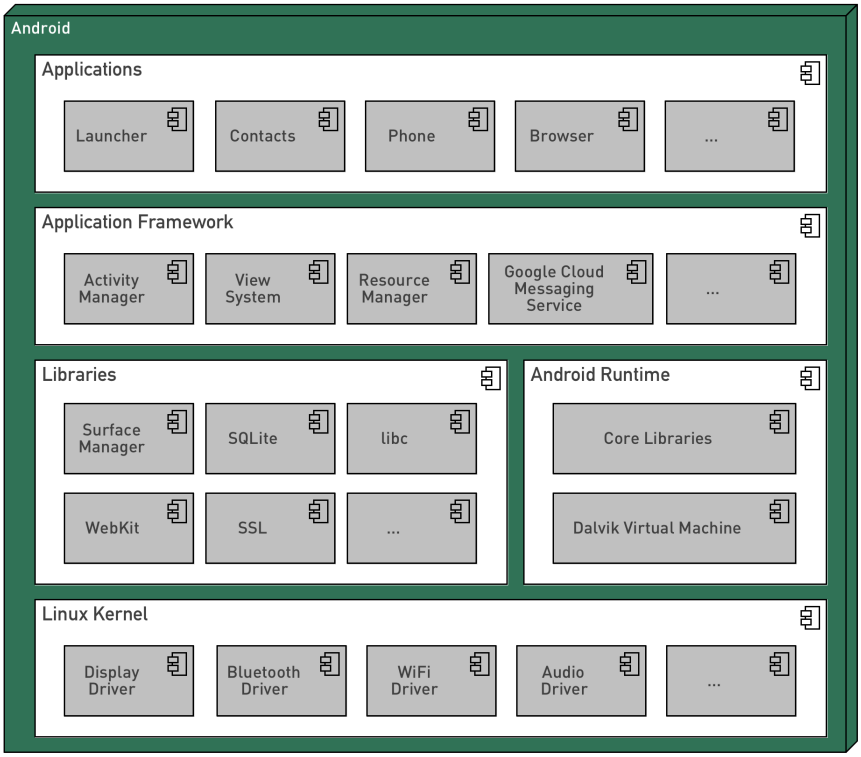


Abb. 2.7: Vereinfachte Android Architektur (in Anlehnung an Open Handset Alliance, 2014)

mit Informationen versorgen zu können, soll nachfolgend die *Google Cloud Messaging* Technologie aus dem *Application Framework* etwas genauer betrachtet werden.

2.4.1 Google Cloud Messaging

Google Cloud Messaging (*kurz:* GCM), auch als Push-Benachrichtigungen bezeichnet, ist ein kostenloser Dienst, der es ermöglicht Nachrichten an Geräte mit Android Betriebssystem zu senden. GCM verwaltet Warteschlangen der Nachrichten automatisch und übernimmt die Übertragung an Geräte. Letzteres ist aufgrund der in der Praxis häufig wechselnden Netzwerke keine triviale Aufgabe und wurde erst im Verlauf der letzten Jahre ausreichend gelöst. Intern teilt das Smartphone dem GCM Server ständig seine aktuelle Adresse mit, unter der es erreichbar ist.
Vor der Einführung von GCM haben alle Entwickler, die einen solchen Dienst benötigten, zum Teil auf eigene oder wenig verbreitete Lösungen zurückgegriffen. In der Summe kam es durch die vielen verschiedenen Ansätze zu höherem Netzwerkverkehr und damit

verbundener Akku-Nutzung, als es bei einer einzigen, zentralen Lösung wie GCM der Fall ist. Hauptvorteil von GCM ist folglich, dass auf dem Smartphone nur ein einziger Server-Socket nach außen hin offen gehalten werden muss.

Abb. 2.8: Schematische Darstellung des Google Cloud Messaging

Um den GCM Dienst zu nutzen, muss sich das jeweilige Client Gerät zunächst am GCM Server anmelden. Es sendet dabei eine *Application ID* an den Server und erhält sein spezifisches *Device Token*. Dieses muss es dem Anwendungsserver mitteilen, damit er es speichern und in Zukunft für die Adressierung von Nachrichten nutzen kann. Nun kann der Anwendungsserver über den GCM Dienst Nachrichten an ein bekanntes Smartphone senden (vgl. Abb. 2.8). Er sendet die Nachricht dazu zunächst an den GCM Server, wo sie mit anderen Nachrichten in einer Warteschlange gespeichert wird, bis das Ziel-Gerät online ist. Die Nachricht wird an das Gerät weitergeleitet und an die entsprechenden Dienste der Applikationen verteilt. Die eigentliche Applikation muss dafür nicht laufen und auch das Gerät kann im StandBy Modus sein. Die oben genannten Dienste werden einfach im Android Manifest für ihren Nachrichtentypen registriert (vgl. Google Inc., 2014d).

2.5 Android Wear

Das von Google entwickelte Betriebssystem Android Wear wurde im März 2014 vorgestellt. Es ergänzt Googles Smartphone Betriebssystem *Android* und reiht sich in die Familie von *Android TV* und *Android Auto* ein. Konzipiert ist es für Smartwatches und andere sog. Wearables.

Diese Geräte können ein rechteckiges, oder auch ein rundes Display aufweisen. Sie haben in der Regel keine direkte Internetverbindung und autark nur einen sehr begrenzten Funktionsumfang. Der eigentliche Mehrwert ensteht durch die Kopplung der Smartwatch über Bluetooth mit einem Android Smartphone. Hervorzuheben sind die Möglichkeiten,

Benachrichtigungen zwischen Smartphone und Smartwatch synchronisieren zu lassen, die Spracheingabe zu nutzen, vollwertige Anwendungen zu entwickeln und mit diesen Daten auszutauschen. Im Folgenden soll zunächst auf diese vier Möglichkeiten eingegangen werden.

2.5.1 Synchronisierte Benachrichtigungen

Das Smartphone Betriebssystem Android bietet den installierten Applikationen (*Apps*) ein ausgereiftes Benachrichtigungssystem. Durch Android Wear können diese Benachrichtigungen ohne weiteres Zutun von Benutzer oder Entwickler automatisch auch auf gekoppelten Wearables erscheinen (s. Abb. 2.9).

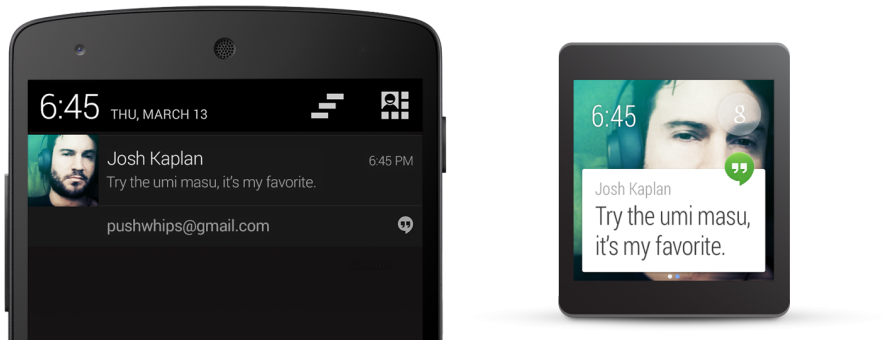


Abb. 2.9: Die selbe Benachrichtigung auf Smartphone und Smartwatch Google Inc. (2014b)

Wird auf einem der Geräte eine Benachrichtigung entfernt, so verschwindet sie auch auf dem anderen Gerät. Die möglichen Aktionen (*Actions*), welche auf dem Smartphone als Buttons (virtuelle Tasten) unter der eigentlichen Benachrichtigung erscheinen, sind auch auf dem Wearable auslösbar. So lassen sich beispielsweise bei eintreffenden E-Mails Titel und Absender lesen und anschließend kann der Nutzer die E-Mail durch ein seitliches Wischen und dem Klick auf „Archivieren“ archivieren. Auch kann er Benachrichtigungen - wie auf dem Smartphone - durch ein Streichen in die entgegengesetzte Richtung entfernen.
Mit bereits relativ geringem Aufwand lassen sich Benachrichtigungen für Android Wear anpassen, sodass sie auf dem Wearable andere Aktionen anbieten als direkt auf dem Smartphone.

2.5.2 Spracheingabe

Abb. 2.10: Spracheingabe Android Wear Google Inc. (2014b)

Android Wear bietet im sog. *aktiven Modus* eine überraschend zuverlässige Spracherkennung. Viele Smartwatches sind standardmäßig im *Always On Modus*, d.h. das Display ist stets aktiv, wenn auch gedimmt. Hebt der Nutzer den Arm, um auf die Uhr zu schauen, erkennen entsprechende Sensoren diese Geste und schalten das Display gänzlich aktiv. In diesem Modus, dem oben genannten *aktiven Modus*, besitzt Android Wear eine Spracherkennung, die sich freihändig nutzen lässt (vgl. Abb. 2.10). Durch die Schlüsselwortfolge „Okay Google“ schaltet der Nutzer die Spracheingabe aktiv und kann weitere Befehle nennen, auf die das Gerät reagieren soll. Zum jetzigen Zeitpunkt bietet Google neun eingebaute Sprachbefehle an, die eine Anwendung abonnieren kann. Wird einer dieser Befehle aufgerufen (bspw. „Okay Google, stelle den Wecker auf 16 Uhr“), kann die eigene Applikation auf der Uhr gestartet werden und die gegebenen Informationen verarbeiten. Zwar gibt es auch einen Befehl, um eine App direkt über die Spracheingabe zu starten, beliebige Sprachbefehle kann man für den aktiven Modus derzeit allerdings nicht setzen. Passt also keines der vorgegebenen Sprachbefehle, muss zunächst immer erst die eigene Anwendung gestartet werden, um von dieser ausgehend weitere Sprachbefehle oder andere Eingaben annehmen zu können. Wenn die Spracheingabe im aktiven Modus auf keine der vorhandenen Befehlsmuster passt, wird standardmäßig eine Vorschau von Googles Web-Suche geladen. Die eigentlichen Ergebnisse können per Knopfdruck auf dem gekoppelten Smartphone geöffnet werden.

2.5.3 Wearable Apps

Wie bereits in der Beschreibung der Spracheingabe erwähnt, bietet sich Android Wear auch als eine Plattform für native Anwendungen an. Diese sind zwar keine nativen Android Anwendungen, bieten allerdings in den Grenzen der meist rudimentäreren Sensorik und Aktuatorik einen ähnlich großen Funktionsumfang. Unter anderem gibt es Zugriff auf eben diese Sensoren und Aktoren, Dienste (*Services*) die im Hintergrund laufen können und vollwertige Anwendungen (*Activities*), bei denen das Aussehen ähnlich wie bei Android beliebig angepasst werden kann.

Für Android Wear gibt es keinen eigenen App-Store, in dem Applikationen heruntergeladen werden können. Vielmehr kann jede Android Applikation direkt ihre passende Android Wear App mitbringen. Diese wird bei Installation der Android App automatisch auf das Wearable aufgespielt.
In der weiteren Entwicklung von Android Wear ist damit zu rechnen, dass es beispielsweise durch eigene Bluetooth Host Funktionalität und GPS-Unterstützung ein Stück weit unabhängiger von den gekoppelten Smartphones wird. Denkbar sind laut Google dabei zum Beispiel Musik auf einem Bluetooth Headset direkt von dem Wearable aus abzuspielen, oder etwa Fitnessaktivitäten mit im Wearable integriertem GPS und mit einem über Bluetooth verbundenem Pulsmesser aufzuzeichnen. Das Smartphone könnte für gewisse Aktivitäten sogar verzichtbar werden.

2.5.4 Datensynchronisation

Da die Schnittstellen der mit Android Wear ausgerüsteten Wearables sich meist auf Bluetooth beschränken, ist es für Wearable Apps essentiell, mit dem Smartphone kommunizieren zu können. Alle Daten, die von dritten Servern übertragen werden, müssen zunächst über das Smartphone laufen (vgl. Abb. 2.11). Das Android Wear SDK (Software Development Kit) bietet deshalb verschiedene Wege, über die das Smartphone und das Wearable möglichst einfach kommunizieren können.

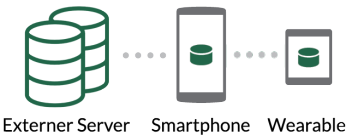


Abb. 2.11: Daten Beschaffung Android Wear

Es gibt die sog. *Messages*, *Data Items* und *Assets*. Letztere sind für große Datenmengen wie Bilder gedacht und daher für diese Arbeit nicht weiter interessant. Die Messages sind Nachrichten im Sinne eines klassischen Request-Response Prinzips. Diese Nachrichten können bidirektional zwischen Wearable und Smartphone ausgetauscht werden. Das Senden ist blockierend und liefert dafür dafür im Fehlerfall eine entsprechende Meldung. Weiterhin gibt es die oben genannten *Data Items*. Sie sind als verteilter Key-Value Speicher zu verstehen, der automatisch über verschiedene Geräte synchronisiert wird.
Beide Methoden erlauben die Registrierung von speziellen Diensten zum Empfang der Nachrichten bzw. Benachrichtigungen über Änderungen an den Daten.

2.5.5 Verwendung von Android Wear

Auf die genaue Verwendung von Android Wear im Rahmen dieser Thesis wird noch im späteren Verlauf dieser Arbeit detailliert eingegangen. Die besondere Eignung soll allerdings bereits hier herausgestellt und durch folgenden Auszug verdeutlicht werden:

> *The best experience on a wearable device is when the right content is there just when the user needs it. You can figure out when to show your cards with sensors, or events happening in the cloud. For the cases where it's impossible to know when the user needs your app, you can rely on a voice action or touch.*
>
> – Google Inc. *Design Principles for Android Wear* Google Inc. (2014c)

Dieser Auszug aus den Android Wear *Design Principles* beschreibt im Grunde zwei mögliche Interaktionsmuster mit den Wearables. Letzteres Muster beschreibt ein sehr verbreitetes und wenig überraschendes Verhalten. Der Benutzer startet die Applikation in dem Moment, in dem er sie benötigt. Das zuerst genannte Muster ist sehr viel interessanter, da dem Anwender die Inhalte, welche er benötigt, in dem entsprechenden Moment automatisch angezeigt werden sollen. Auf diese Weise wird die Interaktionsnotwendigkeit für den Nutzer so gering wie möglich gehalten.
Beide Konzepte zusammen bieten dem Nutzer stets eine passende Möglichkeit mit dem zu entwickelnden System zu interagieren. Im Rahmen der geplanten Prototypen-Entwicklung gilt: Je geringer die aktive Kommunikation auf Nutzerseite mit dem System ausfällt, desto erfolgreicher ist der intelligente Ansatz der Software umgesetzt.

3 Analyse

Dieses Kapitel befasst sich mit der Analyse der in der Einleitung genannten Problemstellung. Es sollen die Anforderungen an das zu entwickelnde System benannt werden und auch die Rahmenbedingungen für die Umgebung des Systems definiert werden. Schließlich wird das Vorhaben auf seine Umsetzbarkeit hin überprüft und auf vergleichbare Arbeiten hingewiesen.

3.1 Zielsetzung und Relevanz

Die eingangs genannte Problematik soll im Folgenden in eine Zielsetzung für die Entwicklung eines Prototyps überführt werden. Grundlegend besteht das Ziel darin, ein Konzept zu entwerfen, nach dem ein lernendes Smart Home System auf sinnvolle Weise mit seinen Nutzern oder Bewohnern kommunizieren kann.
Dabei sollen unterschiedliche Alltagsszenarien berücksichtigt werden und als Grundlage für mehrere Teilkonzepte dienen. Das System soll währenddessen zunehmend in der Lage sein, aus Benutzergewohnheiten zu lernen und die Kommunikation mit dem Nutzer zu optimieren. Wie in der Einleitung bereits deutlich gemacht, wird das eigentliche Lernverfahren in dieser Arbeit nur oberflächlich behandelt. Der Schwerpunkt liegt auf Interaktion und einer flexiblen Softwarearchitektur.

In den Forschungszweigen des Smart Home und Ambient Intelligence (Kapitel 2.2) gibt es bereits viele Studien über mögliche Interaktionsweisen. Einige dieser Studien thematisieren weniger handliche Geräte, wie interaktive Möbel, Tische oder Tablets. Die Interaktion mit dem System stellt hier stets einen gewissen physischen Aufwand für den Nutzer dar. Andere Konzepte versuchen die Notwendigkeiten von sichtbaren Bedienfeldern gänzlich aufzuheben. Es gibt Ansätze, nach denen Kamerabilder oder Inertialsensor genutzt werden, um Gesten von Nutzern zu analysieren (s. Verwandte Arbeiten, Kapi-

tel 3.4). Viele dieser Konzepte haben zudem gemeinsam, dass sie den Nutzer räumlich einschränken. Befindet er sich außerhalb des Sensorbereichs, oder fernab von Kontrolleinheiten, ist ihm die Interaktion nicht möglich. Weiterhin lässt sich sagen, dass, während ein Teil der Ansätze nicht mehr ganz zeitgemäß wirkt, der andere Teil sich als ausgesprochen futuristisch präsentiert und bislang keinen oder lediglich geringen Einzug in reale Systeme halten konnte.
Dies wird deutlich, wenn man den Markt der derzeit kommerziell vertriebenen Smart Home Systeme betrachtet. Systeme, wie die des Stromanbieters RWE, des Routerherstellers AVM, oder das QIVICON-System der Telekom versuchen sich zwar gegenseitig mit solider Aktorik zu übertrumpfen, können aber zur Steuerung dieser nichts Innovativeres als ein Webinterface oder eine Smartphone-App vorweisen.

An diesem Punkt soll das hier beschriebene Konzept ansetzen. Es folgt dem aktuellen Trend der Wearables und setzt diese insgesamt bereits sehr verbreitete und damit zukunftsträchtige Technologie in den Kontext der *Ambient Intelligence*. Gemäß der im Kapitel 2.2 erläuterten *Evolution des Computers* ist es dem derzeit letzten Evolutionsschritt zuzuordnen.
Auch die oben und in den nachfolgenden Abschnitten angeführten Konzepte sind zum Teil in der letzten Evolutionsstufe - dem *Disappearing computing* - einzuordnen, fordern aber meist teure Sensorinstallationen oder bedürfen weiterer Forschung (s. Kapitel 3.4). Wearables dagegen haben, wie die Zahlen aus Kapitel 2.1 (Abb. 2.1) zeigen, bereits einen stark wachsenden Markt und erlangen auf diese Weise an hohe Verbreitung und Akzeptanz der Nutzer.

Da ein Wearable sich aus seiner Definition heraus an den Körper des Nutzers anschmiegt und so auch einen erheblichen Einfluss auf dessen Alltag haben kann, gilt es den bereits mehrfach genannten Interkationsmustern der Wearables besondere Aufmerksamkeit zu widmen. Diese legen fest, wann und in welcher Form, sich das System bzw. das Client-Gerät an den Nutzer wenden kann, bzw. in welcher Form der Nutzer mit dem System interagieren kann. Die Anforderungen an diese Muster sollen im nächsten Abschnitt festgelegt werden.

3.2 Szenarien und Anforderungen

Die grundlegendste Anforderung an das zu entwickelnde System wurde bereits mehrfach genannt: Ein Nutzer soll durch ein Wearable in der Lage sein, mit einem lernenden Hausautomationsystem zu interagieren. Im Folgenden soll diese Anforderung mithilfe einiger beispielhafter Szenarien näher spezifiziert werden. Diese Szenarien sollen auch als Grundlage für die Entwicklung der Interaktionsmuster (s. Kapitel 4.3) gelten.

3.2.1 Szenarien

Es wurden drei grundlegende Szenarien analysiert, die im Rahmen der technischen Randbedingungen mustergültig für die Ermittlung der benötigten Interaktionsstrukturen sein sollen.

Szenario 1 - Passives System, aktiver Nutzer

> Der Nutzer möchte abends das Licht auf seiner Terrasse einschalten.

Um seinen Wunsch zu erfüllen, wird der Anwender in diesem herkömmlichen Szenario selbst aktiv und bedient sich beispielsweise eines Wandschalters, einer Tablet-App oder einer Smartphone-App. Der konventionelle Wandschalter setzt physische Anwesenheit des Nutzers voraus und stellt somit einen Bequemlichkeits- und ggf. auch Zeitnachteil dar. Er soll an dieser Stelle nicht weiter betrachtet werden.
Tablet und Smartphone bieten dem gegenüber bereits einen Mobilitätsvorteil, können aber weiterhin an einem anderen Ort als dem Benötigten liegen. Ein Wearable, wie beispielsweise eine Smartwatch, ist stets zur Hand. Diese fortwährend nutzbare, ubiquitäre Steuerung soll die erste Anforderung an ein modernes Hausautomationssystem sein.
Im Gegensatz zu einem Tablet bietet eine Smartwatch nur ein relativ kleines Display, weshalb viele Bedienelemente auf kleinem Raum dargestellt werden müssten. Eine weitere Anforderung an das Bedienkonzept ist somit eine Lösung für diese Problematik zu entwickeln. Der Nutzer soll schnell in der Lage sein, die richtigen Bedienelemente für die gewünschte Aktorik zu finden und zu nutzen.

Szenario 2 - Passiv vorschlagendes System

> Der Bewohner betritt die heimische Küche, möchte dort Licht haben, Radio hören und gerne einen Kaffee trinken.

Der Nutzer könnte diese Aktionen wie in *Szenario 1* einzeln und manuell veranlassen. Es handelt sich allerdings um ein Muster, das bisher durchaus öfter aufgetreten sein könnte. Hat das System bereits Informationen über den Nutzer sammeln können, lässt sich eine gewisse Wahrscheinlichkeit für den wiederholten Wunsch über die Ausführung der oben genannten Aktionen feststellen. Mit einer gewissen Wahrscheinlichkeit möchte der Nutzer aber auch etwas ganz Anderes oder möglicherweise gar keine Aktion.
Aus diesem Verhalten lässt sich eine weitere Anforderung ableiten: Das System macht dem Bewohner zum geeigneten Zeitpunkt Vorschläge über passende Aktionen. Da das System nicht sicher wissen kann, ob der Nutzer überhaupt handeln möchte, soll er nicht alarmiert werden. Folglich gibt es keine Benachrichtigung und keine autonome Handlung durch das Smart Home System. Erst wenn der Nutzer selbst eine Aktion ausführen möchte, sucht er die Interaktion über sein Wearable und kann schnell aus vorgeschlagenen Bedienelementen wählen.

Szenario 3 - Aktiv vorschlagendes System

Von dem unter *Szenario 2* beschriebenen Verhalten sind Ausnahmen denkbar, in denen sich der Nutzer ein proaktiv handelndes System wünschen könnte. Ein Beispiel:

> Der Bewohner kommt abends vom Einkaufen nach Hause, er trägt schwere Taschen und möchte gerne Licht im Eingangsbereich haben.

Diese, wie einige andere Situationen auch, lässt kaum Zweifel an dem vom Nutzer gewünschten Systemverhalten aufkommen. Er ist beschäftigt, nicht in der Lage oder einfach zu bequem, die Aktorik selbst zu kontrollieren. In jedem Fall wäre er froh darüber, wenn das System die Aktorik in seinem Sinne steuert.

Auch bei diesen bisher autonomsten Anforderungen, darf der Nutzer nicht entmündigt werden. Er soll stets die Kontrolle behalten und muss im Zweifel noch vor der Ausführung eingreifen können. Ein Konzept zur Adaption dieses Szenarios muss dementsprechend die Fragen beantworten, wie das System sich dem Nutzer mitteilt und wie dieser eingreifen kann. Ferner gilt es zu überlegen, welche Aktorik überhaupt für (teil-) auto-

nome Steuerung geeignet ist. Bei Küchengeräten, wie beispielweise einem Ofen, gäbe es erhebliche Sicherheitsbedenken, die zu berücksichtigen sind.

Sonstige Interaktionsarten

Denkbar sind natürlich diverse weitere Interaktionsarten mit entsprechenden Systemen, beispielsweise regelbasierte Systeme oder völlig autonom arbeitende Smart Home Systeme. Diese sind allerdings aus Sicht des Wearable-Interaktionskonzeptes nicht interessant und sollen an dieser Stelle nicht weiter betrachtet werden. Gleichwohl soll die entstandene Architektur dieser Arbeit durch ihre hohe Adaptionsfähigkeit mit mehr oder weniger Aufwand auch für solche Konzepte geeignet sein.

3.2.2 Weitere Anforderungen

Sicherheit und Datenschutz

Keinesfalls soll unerwähnt bleiben, dass es sich bei nahezu allen gesammelten Daten um höchst sensible und private Nutzerdaten handelt.
Grundvoraussetung für jedes System als Verwender oder Erzeuger solcher Daten ist der sichere und vertrauensvolle Umgang mit diesen. Ein großer Teil der Datenhaltung geschieht zwischen Sensornetzwerk und Kontrolleinheit des Smart Homes. Da die Betrachtung dieser Komponenten nicht Bestandteil dieser Arbeit ist, sollen weitere sicherheitskritische Anschauungen auf das Nötigste reduziert werden. Als Anforderung an das Wearable gilt allerdings, dass eben diese Daten betreffende Kommunikation nur mit dem Smart Home Server und nur verschlüsselt stattfinden soll.

3.3 Plattform und Rahmenbedingungen

Die zu entwickelnde Applikation setzt sowohl hardware- als auch softwaretechnisch einige Bedingungen voraus, welche nachfolgend ermittelt werden sollen.

Wie in der Einleitung und den vorausgehenden Abschnitten deutlich gemacht, beschränkt sich diese Arbeit auf Entwurf und Umsetzung der Wearable-Interface Komponenten bzw. deren Gegenstück auf Serverseite. Vorausgesetzt wird deshalb ein funktionierendes

Smart Environment in Form einer Hausautomationssteuerung. Ebenso bedingt diese Arbeit ein lernendes System, das als Teil des Smart Home Systems Benutzergewohnheiten klassifiziert.

3.3.1 Living Place Hamburg

Ein geeignetes Beispiel für eine solche Umgebung stellt das *Living Place* in Hamburg dar.
Das Living Place ist eine, an der Hochschule für Angewandte Wissenschaften im Jahre 2009 ins Leben gerufene Simulation einer Smart Home Umgebung. Es stellt überdies ein interdisziplinäres Kooperationsprojekt vieler Studierender dar und erforscht Aspekte über unser zukünftiges Leben in technologisierten Wohnumgebungen.
Konkret handelt es sich bei dem Labor um eine realitätsnahe und vollständig eingerichtete $140m^2$ Wohnung, mit funktionierender Sanitärtechnik und einer großen Anzahl an Sensoren und elektronischer Aktoren (vgl. Luck u. a., 2010).

Abb. 3.1: Logo des Living Place Hamburg

Um die Akzeptanz der Nutzer gegenüber neuen Techniken zu analysieren, ist die Wohnung auch als Usability-Labor ausgelegt. Es gibt einen Kontrollraum und einige Arbeitsräume, welche sich räumlich von der eigentlichen Wohnsimulation abgrenzen. Der Aufbau lässt sich Abbildung 3.2 entnehmen.

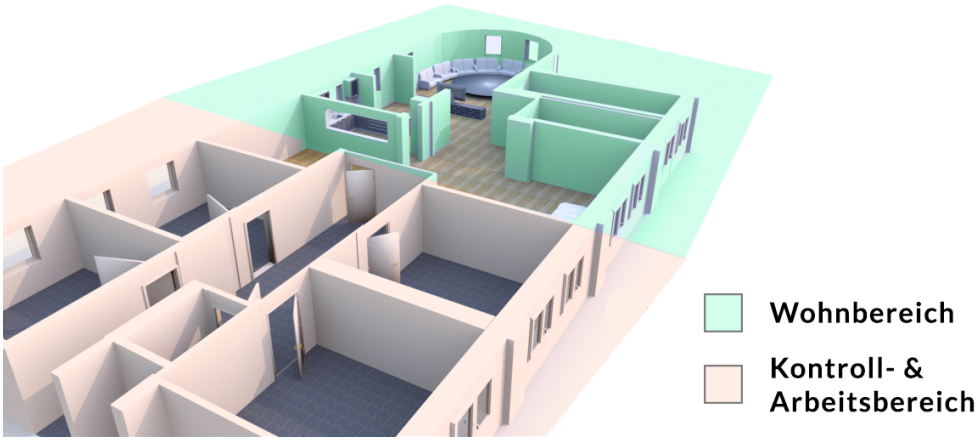


Abb. 3.2: Aufbau des Living Place Hamburg

Im Rahmen des *Living Place* gab es bereits zahlreiche Studienarbeiten, die zu einem großen Fundus an Wissen über Smart-Home-Interaktion angewachsen sind.
Einige für diese Thesis relevante Arbeiten werden im kommenden Abschnitt (Verwandte Arbeiten, Kapitel 3.4) betrachtet.

Darüber hinaus soll das Living Place auch architektonisch als ein Bezugsbeispiel für diese Arbeit gelten. Die gut dokumentierte Architektur und das Vorhandensein der benötigten Komponenten machen es zu einem realistischen Einsatzgebiet für den entwickelten Prototypen. Der grundlegende Aufbau wird im Folgenden zusammengefasst.

Sensorik und Kontextinterpretation

Hauptaufgabe eines zentralen Teils des Living Place ist es, mit Hilfe von Sensorik eine Kontextinterpretation und deren Austausch zwischen Interessenten zu gewährleisten. Dieser Austausch geschieht durch eine ausgefeilte Softwarearchitektur, welche im kommenden Abschnitt beschrieben wird (vgl. Ellenberg, 2011).
Für die Kontextinterpretation werden nach Ellenberg Sensoren in Form von beispielsweise Kameras, Drucksensoren oder Kontaktschaltern genutzt.

Softwarearchitektur des Living Place

Lange Zeit wurde im Living Place primär auf die sog. *Blackboard-Architektur* gesetzt (vgl. Ellenberg, 2011). Diese gilt aber mittlerweile als überholt und soll zunehmend durch eine sog. *Multi-Agent Architektur* ersetzt werden (vgl. Eichler, 2014). Da Arbeiten bezüglich dieser Thematik erst im Verlauf dieser Arbeit abgeschlossen wurden, wird für diese Thesis vor allem die solide und althergebrachte Blackboard-Architektur eine Referenz sein. Nachfolgend wird diese erläutert. Anschließend folgt ein Ausblick auf die zukünftige Architektur.

Blackboard Architektur

Die Blackboard-Architektur wurde ursprünglich von Dr. Lee Erman im Jahr 1980 das erste Mal beschrieben (vgl. Erman u. a., 1980)).
Sie kommt zur Lösung von Problemen zum Einsatz, für die es keine eindeutige Lösungsstrategie gibt (vgl. Onnebrink u. Kuchen, 2004). Ähnlich dem Teile- und Herrsche

Paradigma, wird die Problemstellung in Teilprobleme zerlegt. Verschiedene Teilnehmer können sich Teilprobleme oder Teillösungen herausgreifen bzw. abonnieren, diese weiterverarbeiten und anschließend selber wieder publizieren.

Das Blackboard stellt demzufolge eine gemeinsame Wissensbasis dar, welche Iteration für Iteration aktualisiert wird. Es dient als Sammlung von Problemen, Teillösungen, Vorschlägen und anderen beigetragenen Informationen.

Im Living Place kommt die Blackboard Architektur insbesondere zur Interpretation und Umwandlung von Sensordaten zum Einsatz. Diese Rohdaten werden Schritt für Schritt zu höherwertigen Kontextinformationen verarbeitet. Beispielsweise können reine Zahlenwerte von Abstandsensoren mit Kamera-Daten verknüpft werden, um schließlich eine Positionsbestimmung der Person im Raum zu ermöglichen.

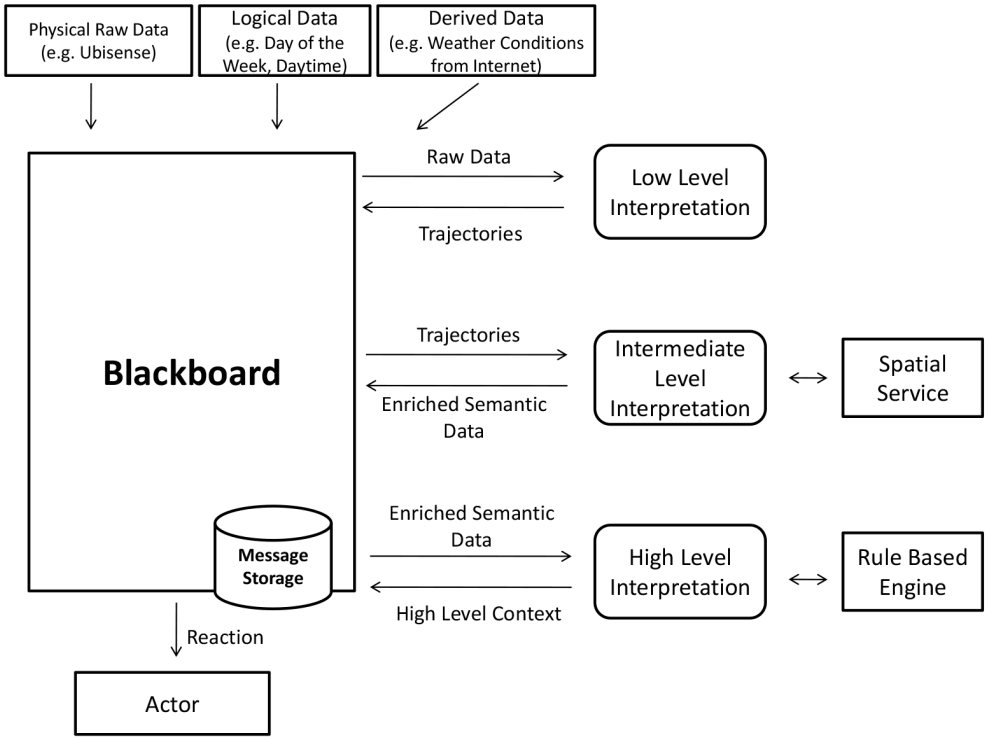


Abb. 3.3: Softwarearchitektur des Living Place (Ellenberg u. a., 2011)

Die Abbildung 3.3 zeigt das architektonische Schema des Living Place. Auffallend ist zunächst die große Komponente *Blackboard* in der Mitte der Zeichnung. Es wird gespeist mit logischen, physischen und anderen Kontextdaten. Auf dieser Basis finden aufeinander aufbauende Interpretationen und Auswertungen statt. Alle abgeleiteten In-

formationen werden wieder auf dem Blackboard veröffentlicht.
Umgesetzt wird das Blackboard durch ein Messaging-System, das auf das sog. *publish–subscribe* Architekturmuster baut. Nachrichten werden demnach nicht direkt an ausgewählte Empfänger gesendet, sondern über einen *Message Broker* allen Abonnenten (*subscribers*) zur Verfügung gestellt.
Bei einem einheitlichen Nachrichtenformat ist die Kopplung der Komponenten untereinander lose und neue Komponenten können ohne weitere Anpassungen mit den bestehenden Komponenten zusammenarbeiten.

Multi-Agenten Architektur

Die *Multi-Agenten Architektur* oder auch *Agentenbasierte Middleware* nach Tobias Eichler baut im Grunde ebenfalls auf dem Blackboard Architekturmuster auf. Eichler sah den Anlass seiner Entwicklung unter anderem darin, dass viele Teilnehmer des alten Systems zunächst aufwendig ihre eigene Implementation für die vorhandene Messaging-Schnittstelle entwickeln mussten.
Kernkonzept des neuen Systems ist es daher, eine einheitliche Middleware zur Kommunikation der verschiedenen Teilnehmer (*Agenten*) zur Verfügung zu stellen (vgl Abb. 3.4). Eine Implementation verschiedener Kommunikationsprotokolle entfällt und auch eine einheitliche Fehlerbehandlung ist möglich.

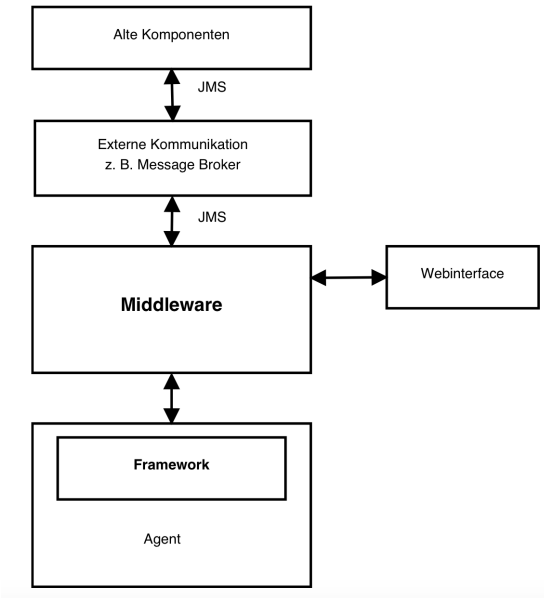


Abb. 3.4: Mutlti Agent Architektur nach Eichler (vgl. Eichler, 2014)

Zentrales Architekturmerkmal ist die Gruppenkommunikation, bei der Agenten verschiedenen Gruppen zugeordnet sind, an die Nachrichten verschickt werden. Auch alte Komponenten, die weiterhin auf das alte Konzept setzen, können wiederum über spezielle Agenten an das neue System angebunden werden. Beide Architekturen können so - wenigstens übergangsweise - parallel existieren.
Die Middleware selbst verfügt weiterhin über Logging-Tools und ein Webinterface zur effizienten Verwaltung, Überwachung und Optimierung dieser.

3.4 Verwandte Arbeiten

Dieser Abschnitt thematisiert verschiedene für diese Thesis relevante Arbeiten. Sie behandeln Interaktionsformen mit Smart Home Umgebungen und sollen als Teil dieses Kapitels einen kompakten Überblick über verwandte Themengebiete liefern.

3.4.1 Gestenbasierte Steuerung

Es gibt eine große Anzahl an Arbeiten, welche sich der durch Gesten unterstützten Steuerung von Smart Home Umgebungen widmet. Aus Rahman u. a. (2009) und Neßelrath u. a. (2011), wie auch vielen anderen Arbeiten, geht dabei ein hohes Potential für diese Interaktionsform hervor. Die nötigen Daten werden mithilfe von Infrarotkameras oder Hand-Eingabegeräten, wie der Wiimote, gesammelt. Auch die Smart Home Fernbedienung *GeeAir* (s. Abb. 3.5) setzt auf den Wiimote-Controller (vgl. Pan u. a., 2010). Dieser enthält unter anderem einen Inertialsensor zur Bewegungsdetektion. Ein Intertialsensor ist ein Sensor zur kombinierten Messung von Beschleunigungen und Drehraten.

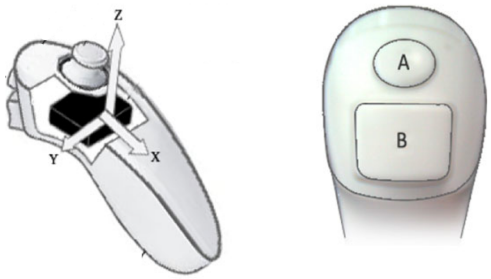
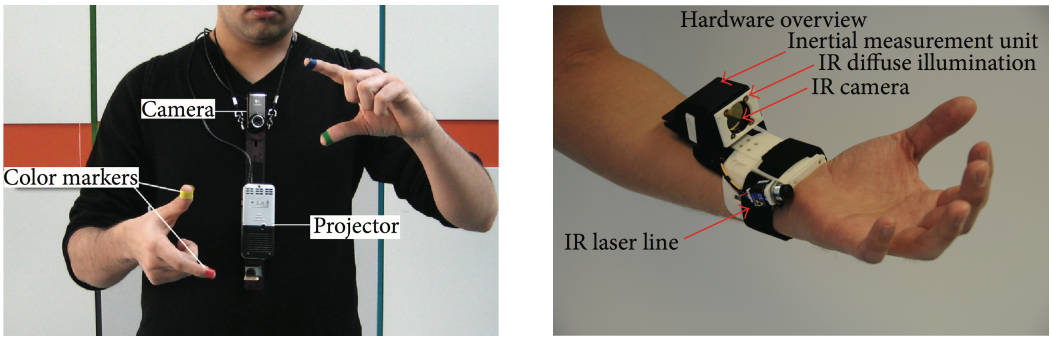


Abb. 3.5: Skizze der GeeAir (WiiMote) (vgl. Pan u. a., 2010)



(a) „The SixSense system“ (Mistry u. Maes, 2009) (b) „Digits“ (Kim u. a., 2012)

Abb. 3.6: Tragbare Kameras zur Gestenerkennung (aus LaViola, 2013)

In seiner Arbeit mit dem Titel „3D Gestural Interaction: The State of the Field“ untersucht Joseph LaViola zusammenfassend den Forschungsstand der dreidimensionalen Gestenerkennung (vgl. LaViola, 2013). Dort werden auch Fremdansätze mit tragbaren Kameras vorgestellt, so z.B. Kim u. a. (2012) und Mistry u. Maes (2009) (s. Abb. 3.6a u. 3.6b).

Bei Mistry u. Maes wird auf eine um den Hals gehängte Kamera gesetzt. Das Gerät beinhaltet außerdem einen eigenen Projektor zur Darstellung von Inhalten. Zur korrekten Erkennung von Gesten ist es darauf angewiesen, dass verschiedenfarbige Bänder an jedem Finger beider Hände getragen werden.
Der Ansatz von Kim u. a. setzt auf eine am Handgelenk getragene Kamera zur Interpretation von Finger-Gesten. Es verwendet eine Infrarotkamera und ist ebenfalls mit einem Inertialsensor ausgestattet. Durch die Kombinationen der Daten lässt sich auf Gesten des gesamten Armes zurückschließen.

Fazit zur gestenbasierten Steuerung

Der Kamera-Ansatz ist bei fixen Installationen meist blickwinkelabängig und bedarf unter realen Bedingungen großer Installationen, um möglichst den gesamten Wohnraum zu erfassen. Einige Ansätze tun sich darüber hinaus schwer mit schlechten Lichtverhältnissen.
Ansätze mit der *Wiimote* oder tragbaren Kameras sind im Grunde stabiler in den Ergebnissen, haben aber den erheblichen Nachteil, dass man das Eingabegerät im Alltag nicht permanent mit sich führt bzw. die derzeitigen Prototypen wenig alltagskonform sind.

Hinzu kommen spezielle Einschränkungen, wie die benötigten Farbmarker bei Mistry u. Maes oder eine geringe, prognostizierte Akkulaufzeit (vgl. LaViola, 2013).

3.4.2 Sprachsteuerung

Einen anderen Ansatz verfolgt Witt in seiner Masterthesis von 2011 (vgl. Witt, 2011): In dieser Arbeit bediente er sich der Spracherkennung, um Befehle des Bewohners entgegen zu nehmen. Gesprochene Befehle werden dabei zunächst vorverarbeitet, anschließend werden Merkmale extrahiert, die wiederum einem stochastischen Verfahren unterzogen werden. Am Ende des Prozesses wird die erkannte Wortsequenz zur Weiterverarbeitung freigegeben.

Neben der „nicht optimalen Erkennungsperformanz“, leidet dieser durchaus interessante Ansatz allerdings ebenfalls unter der Infrastrukturnotwendigkeit und sozialen Aspekten (vgl. Witt, 2011, S.96). Die benötigten Mikrofone könnten flexibler eingesetzt werden, als Kameras, bedeuten aber auch einen gewissen zusätzlichen Ausrüstungsaufwand.

3.4.3 Konventionelle Steuerung mit mobilen Geräten

Kolbaja entschied sich in seiner Bachelorthesis von 2012 dafür, den konventionellen Ansatz für Smart Home Steuerungen einer Neuauflage zu unterziehen.

Es gibt bereits einen großen Markt kommerzieller Smart Home Systeme samt mobiler Steuerapplikationen. Der Fokus seiner Arbeit liegt deshalb darauf, einen auf Android basierten und durch Plugins erweiterbaren Smart-Home-Controller zu entwickeln. Es gelang ihm, eine im Kern erweiterbare Applikation zu entwerfen, welche aber, wie er selber im Fazit einräumt, nicht außerhalb des Testlabors *Living Place* verwendet werden kann, da es zu sehr auf dieses abgestimmt ist.

Der Ansatz zeigt eine solide Steuerapplikation, wie sie schon heute kommerziell vertrieben wird. Es mangelt ihr allerdings an Übertragbarkeit auf andere Systeme.
Die prototypische Nutzung auf Basis von Tablets ergänzt ein Smart Home sinnvoll. Für sich allein genommen, ist dieses Interaktionskonzept aber in der Praxis nicht ausreichend. Der normale Nutzer hat nicht unzählige Smartphones oder gar Tablets herumliegen, um das System aus beliebigen Situationen heraus bedienen zu können.
Auch muss ein Tablet meist erst entsperrt, die entsprechende Applikation gestartet und

die gewünschte Funktionalität gefunden werden. Dies kostet in der Summe viel Zeit und kann zu Frustration des Anwenders führen.

3.5 Fazit

In diesem Kapitel wurden grundlegende Szenarien zur Interaktion mit einem Smart Home vorgestellt. Es wurden Anforderungen an eine wünschenswerte Steuerung gestellt, welche Grundlage für das folgende Kapitel Entwurf sein sollen.
Es wurden weiterhin Zielsetzung und Relevanz für ein solches System erörtert und gezeigt, warum es neuer, mobiler Konzepte bedarf. Außerdem wurde mit dem *Living Place* ein musterhaftes Smart Home präsentiert, in dessen Umgebung das hier entwickelte System arbeiten könnte.

Schließlich wurden verwandte Arbeiten betrachtet und unterschiedliche Interaktionsszenarien mit Augenmerk auf kritikwürdige Punkte untersucht. Diesen Konzepten soll dabei in keinster Weise ihre Daseinsberechtigung abgesprochen werden. Sie sind größtenteils sehr interessant und auf Dauer sicher auch zukunftsweisend. Es konnte allerdings herausgestellt werden, dass sie unter Gesichtspunkten wie Finanzierbarkeit, Ausrüstungsaufwand, Erkennungsrate und Alltagstauglichkeit zum Teil der Nachbesserung bedürfen.

Als Kernanforderung lässt sich zusammenfassen: Es gilt ein Konzept zu entwickeln, dass den drei beschriebenen Szenarien auf geeignete Weise begegnet. Alltagskonformität des Steuerungsgerätes ist dabei ebenso wichtig, wie geringer Aufwand an zusätzlichen Hardware-Installationen. Das dabei entwickelte System soll sich mit geringen Aufwendungen in ein bestehendes Smart Home System integrieren lassen und bedingt deshalb eine adaptionsfähige Softwarearchitektur.

Die Analyse verdeutlichte, welcher Vorteil von einer Wearable zentrierten Smart Home Steuerung ausgehen kann. Dieser Ansatz ist im Vergleich meist schlanker und kostengünstiger. Vor allem aber ist er pragmatischer, wie Wachstumszahlen und untersuchte Szenarien zeigen.

4 Entwurf

Auf der Basis der vorausgegangenen Kapitel werden in diesem Kapitel ein Konzept und eine geeignete Architektur vorgestellt, die den genannten Anforderungen gerecht werden sollen. Das Kapitel beschreibt außerdem die Entwicklung des Prototyps und dessen Anwendungsszenarien.

4.1 Konzept

Das nachfolgende Konzept hat zum Ziel, eine Architektur zur Interaktion mit Smart Home Systemen zu beschreiben. Dabei sollen, wie vielfach erwähnt, Wearables zum Einsatz kommen.
Der derzeit stärkste Repräsentant aus der Familie der Wearables ist die Smartwatch. Viele Menschen sind es gewohnt, Uhren zu tragen. Nach einer Umfrage des *Spiegel Manager Magazins* aus dem Jahr 2011 sind rund 77 % der Deutschen zwischen 14 und 69 Jahren wenigstens gelegentliche Armbanduhrenträger (vgl. Spiegel, 2011). Smartwatches sind optisch und haptisch sehr ähnlich. Sie fallen kaum auf und lassen sich bestens in den Alltag integrieren. Das hohe Potential und die Sozialverträglichkeit dieser Geräte wurde bereits ausführlich im Kapitel 2.1 (Wearables) thematisiert. Die Smartwatch soll deshalb als Grundlage für den entwickelten Prototyp gelten. Als Betriebssystem kommt AndroidWear zum Einsatz. Das LivingPlace Hamburg (vgl. Kap. 3.3.1) soll mit seiner bereits vorgestellten Blackboard-Architektur in weiten Teilen als mustergültige Referenz einer Smart Home Umgebung gelten.
Das entwickelte Gesamtsystem wird im Folgenden als *HAWS* (Home Automation Wear Control System) bezeichnet. Es besteht aus einem Server, der die Kommunikation zum Smart Home System übernimmt, einer Android Wear Applikation, in der die eigentli-

che Steuerung stattfindet, und einer Android Applikation auf einem Mobiltelefon, die zwischen den beiden anderen Komponenten vermittelt.

Die Umsetzung des in diesem Kapitel entwickelten Systems soll den Bewohner einer Smart Home Umgebung dazu befähigen, nahezu alle im Alltag durchzuführenden Aktorsteuerungen über eine ständig mit sich geführte Smartwatch abzuwickeln. Grundlage für dessen Verwendung stellen die Interaktionsmuster dar, welche im Abschnitt 4.3 beschrieben werden. Generell gilt, die Interaktion möglichst weit zu vereinfachen und dem Nutzer entsprechend entgegen zu kommen. Intelligente Sortierung, Kategorisierung und überlegte Vorschläge mithilfe eines lernenden Systems sollen dabei Verwendung finden.

Der anschließende Abschnitt *Hardware* benennt Notwendigkeiten und Rahmenbedingungen für die Entwicklung des HAW Systems. Es folgt der konzeptuell wichtige Abschnitt *Interaktionsmuster*. Im Anschluss werden die Softwarearchitektur und die technische Kommunikation zwischen den einzelnen Komponenten beschrieben. Schließlich wird das Lernen des Systems kurz thematisiert (Abschnitt 4.5). Es soll im Rahmen dieser Arbeit jedoch nicht vertieft werden.

4.2 Hardware

An dieser Stelle soll die für den Prototyp benutzte Hardware spezifiziert werden. Konkret wird diese dabei durch die *LG G Watch* repräsentiert. Die LG G Watch ist eine auf Android Wear basierende Smartwatch des südkoreanischen Unternehmens *LG Electronics*. Sie weist ein 1,65 Zoll großes IPS Display bei einer Auflösung von 280x280 Pixel auf. In ihr arbeitet ein 1,2 Ghz starker Quadcore Prozessor mit 512 MB RAM. Weiterhin ist sie wasserdicht, verfügt über ein Mikrofon zur Sprachsteuerung, hat einen 400 mAh großen Akku und kommuniziert über Bluetooth 4.0 mit dem gekoppelten Smartphone (vgl. LG Electronics, 2014).

Abb. 4.1: LG G Watch (LG Electronics, 2014)

Die Akkuleistung ist ausreichend, um bei moderater Nutzung gut über einen Tag zu kommen. Die Uhr verfügt dabei über einen im Grundlagen-Kapitel beschrieben Always-On Mode, durch den das Display - wenn auch gedimmt - permanent aktiv ist. Hebt der

Nutzer seinen Arm auf bestimmte Weise, wird die Uhr gänzlich aktiv.
Die HAWS App wird für jedes aktuelle Android Wear Gerät lauffähig sein. Als Smartphone wird ein HTC One (M7) genutzt, welches mit Android 4.4.2 bestückt ist. Das spezielle Gerät hat aber faktisch keine Relevanz für das *HAW System*, da jedes Android Smartphone mit einer Android Version ≥ 4.3 geeignet ist. Vorausgesetzt werden hardwareseitig lediglich WLAN- und Bluetooth-Konnektivität.

Was die Smart Home Infrastruktur betrifft, so gilt das vorgestellte Living Place als geeignete Referenz für die folgenden Abschnitte. Der dort beschriebene HAWS Prototyp soll aber nicht allein für diesen Ort entwickelt werden, denn es handelt sich bei dieser Arbeit um einen generischen Ansatz.

4.3 Interaktionsmuster

Dieser Abschnitt beschreibt die notwendigen Interaktionsmuster, welche sich aus den Anforderungsszenarien des Kapitels Analyse ergeben. Ein Interaktionsmuster ist dabei ein Verhaltensmuster, das bei definiertem Verhalten des Nutzers das erwartete Kommunikationsverhalten des Systems beschreibt.
Nachfolgend werden dazu zunächst die aus dem Kapitel Analyse referenzierten Szenarien ins Gedächtnis gerufen und dann ein Lösungsansatz präsentiert. Dabei sollen auch die Integration in das System und technische Besonderheiten betrachtet werden.

4.3.1 „Klassische“ Steuerung

> **Passives System, aktiver Nutzer:**
> Der Nutzer möchte abends das Licht auf seiner Terrasse einschalten.

Das erste beschriebene Szenario stellt einen typischen Anwendungsfall dar. Er erfordert eine klassische Steuerung im Sinne einer Fernbedienung. Der Nutzer erwartet keine proaktiven Handlungen des Smart Home. Er wird selbst aktiv und steuert gewünschte Aktorik nach Wunsch. Hierbei behält er permanent die volle Kontrolle.
Dieses Szenario ist auf den ersten Blick wenig intelligent, aber unverzichtbar. Es steht außer Frage, dass derzeit kein lernendes System das Verhalten des Nutzers vollständig abbilden und vorhersagen kann. Der Nutzer soll deshalb selbst bei *smarten* Systemen stets auf manuelle Bedienung zurückgreifen können. Seine Befehle sollen dabei im Zwei-

fel gegenüber solchen priorisiert werden, die vom System selbst kommen. Weiterhin sei im Hinblick auf nicht funktionale Anforderungen wie die Benutzbarkeit erwähnt, dass die Einschränkungen von Wearables zu Optimierungsbedarf bei der Darstellung einer solchen Steuerung führen. Die Steuerung lässt sich nicht ohne weiteres auf einer Smartwatch wie der LG G Watch realisieren, da das Display räumlich sehr eingeschränkt ist. Der Nutzer soll mit möglichst wenig Interaktion zur gewünschten Aktor-Steuerung finden. Dazu notwendig ist eine intelligente Kategorisierung der gesamten Aktorik. Diese Thematik soll allerdings später erneut aufgegriffen werden.

Systemintegration

Zunächst gilt es, die viel grundlegendere Frage zu klären, wie der Nutzer überhaupt zur eigentlichen Steuerung gelangt. Android Wear bietet standardmäßig die Möglichkeiten, native Apps direkt zu starten oder eine sog. Benachrichtigungkarte seiner Applikation einzublenden.
Zum Starten einer App bieten sich Touchscreen und Sprachsteuerung an. Über den Touchscreen gilt es allerdings durch diverse Untermenüs und lange Listen zu navigieren, bis man die gewünschte Applikation starten kann. Abbildung 4.2 zeigt die dabei notwendigen Schritte. Die blauen Kreise symbolisieren Benutzeraktionen. Sie werden im Anhang unter 6.2 genauer erklärt. Im Schnitt benötigt selbst ein erfahrener Anwender etwa sechs bis zehn Sekunden zwischen dem erstem und dem letztem Schritt - die Startzeit der Anwendung und darin verbrachte Zeit noch nicht eingerechnet.

Abb. 4.2: Schritte zum Starten einer Android-Wear App über den Touchscreen

Ein ähnliches Bild zeichnet sich beim Starten einer Anwendung über die integrierte Sprachsteuerung ab: Vom „Okay Google“ bis zum Start verschiedener Anwendungen vergehen in mehreren Versuchen etwa fünf bis acht Sekunden - sofern die gewünschte

Applikation überhaupt erkannt wurde. Die notwendigen Schritte lassen sich der Abbildung 4.3 entnehmen. Diese relativ hohe Latenz hängt unter anderem damit zusammen,

Abb. 4.3: Schritte zum Starten einer Android-Wear App durch Spracheingabe

dass die gesamte Spracheingabe online über Googles eigene Server ausgewertet wird. Wie bereits im Kapitel Grundlagen beschrieben, lassen sich eigene Sprachbefehle an dieser Stelle auch nicht integrieren. Es bedarf zunächst immer erst des Startens der eigentlichen Anwendung.

Die oben angesprochenen Benachrichtigungskarten können, wie in Abbildung 4.4 gezeigt, beim Blick auf die Uhr bereits mit einem Klick zum Start der Applikation genutzt werden.

Die Karten bieten im Übrigen auch die Möglichkeit, Steuerelemente in die Benachrichtigung zu integrieren. So kann beispielsweise die Musikwiedergabe des gekoppelten Smartphones darüber gesteuert werden.
Ist die Benachrichtigungskarte also ein geeigneter Platz zum Start der HAWS Applikation? Nur bedingt, da sie dauerhaft fixiert werden müsste. Um lästiges Scrollen durch viele Karten zu vermeiden, müsste die eigene Karte stets höher priorisiert sein, als alle anderen Karten. Technisch ist dies möglich, aber man würde die Idee hinter den Karten boykottieren und die Smartwatch in ihren Funktionen, abseits der Hausautomation, beschneiden.

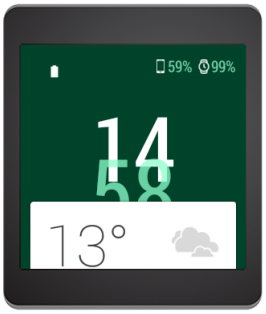


Abb. 4.4: Benachrichtigung in Android Wear überlagert Uhrzeit

Die von Google vorgesehenen Anwendungsszenarien bieten also kein optimales Modell, das zum Konzept einer schnell verfügbaren Steuerung passen würde.

Glücklicherweise basiert Android Wear auf Android und bietet deshalb auch darüber hinausgehende Möglichkeiten an. So kann zum Beispiel systemweit auf Touch-Gesten reagiert werden und auch die eigentliche Applikation lässt sich im Vorfeld bereits initialisieren. In der Folge kann wertvolle Ladezeit eingespart werden.
Für die HAWS Applikation wurden diese Konzepte zusammengeführt: Das Starten der Anwendung wird über Gesten in den sogenannten „aktiven Ecken“ realisiert. Streicht der Nutzer aus einer vorher definierten Ecke über den Bildschirm, so öffnet er damit gleichzeitig die Applikation. Abbildung 4.5a zeigt, welche Bereiche für diese Touch-Gesten denkbar sind. Die Abbildung 4.5b zeigt den beschriebenen Prozess: Links oben wurde als *aktive Ecke* gewählt. Startet man hier eine Swipe- (engl. *Streich-*) Geste vom Startbildschirm aus, lässt sich die komplette Steuerung innerhalb kürzester Zeit erreichen. Von anderen Ecken oder Kanten als der Gewählten, lässt sich diese Geste nicht ausführen. Das ist wichtig, um andere Applikationen und grundlegende Funktionsweisen des Betriebssystems nicht zu behindern.

(a) Aktive Ecken und Kanten (blau)

(b) *Swipe-Geste*, beginnend in der oberen linken Ecke, zum Öffnen der HAWS Steuerung

Abb. 4.5: Konzept der aktiven Ecken zum Öffnen der HAWS Steuerung

Darstellung

Ist die Frage nach dem Starten der Steuerung beantwortet, gilt es, über die Präsentation - das User Interface - nachzudenken. Eine Smartwatch bietet wenig Raum, um viele Bedienelemente gleichzeitig anzeigen zu können. Ihr Mobilitätsvorteil wird zur Einschränkung. Es gilt diesen Nachteil mit entsprechenden Konzepten wett zu machen.

Die HAWS Applikation setzt dafür zunächst auf eine Kategorisierung der steuerbaren Aktoren. Diese Einteilung müsste konkret an die Bedürfnisse des Bewohners angepasst

Abb. 4.6: Ein Beispiel zur Kategorisierung der Aktorik

werden und sollte die Aktoren nach Häufigkeit der Verwendung oder ihrer Art gruppieren. Abbildung 4.6 zeigt, wie ein Startbildschirm mit vier geeigneten Kategorien aussehen kann. Die Kategorien sind dabei nicht durch das System vorgegeben, sondern sollen sich durch den Nutzer konfigurieren lassen. Die Liste der Kategorien und Aktoren kann bei Bedarf durch die Wear Applikationen vom HAWS Server abgerufen werden. Im Gegenzug benachrichtigt der HAWS Server die Smartwatch selbst bei Änderungen der vorhandenen Aktorik. Die Liste der Aktoren auf dem Client-Gerät bleibt so stets auf dem aktuellen Stand. Diese Liste erst im Moment des Aufrufs abzufragen, kann entscheidende Zeit kosten und das Nutzungserlebnis einschränken.

Beim Berühren einer der Kategorien, soll sich eine Liste mit Aktoren dieser Kategorie öffnen. Jedes Element dieser Liste soll den Titel des jeweiligen Aktors und dessen Status enthalten. Je nach Typ des Aktors kann dieser Status unterschiedlich dargestellt werden und mehrere Teilstatus können zusammengefügt werden. So wird etwa bei einer Lichtquelle ein klassischer An/Aus Status angezeigt, möglicherweise auch der Grad der Dimmung oder - sofern konfigurierbar - die Lichtfarbe. Abbildung 4.7 zeigt, wie diese Liste aussehen kann. Denkbar sind auch komplexere Daten, wie beispielsweise Temperaturen von Heizungen oder Öfen. Das System soll dabei möglichst modular erweiterbar sein.

Wie weiter oben beschrieben, ist die Liste der Aktoren auf dem Endgerät hinterlegt und dabei stets aktuell. Nicht vorher abrufen lassen sich dagegen die gerade benannten

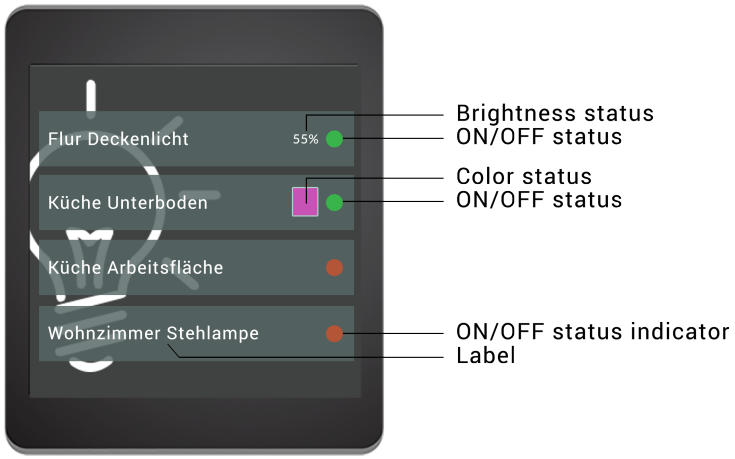


Abb. 4.7: Liste von Aktoren auf der Smartwatch

Status-Daten, da diese sich natürlich regelmäßig ändern können. Der Nutzer soll bereits mit einem schnellen Blick den aktuellen Status der Geräte ablesen können. Es darf nicht vernachlässigt werden, dass eine gute Steuerung auch zur Informationsbeschaffung dienen kann und sollte. Denn ein großer Vorteil des hier vorgestellten Konzeptes ist es, Geräte auch aus der Entfernung, beispielsweise aus dem Nebenraum, bedienen zu können. Eine zuverlässige Statusanzeige ist dabei unerlässlich. Die HAWS Applikation ist daher so entworfen, dass sie eine Liste der Aktoren vorhält und eine schnelle Navigation durch diese ermöglicht. Gleichzeitig ermittelt sie im Hintergrund den aktuellen Status der Geräte und zeigt ihn bei Empfang an. Auf diese Weise kommt es zu keinem Blockieren durch die Statusermittlung. Ohne Wissen über den absoluten Status, lassen sich die Geräte trotzdem unmittelbar steuern.
Näheres zum Ablauf dieses Verfahrens und zur Aktualisierung der Daten soll im nun folgenden Abschnitt, sowie in den Abschnitten Technische Kommunikation und Architektur, erläutert werden.
Die Sortierung der Liste erfolgt nach Häufigkeit der Verwendung bzw. Manipulation der Aktoren. Eine generell kontextabhängige Sortierung ist ebenfalls denkbar, führt aber zu häufigerem Wechsel der Reihenfolge von Einträgen. In der Folge könnte der Nutzer größere Schwierigkeiten beim Auffinden des gewünschten Bedienfelds haben. Die Idee soll allerdings im nächsten Teilkonzept wieder aufgegriffen werden.

Ist die passendere Aktorik ausfindig gemacht und wurde der Status durch den Nutzer betrachtet, so kann die Aktorik manipuliert werden. Dafür klickt der Nutzer auf den entsprechenden Eintrag. Es öffnet sich ein für die gewählte Aktorik passendes Bedienfeld. Bei einer dimmbaren Lampe ist dies zum Beispiel ein linearer Regler zur Regulierung der Helligkeit. Abbildung 4.8 visualisiert nochmals den gesamten Prozess vom Öffnen der Steuerung, bis hin zum Bedienen des genannten Reglers.

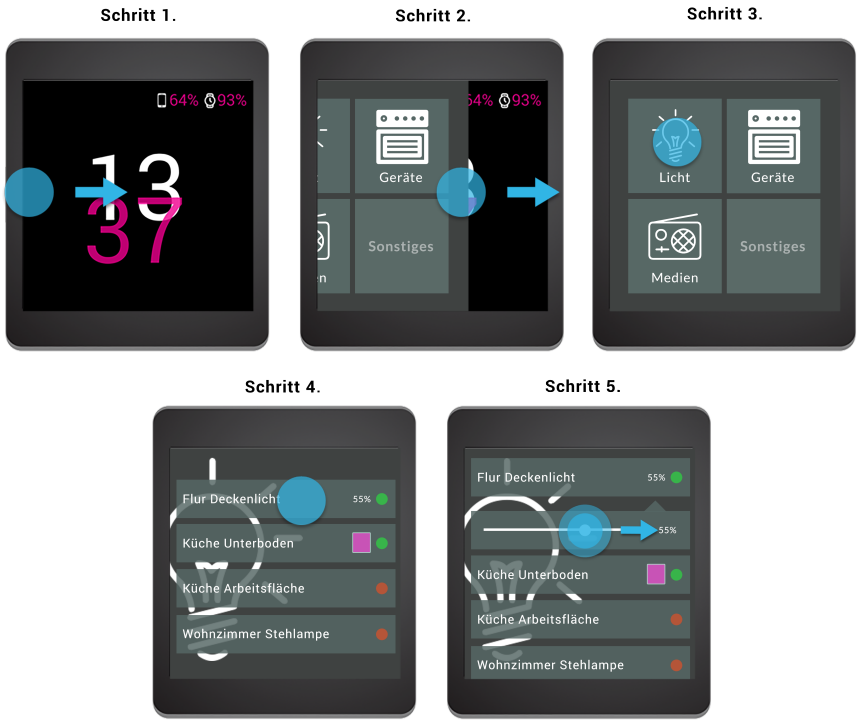


Abb. 4.8: Bedienkonzept der manuellen Steuerung im Überblick

Technische Anforderungen

In technischer Hinsicht stellt das vorgestellte Teilkonzept einige Anforderungen: Zunächst muss der Server über eine Liste der Aktoren, deren Typ und Status verfügen. Die Liste der Aktoren wird durch das Client Gerät abgerufen. Entweder geschieht das zu festen Zeitpunkten, wie etwa dem ersten Start oder wenn der Server den Client über Änderungen benachrichtigt. Der Server muss also eine entsprechende Schnittstelle zum Abgreifen dieser Daten aufweisen. Ferner muss er den Client über Änderungen informie-

ren können.
Android bietet dafür das im Kapitel 2.4 vorgestellte *Google Cloud Messaging*, dessen sich hierfür bedient wird. Schließlich muss der Server noch in der Lage sein, Steuerungs-Nachrichten anzunehmen und an die Aktorik weiterzureichen.

4.3.2 Passiv vorschlagendes System

Ein weiteres Teilkonzept ergibt sich aus dem zweiten, im Kapitel Analyse vorgestellten, Szenario. Zur Erinnerung:

> Der Nutzer möchte abends das Licht auf seiner Terrasse einschalten.

Die hier vorgestellten Teilkonzepte sollen sich gegenseitig sinnvoll ergänzen. Der Nutzer könnte also auf die gerade vorgestellte *Fernbedienung* zurückgreifen. Da das genannte Szenario aber unter Umständen durchaus öfter auftritt, lässt sich der Nutzerwunsch mit einer gewissen Wahrscheinlichkeit vorhersagen.
Das System soll diese Wünsche aus den Kontextdaten der Umgebung prognostizieren und sie dem Nutzer als Vorschläge anbieten. Ist es Abend, möchte der Bewohner möglicherweise ausgesprochen oft das Licht auf der Terrasse einschalten. Geht er in die Küche, möchte er ähnlich oft kochen, benötigt dabei vielleicht Musik und eine eingeschaltete Beleuchtung über der Arbeitsfläche. Daten wie Zeit, Aufenthaltsort und zum Beispiel der Status anderer Geräte dienen als Grundlage für diese Vorhersagen.
Der HAWS Server tauscht sich über die vorhandene Blackboard Architektur mit der *Machine Learning* Komponente aus, um die Wahrscheinlichkeitswerte für den Steuerungswunsch verschiedener Aktoren zu ermitteln. Wird ein festgelegter Schwellenwert dabei überschritten, so wird das Ereignis als wahrscheinlich eingestuft (s. Kap. 4.5). Näheres zur Verteilung der verschiedenen Dienste wird im Abschnitt *Architektur* (Kap. 4.4) beschrieben.
Das System macht dem Bewohner dann zum geeigneten Zeitpunkt Vorschläge über passende Aktionen. Da das System allerdings nicht sicher wissen kann, ob der Nutzer überhaupt handeln möchte, soll er nicht alarmiert werden. Dies könnte den Nutzer, spätestens auf lange Sicht gesehen, belästigen. Folglich gibt es keine Benachrichtigung und keine autonome Handlung durch das Smart Home System. Die Vorschläge erfolgen *passiv*. Erst wenn der Nutzer selbst eine Aktion ausführen möchte, sucht er die Interaktion über sein Wearable und kann schnell aus vorgeschlagenen Bedienelementen wählen.

Systemintegration

Wie gilt es, diese weitere Interaktionsmöglichkeit in die Smartwatch zu integrieren? Sie soll sich deutlich von der ersten Steuerung unterscheiden. Der Nutzer soll differenzieren können, ob er manuell schalten möchte - wie im vorherigen Abschnitt beschrieben - oder ob er die Vorschläge des Systems in Anspruch nehmen möchte. Diese umfassen natürlich nicht alle Aktoren und sind nicht kategorisiert. Schließlich können die derzeit passenden Aktionen aus ganz verschiedenen Gruppen stammen.

Die Liste der Vorschläge soll allerdings ähnlich schnell erreichbar sein, wie die vorgestellte manuelle Steuerung. Diese lässt sich, wie beschrieben, durch aktive Ecken starten. Der Nutzer muss also *Hand anlegen*, bevor er die Aktoren bedienen kann. Das passt sehr gut zum Gedanken einer Fernbedienung. Die Hand wird ohnehin in jedem Fall auf dem Gerät benötigt.

Die Liste der Vorschläge dagegen soll ein unverbindlicheres Konzept darstellen und am besten ohne den Gebrauch einer zusätzlichen Hand einsehbar sein. Auf diese Weise kann der Nutzer die Vorschläge fast nur durch ein Armheben analysieren. Ist kein passender Vorschlag dabei, senkt er den Arm wieder und der Dialog ist beendet. Ein einfaches Armheben genügt allerdings nicht ganz, da normalerweise nur die Uhr und aktive Benachrichtigungen angezeigt würden. Diese sollen schließlich nicht permanent überlagert werden. Inspiriert durch das Bedienkonzept der von dem Unternehmen Microsoft entwickelten Such-Applikation *Torque*, führt ein hin und her drehen des Handgelenks zum Öffnen der Vorschlagsliste (vgl. Microsoft, 2014). Der Nutzer hebt also den Arm, die Uhr bzw. ihr Display wird aktiv und der Nutzer dreht anschließend sein Handgelenk mindestens einmal kurz vor und zurück entlang der Längsachse des Arms (vgl. Abb. 4.9).

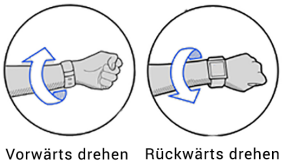


Abb. 4.9: Handgelenksgeste zum Anzeigen der Vorschläge (orig. Microsoft, 2014)

Darstellung

Nach erfolgreichem Starten wird eine Liste gezeigt, die optisch und funktional fast identisch mit der Kategorieansicht des ersten Konzeptes ist. Wieder werden Titel, Status und Steuerelemente gezeigt (vgl. Abb. 4.10). Hier ist die Liste allerdings absteigend

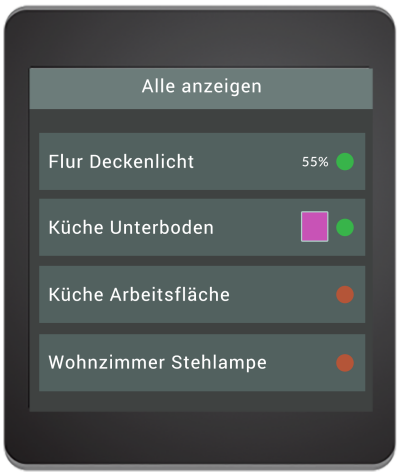


Abb. 4.10: Liste passiver Vorschläge

nach der ermittelten Wahrscheinlichkeit für die einzelnen Aktoren sortiert. Dabei wird der Einfachheit halber derzeit nur die Wahrscheinlichkeit berücksichtigt, mit der der Nutzer den jeweiligen Aktor steuern möchte. Nicht berücksichtigt sind die Einstellungen des Aktors. Denkbar ist in Zukunft also auch, dem Nutzer bereits Vorschläge für die jeweiligen Einstellungen zu unterbreiten.

Zusätzlich gibt es die Schaltfläche „Alle anzeigen“, die einen schnellen Wechsel zur manuellen Steuerung ermöglichen soll. Sind keine, oder nicht alle erhofften Vorschläge vorhanden, kann der Nutzer die entsprechenden Bedienelemente schnell selbst ausfindig machen.

Technische Anforderungen

Die technischen Anforderungen an das vorgestellte Interaktionsmuster sind in weiten Teilen identisch zu dem ersten Muster. Die Liste der Vorschläge profitiert allerdings ungemein von ihrer kontextabhängigen Aktualität. Sie wird daher erst in dem Moment vom Server abgerufen, da der Benutzer diese erfragt. Sicher könnte der Server das Clientgerät bei jedem Kontextwechsel bzw. damit verbundenen Wahrscheinlichkeitsänderungen informieren, der entstehende Netzwerkverkehr wäre allerdings unverhältnismäßig. Dies würde auch zu Lasten der kleinen Smartwatchakkus gehen.
Die Wahrscheinlickeitsermittlung der einzelnen Aktoren erfolgt serverseitig durch die

Machine learning engine, deren Ausarbeitung allerdings nicht Teil dieser Thesis ist. Welche Bedingungen für diese Komponente gestellt werden müssen, lässt sich aber dem gleichnamigen Kapitel 4.5 entnehmen.

4.3.3 Aktiv vorschlagendes System

> Der Bewohner kommt abends vom Einkaufen nach Hause. Er trägt schwere Taschen und möchte gerne Licht im Eingangsbereich haben.

Dieses Szenario wurde im Kapitel *Analyse* als denkbare Ausnahme von dem passiven Verhalten des Systems genannt. Der Nutzer ist beschäftigt, nicht in der Lage oder einfach zu bequem, die Aktorik selbst zu kontrollieren. In jedem Fall wäre er froh darüber, wenn das System die Aktorik in seinem Sinne steuert.

Gibt die *Machine learning engine* eine entsprechend hohe Warscheinlichkeit für einen Aktor aus, sodass dessen baldige Steuerung durch den Nutzer als gesichert gelten kann, wird aktiv eine Benachrichtigung an den Nutzer abgesondert. Denkbar ist auch, dass dieser Wahrscheinlickeitswert nicht nur duch klassische Lernmethoden ermittelt wird, sondern ergänzend dazu auch regelbasiert. So lässt sich das beschriebene *Nach Hause kommen* sehr gut mit Regeln modellieren.

Systemintegration

Das Smartphone des Nutzers erhält eine im Kapitel 2.4 beschriebene Push-Nachricht. Diese wird wiederum unverzüglich mithilfe der im Kapitel 2.5 beschriebenen *Messages* an die Smartwatch weitergeleitet. Die abgesetzte Benachrichtigung ist eine Möglichkeit für den Benutzer, in einen automatischen Prozess einzugreifen. Seine Uhr vibriert, leuchtet auf und zeigt ihm an, welche Steuerbefehle das System im Begriff ist, abzusetzen.
Es ist keine direkte Aktion des Nutzers zum Auslösen dieser Nachricht nötig. Höchstens indirekt führt er sie durch die Beeinflussung des Kontextes und der vorhandenen Sensorik herbei.

Darstellung

Wird die Nachricht empfangen, öffnet sich umgehend die Liste mit den Vorschlägen. Sie ähnelt der Liste aus dem passiven Teilkonzept, hat aber ein paar zusätzliche Anpassungen. Grundsätzlich gilt es möglichst viele optische Komponenten wiederzuverwenden - zwecks Wartbarkeit und *User Experience*. Ein nennenswerter Unterschied ist, dass der in der Liste gezeigte Status nicht der aktuelle, sondern der geplante Status ist. Wie dieser sich zusammensetzt, obliegt der *Machine learning engine*. Denkbar ist hier ebenfalls Lernverfahren einzusetzen oder beispielweise die letzte oder häufigste Einstellung wiederherzustellen.
In Anlehnung an unser Szenario könnte der beschriebene Prozess wie in Abbildung 4.11 aussehen. Ein voreingestelltes Zeitintervall beginnt, ersichtlich für den Nutzer, herunter-

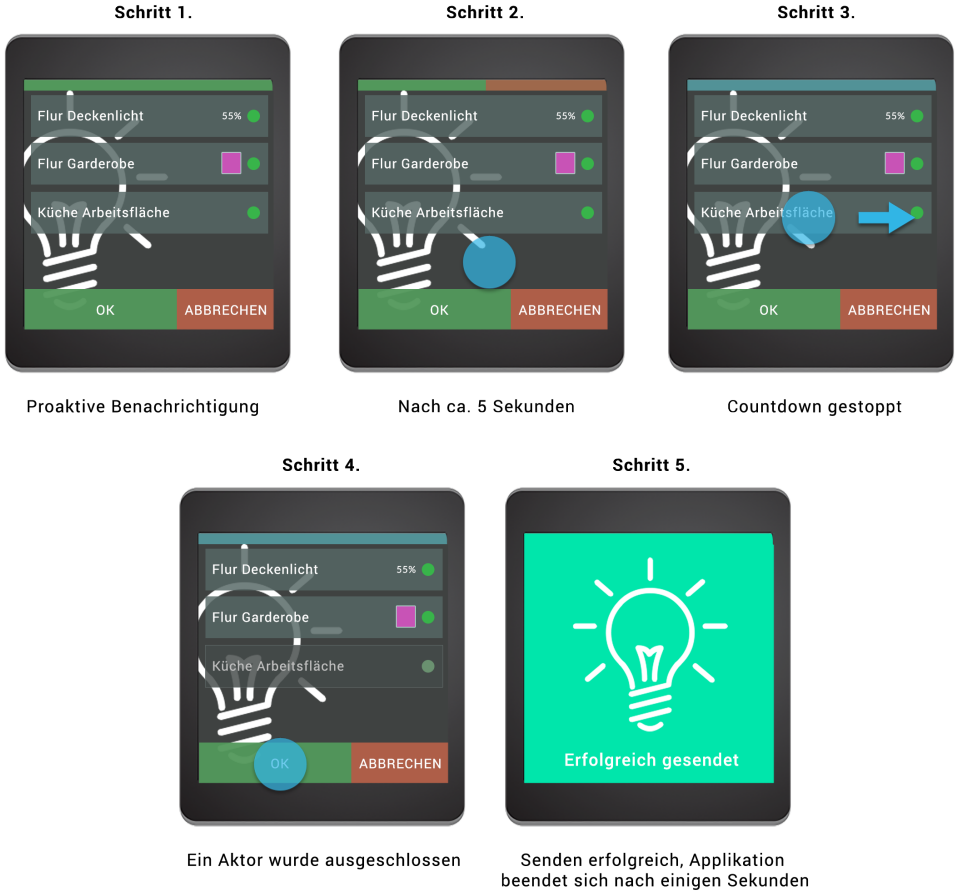


Abb. 4.11: Liste aktiver Vorschläge mit manuellem Eingriff der Nutzers

zuzählen. Erfolgt in dieser Zeit keine Interaktion, werden die Befehle ausgeführt. Der Countdown wird durch einen farbigen Zeitstrahl visualisiert (s. oben), der zunächst in Anlehnung an den *OK* Button grün ist. Der Strahl wird in Anlehnung an den *Abbrechen* Button zunehmend von links nach rechts mit Rot aufgefüllt.
Eine einzige Berührung des Bildschirms stoppt den Countdown und der Nutzer hat unbegrenzt viel Zeit einzugreifen. Der Zeitstrahl erscheint in einem neutralen Blau. Nun kann er die Aktoren, wie aus den anderen Teilkonzepten gewohnt, anpassen. Er kann beispielsweise die Helligkeit von Lichtern regulieren, aber auch Aktoren komplett aus dem Szenario entfernen. Abbildung 4.11 zeigt beispielsweise, wie der Aktor *Küche Arbeitsfläche* mithilfe einer Streich-Bewegung aus der Liste entfernt bzw. ausgegraut wird (s. Schritt 3).
Diese Anweisungen werden folglich nicht direkt bei ihrer möglichen Anpassung durch den Nutzer ausgeführt. Erst eine Bestätigung, durch Klick auf *OK*, führt alle Aktorbefehle zusammen aus. Hintergrund ist, dass der Nutzer vielleicht nur einige wenige von vielen Vorschlägen bearbeiten möchte. Jene, welche er nicht bearbeitet, sollen nicht erst manuell ausgelöst werden müssen.

Die technischen Anforderungen an dieses Interaktionsmuster wurden bereits in den Abschnitten *Darstellung* und *Systemintegration* spezifiziert. Eine nähere Betrachtung der technischen Besonderheiten, und insbesondere der technischen Kommunikation an diesen Teil des Gesamtkonzeptes, wird in Kapitel 4.4.1 erfolgen.

4.3.4 Zusammenfassung

In diesem Kapitel wurden die Szenarien der Analyse erneut aufgegriffen, um passende Lösungen für diese zu präsentieren. Es wurde gezeigt, wie dem Nutzer eine zwar herkömmliche, aber durchdachte Fernbedienung zur manuellen Steuerung geboten wird. Darauf aufbauend wurde das Konzept der passiven Vorschläge betrachtet. Es soll dem Nutzer mit möglichst wenig Interaktion stets die passenden Aktionsvorschläge liefern. Komplettiert wurden die beiden Interaktionsmuster schließlich durch das dritte Konzept der aktiven Vorschläge. Durch sie kann der Interaktionsbedarf in Sonderfällen auf Null reduziert werden. Der Nutzer kann aber anders als in einem vollautomatischen System stets in der Prozess eingreifen.

In den nun folgenden Kapiteln sollen die technischen Aspekte des vorgestellten Konzeptes betrachtet werden.

4.4 Architektur

Dieses Kapitel wird die grundlegenden Design-Entscheidungen zum Aufbau des HAW Systems beschreiben.

Wie bereits in den anderen Abschnitten erwähnt wurde, besteht das System grob aus der Android Wear Applikation auf der Smartwatch, der Android Applikation auf dem Smartphone und dem HAWS Server (vgl. Abb. 4.12).

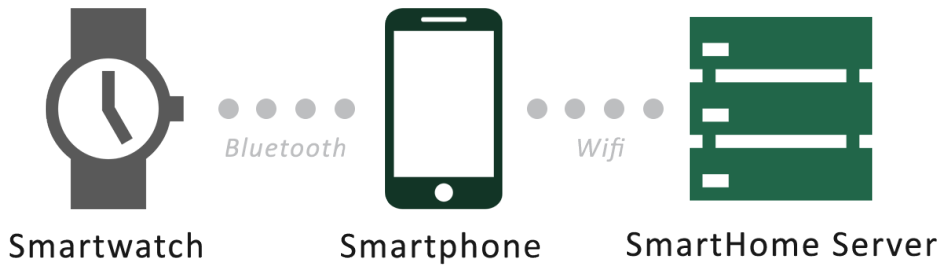


Abb. 4.12: Verteilung des HAW Systems

Während Smartwatch und Smartphone sich über Bluetooth austauschen, ist die Verbindung zwischen Smarthpone und Server eine Netzwerkverbindung. In der Regel wird dabei WLAN verwendet. Die Verbindung ist allerdings nicht auf ein lokales Netzwerk beschränkt, sondern kann auch global über das Internet hergestellt werden.

Weiter verdeutlicht wird die Verteilung durch nachfolgende Grafik (Abb. 4.13). Erkennbar werden vor allem die Abhängigkeit der Wear Applikation von dem Android Gerät, sowie die der HAWS Applikationen vom jeweiligen Betriebssystem.
Die Bluetooth Verbindung zwischen Smartwatch und Smartphone wird aufgeschlüsselt in die, unter Kapitel 2.5 vorgestellten, sog. *Messages* und *Data Items*.
Die Schnittstelle des Servers steht in Form einer REST API zur Verfügung. Umgekehrt kann der Server den Client durch sog. Push-Nachrichten über Änderungen informieren (vgl. Kap. 2.4). Das Programmierparadigma *Representational State Transfer* (REST) wurde im Jahr 2000 durch Roy Fielding als Dissertationsthema eingereicht (vgl. Fielding, 2000). Es spezifiziert im Rahmen des HTTP Protokolls wie Adresse (URL) und Inhalt sich zueinander verhalten sollten und wurde bei der Entwicklung der Schnittstelle berücksichtigt.

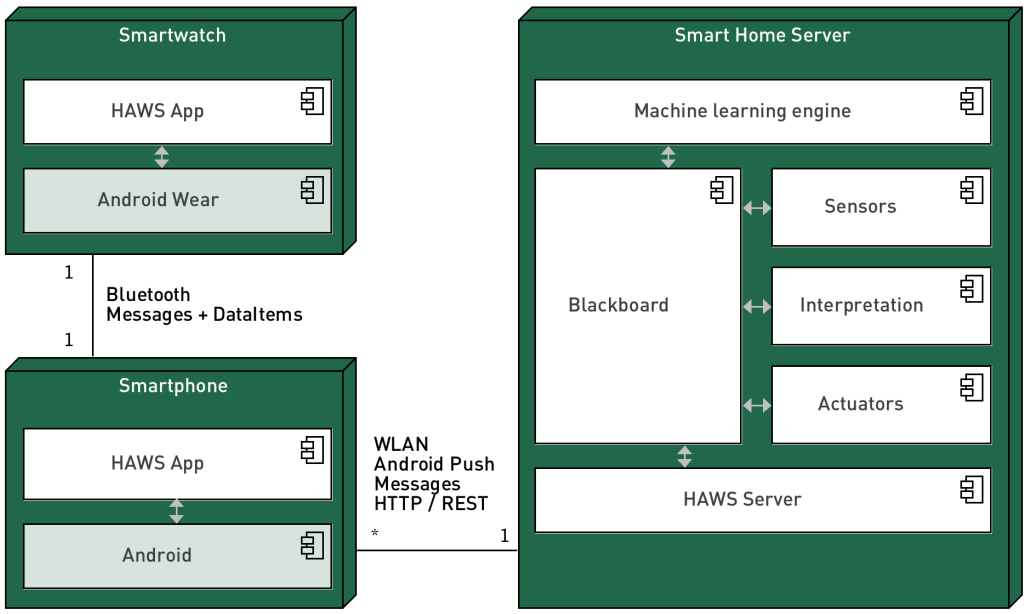


Abb. 4.13: Architektur des HAW Systems

Der Smart Home Server wird vorausgesetzt und soll durch einen Adapter an den HAWS Server gebunden werden. Wie bereits mehrfach erwähnt, kommt dabei eine Blackboard-Architektur zum Einsatz. Durch die mit dem Adapter entstehende Flexibilität sind aber auch andere Architekturen denkbar. Der Adapter ist selbst Teil des HAWS Server und konvertiert Informationen bzw. Nachrichten des Blackboards entsprechend hin und zurück.
Der HAWS Server ist nicht auf die eigene Interpretation von Sensordaten angewiesen. Alles was er an Kontextdaten benötigt, erhält er durch die *Machine learning engine*, welche sich mit dem Adapter in einem einheitlichen Format verständigen soll. Einzig den aktuellen Status der Aktoren muss der HAWS Server selber vom Blackboard abgreifen. Eintreffende Steuernachrichten des Clients werden durch den HAWS Server in ein, für die einzelnen Akoren verständliches, Format umgewandelt und auf dem Blackboard publiziert.

Betrachtet man das vorgestellte Blackboard auf abstrakter Ebene und sieht sich die Nutzung der Schnittstellen an, gelangt man zu einem Bild, wie es Abbildung 4.14 veranschaulicht. Kontaktpunkte mit mit dem HAWS Server sind weiß, Schnittstellen mit denen der HAWS Server nicht direkt interagiert sind ausgegraut.

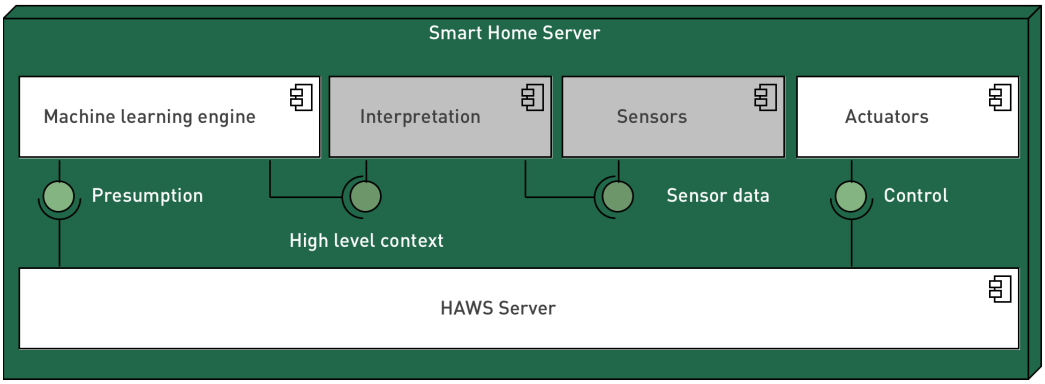


Abb. 4.14: Schnittstellen innerhalb eines musterhaften Smart Home Servers

4.4.1 Ablaufszenarien

Zur Verdeutlichung der Architektur und Herausstellung einiger Besonderheiten sollen im folgenden Szenarien in verschiedenen Schichten der Architektur untersucht werden. Dazu werden primär die vorgestellten Interaktionsmuster betrachtet.

„Klassische“ Steuerung

Das Interaktionsmuster der klassischen Steuerung bedingt zunächst Wissen des Clients über die Aktoren und dessen zugehörigen Kategorien. Um dem Client eine Liste der Aktorik zu Verfügung zu stellen, muss auch der HAWS Server zunächst in Besitz dieser Daten gelangen. Der Server pflegt dazu eine persistente Datenbasis über die vorhandene Aktorik. Die Aktorik kann dabei manuell durch den Benutzer eingetragen werden oder ggf. automatisch entdeckt werden.
Zentrale Idee ist, dass jeder Typ von Aktoren auf dem Server durch eine Klasse dargestellt wird. Diese Typ-Klassen können voneinander erben und so Spezialisierungen ausbilden. Eine *Leuchte* wird beispielsweise durch eine *RGB-Leuchte* spezialisiert. Beide Klassen sind allerdings selbst nur abstrakt. Funktional werden sie erst durch einen implementierenden Adapter, eine auf ein spezielles Gerät zugeschnitten Protokolleinheit. Abbildung 4.15 zeigt eine solche Klassenhierarchie am Beispiel einer RGB Leuchte, mit der sich Licht in verschiedenen Farben erzeugen lässt. Das eigentliche Gerät soll ein sog. *LW12 RGB Controller* sein, der per WLAN ansteuerbar ist und angeschlossene RGB Leuchten auf die gewünschte Farbe konfiguriert. Interfaces und Attribute sind an

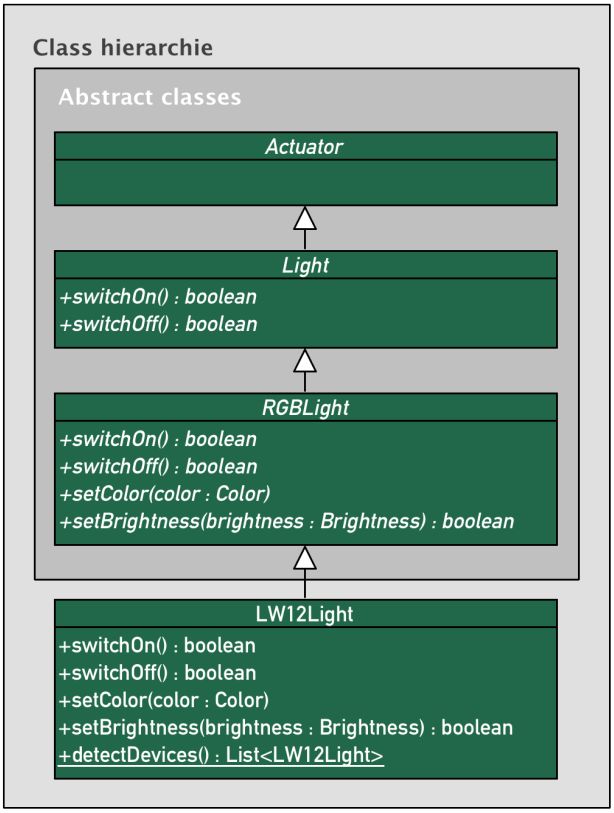


Abb. 4.15: Vereinfachte Klassenhierarchie zur Steuerung einer RGB Leuchte

dieser Stelle nicht berücksichtigt. Die oberen drei Klassen der Hierarchie sind abstrakt und werden erst durch die Klasse *LW12Light* und ihre Methoden realisiert. Die Klasse LW12Light ist ein speziell für diesen LED Controller geschriebener Adapter, der das entsprechende Protokoll des jeweiligen Aktors spricht und hierarchisch gesehen nach oben hin abstrahiert.
Je nach Art der Aktorik, ist auch ein Auto-Discovery-Mechanismus denkbar. Er wird beispielsweise beim genannten Controller als statische Methode der Realisierungsklasse implementiert und liefert alle gefundenen Geräte zurück.
Weiterhin ist die Klasse für die Annahme oder Abfrage von Statusinformationen verantwortlich. Je nach Typ der Aktorik werden diese aktiv durch den Aktor mitgeteilt, müssen abgefragt werden oder können unter der Annahme der Abgeschlossenheit des Systems (*closed world assumption*) vermutet werden. Letzteres bedeutet, dass die Lampe nach dem Einschalten so lange für *AN* gehalten wird, bis das System sie wieder abschaltet.

Nun da der Server in der Lage ist, die vorhandenen Aktoren zu ermitteln, können sie durch den Client abgerufen und verwendet werden. Das Sequenzdiagramm unter Abbildung 4.16 zeigt, wie die Liste der Aktoren regelmäßig durch das Smartphone ermittelt wird [1 - 1.1]. Sie werden unverzüglich in die, in Kapitel 2.5 beschriebenen Data

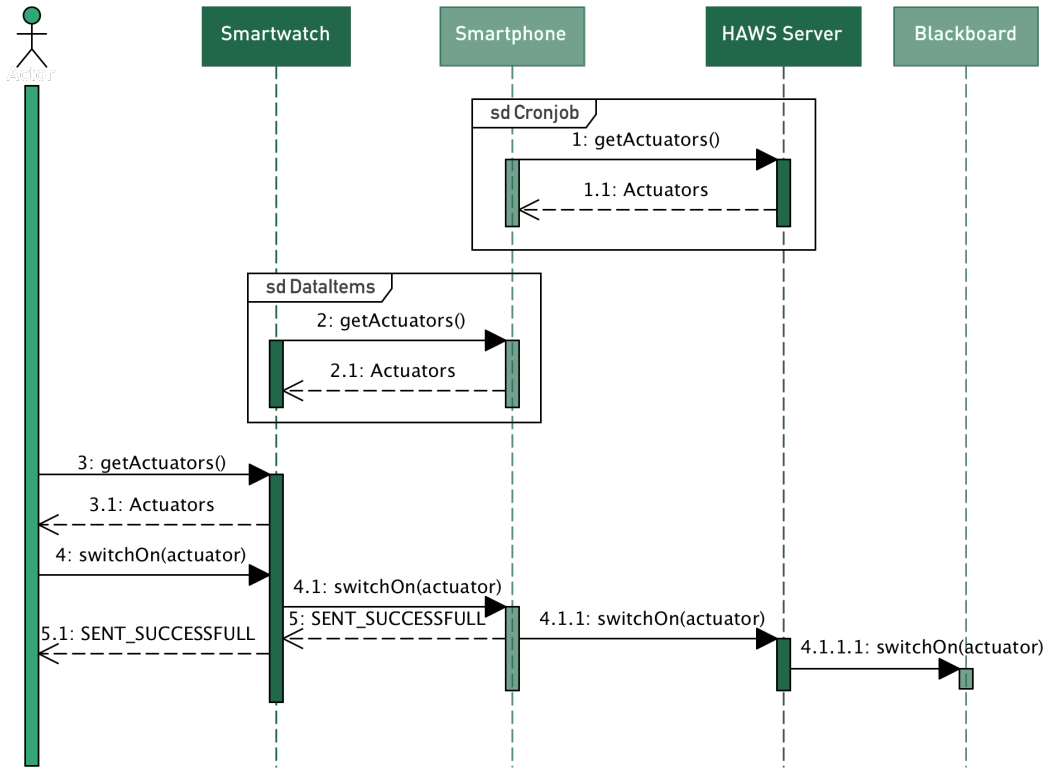


Abb. 4.16: Sequenzdiagramm: Notwendige Schritte zum Nutzen der *klassischen* Steuerung

Items gespeichert. Android und AndroidWear synchronisieren diese automatisch mit der Smartwatch [2 - 2.1]. Dieser Prozess ist nicht zeitkritisch, da die Aktoren in einer gängigen Smart Home Umgebung nicht sehr stark fluktuieren. In der Regel erfolgt die Synchronisierung durch das Android System dennoch ohne spürbare Verzögerung.
In den Schritten 3 bis 3.1 ruft der Nutzer die Steuerung auf seiner Smartwatch auf. Er entscheidet sich für einen Aktor, den er anschalten möchte. Seine Smartwatch setzt jetzt eine *Message* (vgl. Kap. 2.5) an das gekoppelte Smartphone ab [4.1]. Das Smartphone wiederum macht daraus einen REST-Call an den HAWS Server [4.1.1]. War die Kommunikation mit der REST-Schnittstelle erfolgreich, gibt das Smartphone diese Information an die Smartwatch - und damit an den Nutzer - weiter [5 - 5.1]. Der Server steuert schließlich über den zuvor beschriebenen Adapter den entsprechenden Aktor an. Der

Adapter wird anhand des Aktortyps gewählt, welcher ebenfalls Teil der Nachricht ist. Näheres zu den Nachrichten und der REST-Schnittstelle folgt im Abschnitt *Technische Kommunikation*.

Passiv vorschlagendes System

Als nächstes werden die notwendigen Schritte zum Erhalt der passiven Aktionsvorschläge betrachtet. Wie bereits im Abschnitt Interaktionsmuster beschrieben, werden die Vorschläge auf Client-Seite nicht zwischengespeichert. Dadurch sollen Netzwerkverkehr und Akkunutzung reduziert werden. Auch ein regelmäßiges *Polling* der Daten ist aus Gründen der Aktualität nicht zweckmäßig, da der Kontext sich sehr oft und schnell ändern kann.

Das Diagramm unter Abbildung 4.17 beschreibt zunächst wie der HAWS Server die

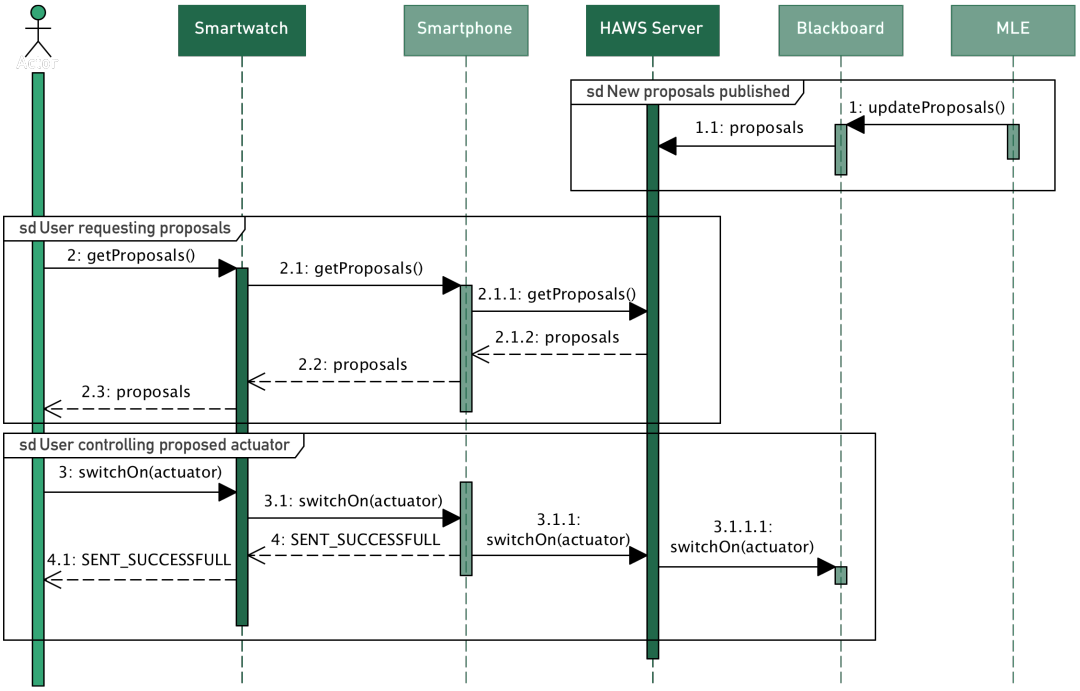


Abb. 4.17: Sequenzdiagramm: Passive Vorschläge

Vorschläge der *Machine learning engine* (MLE) über das Blackboard als Message Broker erhält [1 - 1.1]. Dies findet regelmäßig mit jeder Neuinterpretation des aktuellen Kontextes durch die Machine learning engine statt.

Erbittet der Nutzer nun durch die entsprechende Bewegungs-Geste seiner Smartwatch die Liste der Vorschläge zu sehen [2], wird diese Anfrage bis an den HAWS Server delegiert. Dieser liefert die aktuelle, ihm bekannte Liste zurück [2.1 - 2.3].
Schließlich kann der Nutzer, ähnlich wie bereits am Sequenzdiagramm der manuellen Steuerung beschrieben, den Befehl zum Ausführen einer Aktion geben [3 - 4.1].

Aktiv vorschlagendes System

Schließlich soll auch das dritte Szenario betrachtet werden. Die Abbildung 4.18 veranschaulicht, wie dem Nutzer proaktiv Vorschläge durch das System präsentiert werden.

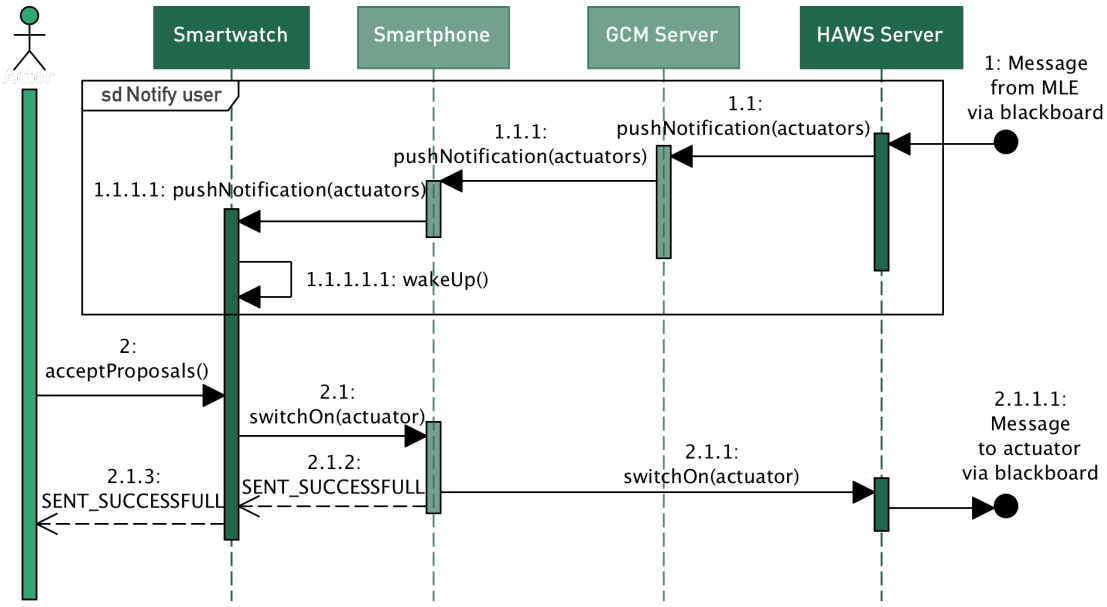


Abb. 4.18: Sequenzdiagramm: Aktive Vorschläge

Es gilt generell zu differenzieren, ob der Nutzer diese Empfehlungen unverändert und ohne Interaktion akzeptiert, oder ob er eingreift und die Aktoren modifiziert bzw. den Dialog komplett abbricht. Das Diagramm zeigt den einfachen Fall, in dem der Nutzer die Vorschläge ohne weitere Interaktion akzeptiert. Aber auch eine Änderung der Vorschläge würden, abgesehen von der zusätzlichen Nutzerinteraktion, keinen anderen Ablauf bewirken. Ein Abbruch hätte wiederum lediglich den Wegfall der Nachrichten 2 - 2.1.3 zur Folge.

Betrachtet man die Abbildung 4.18 am Beginn, fällt auf, dass die Interaktion hier vom Server ausgeht. Liefert die *Machine learning engine* dem Server besonders hohe Wahrscheinlichkeitswerte für die jeweiligen Aktoren, so sendet der Server eine Push-Benachrichtigung an den Client [1 - 1.1.1]. Dies geschieht über den dafür vorgesehenen *Google Cloud Messaging* Server (*hier: GCM Server*). Kommt die Nachricht auf dem Wearable an, wird das Gerät aufgeweckt. Es vibriert und informiert den Nutzer über vorgeschlagene Aktorik. Wie in Kapitel 4.3.3 beschrieben, kann er nun mit dem Gerät interagieren, oder aber auch einfach nichts tun. Durch Akzeptieren oder fehlende Interaktion wird, ähnlich wie in den letzten beiden Abschnitten beschrieben, der Befehl zur Steuerung der Aktorik über das Smartphone an den Server gesandt [2 - 2.1.1.1]. Der Nutzer erhält umgehend die Erfolgsnachricht durch das Smartphone auf seine Smartwatch gesendet [2.1.2 - 2.1.3].

Lehnt er die Vorschläge komplett ab, würde es keine weiteren Nachrichten durch die Smartwatch an Smartphone oder gar den Server geben. Die Nachrichten 2.1 bis 2.1.1.1 würden somit wegfallen.

Anders als die anderen Konzepte weist dieses in Hinblick auf Mehrbenutzerfähigkeit eine Besonderheit auf. Das System muss sich entscheiden, welchen Nutzer es benachrichtigt. Das derzeitige Konzept sieht vor, nur einen anwesenden Nutzer zu benachrichtigen. Für die Zukunft ist es allerdings auch denkbar, den Nutzer zu benachrichtigen, der näher an der zu schaltenden Aktorik ist, oder etwa auch alle Nutzer zu benachrichtigen. Sobald einer interagiert, würde der Dialog für alle anderen abgebrochen, um ein Hin- und Herschalten der Aktorik zu vermeiden.

Die oben dargestellten Diagramme boten bereits einen tieferen Einblick in die Kommunikation und Funktionsweise der zu Beginn dieses Kapitels beschriebenen Systemkomponenten. Der Aufbau dieser Komponenten soll in den nun folgenden Abschnitten beschrieben werden.

4.4.2 HAWS Server

Der bereits vielfach erwähnte HAWS Server basiert auf dem PHP Framework *Laravel*. Laravel ist ein Open-Source-Framework, das 2011 von Taylor Ortwell initiiert wurde (s. a. Taylor Otwell, 2014). Es folgt dem MVC (Model-View-Controller) Prinzip und ist durch sein geringes Alter frei von Altlasten. Das Framework zieht weiterhin großen Nutzen aus den Neuerungen aktueller PHP Versionen, wie beispielsweise Namespaces und anonyme Funktionen (Lambdas).
Die Entscheidung einen Webserver als Backend zu nutzen, ist unter anderem durch die gute REST Unterstützung, sowie die Möglichkeit später Web-Clients bequem mit einbinden zu können, getroffen worden.

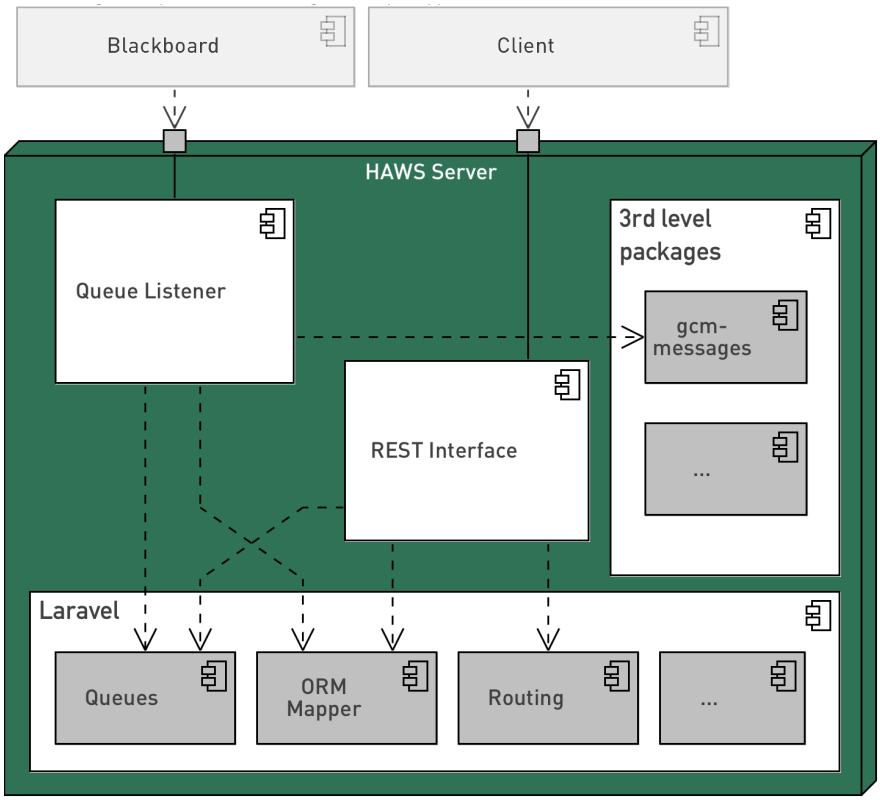


Abb. 4.19: Architektur des HAWS Servers

Die Abbildung 4.19 beschreibt den vereinfachten Aufbau des HAWS Servers. Grundlage bildet dabei das MVC Framework mit seinen vielen hilfreichen Komponenten. Ergänzt werden diese durch externe Komponenten, wie zum Beispiel solche, die die Nutzung der

Google Cloud Messaging Dienste für Push-Benachrichtigungen ermöglicht.
Die Schnittstelle nach außen wird gebildet durch das REST Interface und den Queue-Listener Service. Der letztere Dienst wartet auf eingehende Nachrichten der MLE oder etwa Statusänderungen von Aktoren. Anders als über einen Webserver und damit PHP unüblich, wird dieser Teil über die Konsole als Dienst gestartet und läuft permanent im Hintergrund. Für das Senden und Empfangen von Nachrichten unterstützt Laravel gängige Queues wie Beanstalkd, IronMQ, Amazon SQS und Redis. Durch die vielen, verfügbaren Erweiterungen ließe sich diese Liste noch fortführen. Der jeweilige Dienst ist dabei das, was zu Beginn dieses Kapitels als *Blackboard* beschrieben wurde. Er übernimmt und steuert die Kommunikation zwischen den einzelnen Komponenten des Smart Home Systems.

Der HAWS Server abonniert Nachrichten der MLE über aktualisierte Wahrscheinlichkeitswerte, sowie Nachrichten über neue Aktoren oder Änderungen von deren Status. Relevante Änderungen führen dann zum Absetzen einer Push-Benachrichtigung an den Client, wie es zuvor im Abschnitt 4.3.3 beschrieben wurde.

Die zweite wichtige Schnittstelle des HAWS Server ist die REST-Schnittstelle - ein Webservice, welcher durch den Client bzw. das Smartphone genutzt werden kann. Laravel bietet hierfür ein sehr ausgefeiltes Routing-System. Die Schnittstelle übernimmt die Registrierung der Clients, liefert vorhandene Aktoren, sowie Vorschläge und nimmt wiederum Steuerungsbefehle der Clients entgegen.
Im Folgenden sollen die wichtigsten Routen dieser Schnittstelle zusammenfassend beschrieben werden. Viele der zuvor in den Sequenzdiagrammen beschriebene Nachrichten finden sich hier wieder. Die Liste erhebt allerdings keinen Anspruch auf Vollständigkeit. Diverse Put- und Delete-Methoden werden aus Gründen der Übersichtlichkeit an dieser Stelle vernachlässigt.

Das Prefix */v1* stellt eine Versionierung in Hinblick auf mögliche Änderungen an der API dar. Die letzte Route */v1/actuators/instructions* entspricht nicht vollständig der REST Spezifikation, wie sie Roy Fielding im Jahr 2000 erarbeitet hat (vgl. Fielding, 2000). Das Erstellen mehrerer Einträge ist nicht vorgesehen, hat sich aber in der Praxis als sehr nützlich bewiesen. Das aktive Interaktionskonzept setzt beispielsweise unter Umständen viele Befehle für verschiedene Aktoren ab. Nach standardkonformen REST wäre jeder Befehl ein eigener Request. Das derzeitige Design dieser Route sieht nur einen einzigen

Verb	Resource	Beschreibung
Client Verwaltung		
POST	/v1/clients	Erstellt einen neuen Client und registriert ihn unter anderem für GCM. *Parameter:* ID, GCM Device Token
Manuelle Steuerung und passive Vorschläge		
GET	/v1/actuators	Liefert eine Liste der steuerbaren Aktoren inklusive ihrer Kategorien zurück.
GET	/v1/actuators/proposed	Liefert Liste der aktuell vorgeschlagenen Aktoren zurück.
Aktor Steuerung		
POST	/v1/actuators/instructions	Erstellt neue Befehle zur Steuerung eines oder mehrerer Aktoren.

Tabelle 4.1: Auszug aus der Dokumentation der REST API des HAWS Servers

vor. Näheres zu den Inhalten der Server-Antworten findet sich im Abschnitt Technische Kommunikation.

Denkbar wäre generell auch, eine einheitliche Schnittstelle für die Clients und für interne Komponenten, wie MLE und Aktoren, zu schaffen. Da allerdings für ein realistisches System entwickelt wird, dessen Kernvoraussetzung in Anlehnung an das *Living Place Hamburg* eine Blackboard Architektur in Form eines Queue-Messaging-Systems ist, lassen sich diese Komponenten nicht ohne weiteres auf eine REST-Schnittstelle umstellen. Umgekehrt sollen die Clients losgebunden von der Technologie des Smart Home Systems sein, damit die HAWS Applikation als eigenständige und portierbare Komponente bestehen bleibt.

4.4.3 Android Applikation

Die Android Applikation auf dem Smartphone dient als Vermittler zwischen Smartwatch und Server. Sie ist folglich Nutzer der gerade beschriebenen REST Schnittstelle.
Wie Abbildung 4.20 veranschaulicht, gibt es kein User Interface. Die Applikation ist eine Ansammlung von Services zur Kommunikation. Es gibt nach außen hin zwei Schnittstellen, die über das Android Manifest am System registriert werden. Beim Eintreffen von Nachrichten des GCM Servers oder des Wearables werden automatisch die entsprechend registrierten Services zur Bearbeitung gestartet.
Trifft eine Nachricht vom GCM Server ein, wird diese durch den *GCM Intent Service*

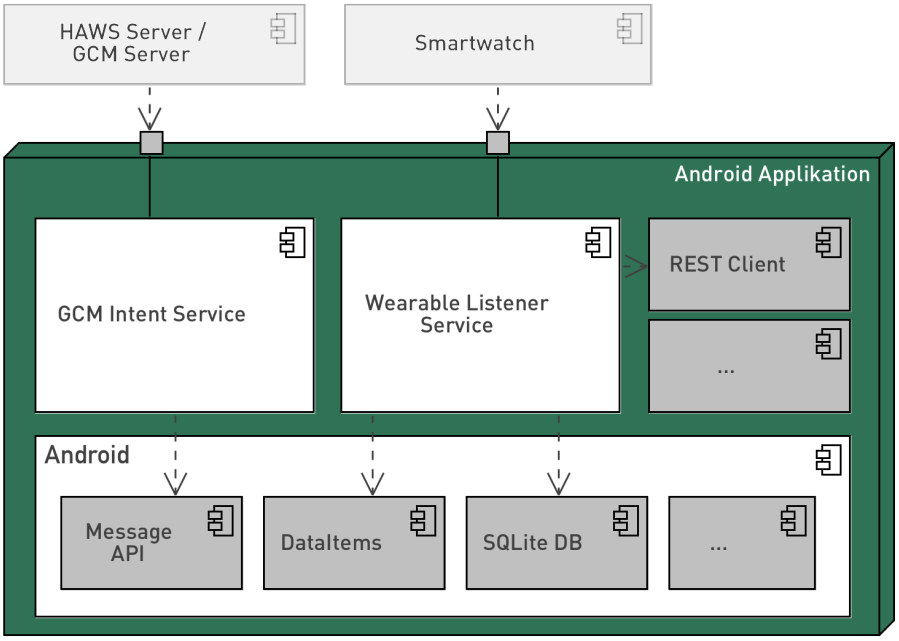


Abb. 4.20: Architektur der HAWS Smartphone Applikation

bearbeitet. Wie aus den Sequenzdiagrammen der letzten Abschnitte hervorgeht, handelt es sich dabei primär um aktive Vorschläge, die sich durch ihre hohen Wahrscheinlichkeitswerte auszeichnen. Der Service leitet diese umgehend über die *Message API* an die Smartwatch weiter.
Die Smartwatch wiederum schickt nach erfolgreicher Nutzerinteraktion Steuerungsbefehle zurück an das Smartphone. Das System startet den *Wearable Listener Service* zur Verarbeitung dieser Nachrichten. Mithilfe des *REST-Clients* werden die Befehle an den Server weitergeleitet.
Weiterhin werden Anfragen der Smartwatch über aktuelle Aktorlisten und passive Vorschläge beantwortet. Sie werden erneut durch den REST Client an den Server delegiert, können aber auch aus der *SQLite Datenbank* gelesen werden, wo sie zwischengespeichert werden. Die Ergebnisse werden außerdem als Data Items automatisch synchronisiert und auf Smartwatch und Smartphone persistiert.

4.4.4 Android Wear Applikation

Schließlich soll der Aufbau der Android Wear Applikation betrachtet werden. Sie ist als View-Komponente des Gesamtsystems zu verstehen und besteht primär aus Kommunikation und Userinterface.

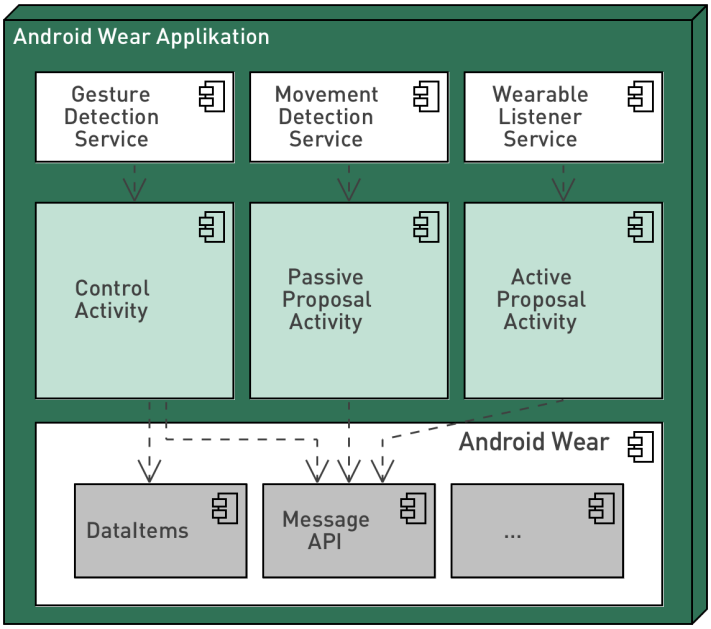


Abb. 4.21: Architektur der HAWS Wear Applikation

Die drei beschriebenen Szenarien finden sich in Form von drei verschiedenen *Activities* wieder, denen jeweils ein spezifischer Service als Ausgangspunkt gilt (vgl. Abb. 4.21).
Die *Control Activity*, welche die manuelle Steuerung repräsentiert, wird beispielsweise durch den *Gesture Detection Service* gestartet. Das ist ein Dienst, der systemweit auf die, in Kapitel 4.3 beschriebenen, Wischgesten in den aktiven Ecken wacht.
Die nächste Activity für passive Vorschläge wird durch den *Movement Detection Service* gestartet, welcher mit Hilfe des Inertialsensors die Rotationsbewegungen der Smartwatch überwacht. Die dritte und letzte Activity ist für die Darstellung aktiver Vorschläge des Servers zuständig. Sie werden per Push-Benachrichtigung an das Smartphone geleitet und anschließend als *Message* an die Smartwatch geschickt. Dort werden sie durch den *Wearable Listener Service* verarbeitet und führen zum Start der Anwendung.

4.5 Machine learning engine

Dieser Abschnitt wird einen kurzen Überblick über einen möglichen Lernprozess des beschriebenen Systems liefern. Eine vollständige Beschreibung und Ausarbeitung würde den Rahmen dieser Thesis allerdings übersteigen.

Die Machine learning engine (MLE) Komponente integriert sich ebenso wie der HAWS Server in die Blackboard Architektur des Smart Homes. Sie ist unabhängig vom Server selbst und kann auch durch andere Teilnehmer des Blackboards genutzt werden.
Sie abonniert Nachrichten, die den Kontext des Systems beschreiben. Das können Dinge sein wie die aktuelle Zeit, Wetter, Kalender-Daten, Temperatur und Bewegungs- oder Positionsdaten der Nutzer.

Die Emittlung von Positionsdaten wird beispielsweise durch die von Basener in seiner Masterarbeit beschriebene Architektur zur Aktivitätsentdeckung bewerkstelligt (vgl. Basener, 2013). Nach Analyse der Sensordaten werden mögliche Aktvitäten prognostiziert und anschließend auf das Blackboard publiziert, wo sie darauf aufbauende Dienste, wie die MLE weiterverarbeiten können. Basener integriert in seiner Arbeit die Positionsdaten in sog. *Functional Spaces* um bestimmte Gegenstände oder Möbel herum. Komponenten wie der MLE genügt es nämlich meist, grob granularere Positionsinformationen zu haben, wie „Person sitzt auf dem Sofa“ anstatt „Person ist auf Position 1.3992, -12,4232“. Die *Functional Spaces* stellen somit ein Mapping zwischen Positionsdaten und Objektzugehörigkeit dar.

Nun, da die MLE im Besitz von reichlich aktuellen Kontextinformationen ist, kann sie ihre Aufgabe erfüllen. Aus dem letzten Abschnitten geht hervor, dass sie eine Liste der Aktoren bereitstellen soll, die um die Wahrscheinlichkeitswerte angereichert ist, mit denen die jeweiligen Aktoren im aktuellen Kontext durch den Nutzer bedient werden könnten. Die aktuell vorhadene Aktorik kann die MLE, ähnlich wie der HAWS Server, ermitteln (vgl. Kap. 4.3.1).

Die für das Training benötigten Daten kann die MLE als Blackboard-Teilnehmer selbst regelmäßig ermitteln und kann versuchen, die Klassifikation damit zu optimieren. Denkbar ist auch, dass die MLE dem Nutzer erlaubt, die Ergebnisse zu bewerten, um die Qualität der Ergebnisse zu erhöhen.

Für die Ermittlung der Wahrscheinlichkeitswerte bieten sich diverse Lernverfahren an. Besonders verbreitet sind dabei *künstliche neuronale Netze* und *Support Vector Machines*. Beide werden im Kapitel Grundlagen bechrieben. Für jeden Aktor wird eine eigene Berechnung auf Basis der Trainingdaten durchgeführt. Während allerdings neuronale Netze sehr gut in der Lage sind, Wahrscheinlichkeitswerte in Form von Fließkommazahlen auszugeben, ist dies etwas gegen die Natur der Support Vector Machines. Je nach Implementation wären dafür feste Werte wie 0% und 100% denkbar. Aber auch zusätzliche Klassen zur Abstraktion verschiedener Wahrscheinlichkeiten zu nutzen, z.B. in 20er Schritten, stellt eine Möglichkeit dar.

Wie auch immer die MLE ihre Werte ermittelt, wichtig ist, dass sie schließlich in ein verständliches Format überführt und Interessierten zur Verfügung gestellt werden. Wie dies aussehen kann, wird im nächsten Abschnitt beschrieben.

4.6 Technische Kommunikation

Nahezu alle in dieser Arbeit beschriebenen Architekturkomponenten und Interaktionsmuster setzen stark auf technische Kommunikation in Form von Nachrichten. Abläufe und Schnittstellen wurden dabei bereits beschrieben, an dieser Stelle sollen die Inhalte einiger exemplarischer Nachrichten betrachtet werden.Dabei werden nur Nachrichten innerhalb des HAW Systems betrachtet, da Nachrichtenformate innerhalb der Smart Home Umgebung individuell unterschiedlich sind.

In allen Kommunikationskanälen wird dabei auf JSON Nachrichten gesetzt. JSON steht für *JavaScript Object Notation*. Es handelt sich dabei um ein kompaktes Datenformat, das weit verbreitet ist und von nahezu jeder größeren Programmiersprache oder Programmierumgebung in eigene Datenformate konvertiert werden kann.

Anstelle die gesamten Nachrichten zu betrachten, sollen aus Übersichtsgründen zunächst einzelne Teile vorgestellt werden, wie sie sich in den größeren, zusammengesetzten Nachrichten wiederfinden.
Ein Teil bzw. Objekt, das dabei besonders oft verwendet wird, ist der Aktor. Das Listing 4.1 zeigt, wie ein solcher Aktor in JSON beschrieben werden könnte.

```
{
  "actuator": {
    "id": "17425810_c9de17",
    "name": "My favorite light",
    "area": "Living room",
    "categorie": "Light",
    "types": {
      "0": "Actuator",
      "1": "Light",
      "2": "RGBLight",
      "3": "LW12Light"
    },
    "status": {
      "0": {
        "available": true
      },
      "1": {
        "on": true
      },
      "2": {
        "brightness": 0.45,
        "red": 0.2,
        "green": 0.5,
        "blue": 0.4
      },
      "3": {
        "routine": null
      }
    }
  }
}
```

Listing 4.1: Ein Aktor wird in JSON beschrieben

Neben den wenig überraschenden Kerndaten im Kopf der Nachricht, fallen die Schlüssel *types* und *status* auf. Ersterer Schlüssel bildet dabei die Klassenhierarchie aus Kapitel 4.4.1 ab (s.Z. 7-12). Auf diese Weise kann der Smartwatch-Client immer so viele Besonderheiten der Aktorik unstützen, wie er kennt. Sind die unteren Klassen der Hierarchie noch sehr häufig und standardmäßig unterstützt, ist die Klasse *LW12Light* (s.Z. 11)

vielleicht unbekannt für die aktuelle Version des Clients. Er kann diese Information einfach ignorieren und den Aktor wie ein *RGBLight* behandeln - der hochwertigste, ihm bekannte Typ.
Auch die Statuselemente des Schlüssels *status* sind modular aus den Status der verschiedenen Typen zusammengesetzt. Sie beschreiben den aktuellen Status des Aktors auf verschiedenen Ebenen. Die generierten, numerischen Schlüssel der gerade beschriebenen Typen werden hier wiederverwendet, um den jeweiligen Typen des Status zu identifizieren. Folglich gehört im Beispiel der Status *2* zum Typen *RGBLight* (vgl. Z. 10, 20ff).
Der Status wiederum passt individuell zum Typen und beinhaltet spezifische Unterschlüssel oder gar Strukturen. Im Fall des *RGBLight* sind dies Helligkeitswert und RGB-Werte. Der Status *3* zeigt dagegen zum Beispiel, dass ein LW12-RGBLight zusätzlich noch Routinen für automatische Farbwechsel unterstützt, derzeit aber keine dieser Routinen angewendet wird (vgl. Z. 26ff).

Im Folgenden soll die Antwort des HAWS Servers für die Abfrage vorhandener Aktoren gezeigt werden. Sie wird durch das erste Interaktionsmuster verwendet und durch die REST Route */v1/actuators* zurückgegeben.

```
1  {
2    "actuators": [
3      {
4        "id": "17425810_c9de17",
5        ...
6      },
7      {
8        "id": "7026430_8abe0a",
9        ....
10     }
11   ]
12 }
```

Listing 4.2: Rückgabe der Route /v1/actuators

Der Typ *Actuator* wird hier wiederverwendet und in die Aufzählung *actuators* integriert. Die für diesen Anwendungsfall wichtige Kategorie der Aktoren steckt in ihrer Beschreibung und kann bei der Auswertung der Nachricht zusammengeführt werden. Eine Sortierung nach der Kategorie ist also nicht nötig.

Als nächstes wird die Nachricht für das zweite Interaktionsmuster, die passiven Vorschläge, betrachtet. Sie wird durch den Aufruf der Route */v1/actuators/proposed* zurückgegeben.

```
1  {
2    "proposed_actuators": [
3      {
4        "probability": 0.34,
5        "actuator": {
6          "id": "19914615_576aba",
7          ...
8        },
9      },
10     {
11       "probability": 0.21,
12       "actuator": {
13         "id": "18025584_fa7a94",
14         ...
15       },
16     }
17   ]
18 }
```

Listing 4.3: Rückgabe der Route /v1/actuators/proposed

Das Beispiel 4.3 beinhaltet zwei vogeschlagene Aktoren. Wieder wird der Teil *actuator* aus Listing 4.1 verwendet. Diesmal allerdings wird er im Rahmen der *proposedi_ actuators* durch den Wahrscheinlichkeitswert der MLE angereichert (vgl. „probability“ Z.4). Dieser Wert dient der Smartwatch unter anderem als Sortierungs-Kriterium der darzustellenden Liste. Da er außerhalb des *actuator* Objektes ist, lässt sich die interne Klasse *Aktuator* stets ohne Anpassungen wiederverwenden. Der dabei genutzte Ansatz ist dem *separation of concerns* Architekturprinip zuzuordnen.

Schließlich sollen die Antworten des Clients an den Server, sowie das verbleibende, *aktive* Interaktionskonzept betrachtet werden. Da letzteres im Sinne der Wiederverwendbarkeit auf den Rückantworten der Clients aufbaut, werden diese im Folgenden zuerst beschrieben. Sie sind für alle der drei vorgestellten Interaktionskonzepte grundlegend identisch

und werden über die Route */v1/actuators/instructions* an den Server geleitet.

```
{
 "instructions":
 [
   {
    "actuator_id": "19914615_576aba",
    "type": "Light",
    "status":
    {
      "on": true
    }
   },
   {
    "actuator_id": "19914615_576aba",
    "type": "RGBLight",
    "status":
    {
      "red": 0.1,
      "green": 0.6,
      "blue": 1.0,
    }
   }
  ]
}
```

Listing 4.4: Steuerungsnachricht des Clients

Betrachtet man Listing 4.4, sieht man zunächst, dass der Primärschlüssel *instructions* ein Array ist. Durch diese Wahl kann eine sichere Reihenfolge gewährleistet werden. Warum dies nötig ist, wird ersichtlich, wenn man sich die Beispiele für eine solche Anweisung ansieht (vgl. Z. 5ff, 13ff). In beiden Nachrichten wird derselbe Aktor adressiert (s. *actuatorid*). Beide Instruktionen spezifizieren allerdings verschiedene Typen von Aktoren. Auf diese Weise können low-level Anweisungen, wie eine Lampe einzuschalten, ausgeführt werden, bevor Anweisungen höherer Ebenen ausgeführt werden.
Weiterhin gibt es für jede *instruction* den Schlüssel *status*, welcher die eigentliche Steuerungsanweisung enthält. Wie der Name vermuten lässt, sind diese Anweisungen weitestgehend identisch zum abrufbaren Status aus dem ersten Listing dieses Kapitels. Aller-

dings kann nicht jedes Attribut, das sich im Status eines Aktors findet, auch geändert werden. Zunächst ist davon auszugehen, dass Client weiß, welche Attribute veränderbar sind.
Möchte man das Protokoll für möglichst viele verschiedene Aktoren öffnen, wäre es denkbar anzugeben, ob ein Attribut änderbar ist, welchen Typ es hat und wie es dargestellt werden soll. Das Attribut *red* aus Zeile 17 wäre demnach beispielsweise eine änderbare Gleitkommazahl, die für den Nutzer als Slider dargestellt werden kann. Der Client wird damit noch mehr auf seine Rolle der View-Komponente im Gesamtsystem reduziert und enthält zunehmend weniger Logik. Neue Aktoren könnten ohne ein Update des Clients in das System eingespeist werden. Dieser Ansatz soll jedoch nicht weiter verfolgt werden, da er für sich alleine viele Fragestellungen aufwirft, die wiederum über den Rahmen dieser Thesis hinaus gehen würden. Ferner ist die Nutzung der Hierarchie von Aktorklassen bereits ein Ansatz, der in diese Richtung geht.

Schließlich sollen auch die Nachrichten des dritten, proaktiven Interaktionskonzeptes betrachtet werden. Besonders ist, dass diese nicht durch den Client abgerufen werden, sondern vom HAWS Server über den GCM Server an den Client geschickt werden. Der GCM Server sendet einige Metainformationen mit, die an dieser Stelle nicht weiter relevant sind. Das Listing 4.5 zeigt deshalb den eigentlichen Inhalt einer solchen Nachricht.

```
1  {
2    "proposed_instructions": [
3      {
4        "probability": 0.34,
5        "actuator": {
6          "id": "19914615_576aba",
7          ...
8        },
9        "instructions": [
10         {
11           "actuator_id": "19914615_576aba",
12           "type": "Light",
13           ...
14         }
15       ]
16     }
17   ]
```

```
18 }
```

Listing 4.5: Inhalt der GCM Nachricht für eine aktive Benachrichtigung durch den HAWS Server

Der Hauptschlüssel der Nachricht heißt *proposed_ instructions*. Seine Inhalte sind ähnlich zu denen, der zuvor vorgestellten *proposed_ actuators*, weisen aber jeweils zusätzlich den Schlüssel *instructions* auf (vgl. Z.9). Wie das Konzept dieses Interaktionsmuster in Kapitel 4.3.3 bereits deutlich machte, soll dem Nutzer für die aktiven Vorschläge nicht der aktuelle Status der Aktoren angezeigt werden, sondern der nach Ausführung der Instruktion durch den Server. Diese Besonderheit ist notwendig, da der Client im Falle ausbleibender Interaktion durch den Nutzer wissen muss, welche Instruktionen er an den Server senden soll.
Nach erfolgreicher Interaktion wird wieder eine *instruction* Nachricht an den Server geschickt, um die entsprechende Aktorik zu bedienen.

4.7 Fazit

In diesem Kapitel wurde beschrieben, wie die Anforderungen aus dem Kapitel Analyse in ein technisches Konzept und eine Software-Architektur übertragen wurden. Dabei lag das Hauptaugenmerk auf dem Finden verschiedener Lösungen für die zuvor beschriebenen Szenarien. Nicht vernachlässigt wurden dabei auch die Anforderungen an ein flexibles und erweiterbares Software-Design, sowie eine erfolgreiche Integration in ein realitätsnahes Smart Home System. Im folgenden Kapitel sollen die Ergebnisse näher betrachtet werden.

5 Auswertung

Im Folgenden soll der vorgestellte Entwurf ausgewertet werden. Dazu wird dieser zunächst mit den gestellten Anforderungen abgeglichen, anschließend wird die Wahrnehmung des Konzeptes durch Dritte betrachtet. Im letzten Abschnitt sollen alle verbesserungswürdigen Punkte zusammengefasst werden.

5.1 Anforderungsabgleich

Die im Kapitel Analyse ermittelten Anforderungen wurden im Kapitel Entwurf zunächst in ein Interaktionskonzept und anschließend in eine verteilte Softwarearchitektur überführt. Dabei wurde, wie gefordert, für jedes der vorgestellten Szenarien ein passendes Teilkonzept erdacht, das jeweils einen anderen Grad an Autonomie aufweist. Dem Nutzer steht frei, auf manuelle, teilautonome oder autonome Steuerungen zurückzugreifen. Die gewünschte Alltagskonformität lässt sich zwar allein aus diesem Konzept heraus nicht nachweisen, Wearables als Steuerungsgerät sind unter diesem Gesichtspunkt aber sehr vielversprechend. Sie sind kostengünstig, stets zur Hand und als Uhrenersatz sozial verträglich. Teure Sensorinstallationen sind für das Interaktionskonzept selbst nicht von Nöten.
Bei der vorgestellten Architektur wurde weiterhin auf eine geringe Abhängigkeit gegenüber dem Smart Home System geachtet. Sie erfüllt somit die Anforderungen nach Portierbarkeit und Adaptionsfähigkeit. Auch die Nachrichten der technischen Kommunikation sind als erweiterbar konzipiert, setzen auf die Wiederverwendbarkeit verschiedener Teile und beinhalten standardmäßig nur die nötigsten Informationen.

5.2 Resonanz

In diesem Abschnitt soll die Rückmeldung durch verschiedene Personen wiedergegeben und kommentiert werden. Vorweg sei erwähnt, dass die folgenden Aussagen aus keiner fundierten Studie resultieren. Sie sind das Ergebnis von Befragungen einiger weniger Personen - teils mit und teils ohne Hintergrundwissen in der Informationstechnik. Ihnen wurde das Konzept in allen Details erläutert und die prototypischen Entwicklungen präsentiert.

Das Konzept erreichte grundsätzlich sehr positive Resonanz. Gleich mehrere Teilnehmer wünschten sich allerdings eine Integration der, ansonsten von Android Wear gewohnten, Sprachsteuerung.
Wie mehrfach erwähnt, bietet Android Wear derzeit nur begrenzte Möglichkeiten, eigene Sprachbefehle in die Spracherkennung des Hauptsystems einzubinden. Denkbar ist allerdings eines oder durchaus gleich mehrere der drei Konzepte, um eine Spracherkennung zu erweitern. Sie wäre dabei als hilfreiche Ergänzung oder Alternative zur Touchscreen basierten Steuerung zu verstehen. Ein solch hybrides Konzept ist sehr vielversprechend, wäre allerdings zu umfangreich für den Rahmen dieser Arbeit gewesen, da die Thematik der Sprach-Auswertung wiederum viele neue Teilfragen aufwirft.

Weiterhin ist nennenswert, dass einige Nutzer das Konzept der *passiven Vorschläge* nicht allein als Anlaufstelle zum Auffinden gesuchter Aktoren verstehen. Sie versprechen sich zusätzlich Aktoren vorgeschlagen zu bekommen, an die sie gar nicht gedacht haben. Das System soll also nicht nur mutmaßen, was der Nutzer sich wünscht, sondern es soll auch vorschlagen, woran der Nutzer noch gar nicht gedacht hat. Ein interessanter Anwendungsfall wäre zum Beispiel, wenn man Gästen die Rechte einräumt an der Steuerung teilzuhaben. Sie kennen die jeweilige Aktorik gar nicht, können die Vorschläge aber sehr komfortabel nutzen, um sich im Haushalt zurecht zu finden oder dem Eigentümer Arbeit abzunehmen.

5.3 Verbesserungspotential

Nachfolgend soll das in diesem Kapitel bisher genannte Optimierungspotential durch eine Zusammenfassung der bereits innerhalb dieser Arbeit genannten Punkte ergänzt werden.

Das Konzept der passiven Vorschläge sieht vor, dem Nutzer Aktoren zu empfehlen, welche für ihn relevant sein könnten. Die nötigen Einstellungen bzw. Instruktionen für diese Aktoren muss er stets selbst konfigurieren. Ähnlich, wie beim proaktiven Konzept, könnten auch bereits Vorschläge für die jeweiligen Einstellungen der Aktorik erfolgen. Das derzeitige Konzept sieht dies aus Gründen der Übersichtlichkeit nicht vor. Gleichermaßen müsste man überlegen, dem Nutzer das Speichern und Abrufen dieser Einstellungen zu ermöglichen - gegebenenfalls sogar über mehrere Aktoren hinweg, als große *Settings* von Gerätekonfigurationen. All dies sind Komfortmerkmale, deren Umsetzung das eigentliche Konzept kaum tangiert.

Auch in technischer Hinsicht ergaben sich im Laufe der Arbeit Ideen für mögliche Verbesserungen. So wurde beispielsweise im Kapitel Technische Kommunikation bereits beschrieben, dass es aus Gründen der Erweiterbarkeit des Systems von Interesse wäre, das Format, der an den Client gesendeten Nachrichten, auf reine Angaben zur Darstellung zu reduzieren. Der Client wird damit noch mehr auf seine Rolle als View-Komponente im Gesamtsystem reduziert und enthält zunehmend weniger Logik. Das Einbinden neuer Komponenten könnte ohne ein notwendiges Update des Clients stattfinden.

6 Zusammenfassung und Ausblick

Die folgenden Abschnitte sollen die Ergebnisse dieser Arbeit zusammenfassen und abschließend einen Ausblick auf weitere Ansatzpunkte hinsichtlich Weiterentwicklung und Optimierung des entstandenen Systems geben.

6.1 Zusammenfassung

Im Rahmen dieser Arbeit wurde ein Interaktionskonzept zur Steuerung von lernenden Smart Home Umgebungen konzipiert und softwaretechnisch umgesetzt. Diese exemplarische Umsetzung soll musterhaft veranschaulichen, welchen Problemen bei der Interaktion mit lernenden Systemen begegnet werden muss. Es wurde deshalb nicht nur ein Einziges, sondern gleich drei Teilkonzepte mit einem steigenden Grad an Autonomie des Systems entworfen. Diese Konzepte sollen sich gegenseitig ergänzen. Dem Nutzer steht es so in weiten Teilen frei, wie er mit dem System in Kontakt tritt.

Die Nutzung von Smartwatches als zentrales Instrument der Interaktion macht das Vorhaben kostengünstig und durch hohes Absatzwachstum auch pragmatisch. Aufgrund der direkten Anwendung am Körper stellen sie im Vergleich zu Smartphones und Tablets ein unmittelbares und alltagstaugliches Eingabegerät dar.

Die vorgestellte Architektur ist im Ergebnis kompakt und anpassungsfähig. Sie soll sich mit wenig Aufwand auf verschiedene Systeme portieren lassen und kann sehr variabel mit diversen denkbaren Lernverfahren betrieben werden. Weiterhin ist sie bestens zur Unterstützung des entwickelten Konzeptes geeignet.

6.2 Ausblick

Filmemacher und Zukunftsforscher sind sich einig, dass es auf lange Sicht viele innovative Konzepte in unserer täglichen Interaktion mit *smart environments* geben wird.

Es wurde gezeigt, dass in den nächsten Jahren mit einer stark wachsenden Verbreitung von Wearables zu rechnen ist (s. Kap. 2.1). Damit verbunden sind üblicherweise auch Verbesserungen an der eingesetzten Hardware. Neue Sensoren werden hinzu kommen, Bestehende verbessert werden. Neue Medien, wie etwa Smart Glasses oder die eingangs genannten Kontaktlinsen mit integriertem Display, werden sich erschließen. Auch für intelligente Kleidung wird in absehbarer Zeit ein Markt entstehen.

Alles folgt der Evolution des Computers, d.h. es gibt zunehmend mehr, immer besser in unsere Umgebung integrierte Geräte (s. Kap. 2.2). Eine zukünftige Herausforderung für Smart Home Umgebungen wird daher sein, mit immer mehr Geräten kommunizieren zu müssen. Der Anwender wird nicht nur in der Lage sein, die Steuerung seiner Umgebung über seine Smartwatch durchzuführen, sondern ebenfalls über seine Jacke, seine Brille, diverse Möbel und andere Gegenstände. Der Ort der Interaktion wird immer belangloser. Das Ergebnis dieser Arbeit hat mit Android Wear eine flexible Basis, mit der viele mit Display ausgestattete Wearables nutzbar erscheinen. Dennoch wird es nicht bei der Bedienung von Touchscreens bleiben. Die Spracherkennung zum Beispiel hat in den letzten Jahren große technische Fortschritte gemacht und wird daher auch weiterhin an Verbreitung gewinnen.

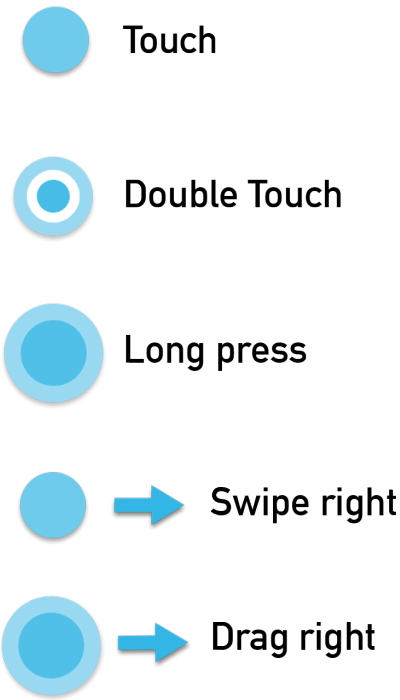
Was herausgestellt werden soll ist, dass keines der Konzepte oder Geräte kurzfristig gesehen eine alleinige Lösung darstellen wird. Sie stehen nicht zwingend in direkter Konkurrenz miteinander. Vielmehr werden sie sich gegenseitig ergänzen und den Anwendern gemäß der Idee des *ubiquitous computing* sehr viele Möglichkeiten zur Interaktion geben. Alle Konzepte zusammen reduzieren die nötige Interaktion für den Anwender auf ein Minimum und eröffnen ihm die Freiheit, sich dem gerade passenden Gerät zu bedienen.

Der hier vorgestellte Ansatz zeichnet sich besonders durch den zuvor beschriebenen Pragmatismus aus. Es ist ein solides Konzept, wie es in *unmittelbarer* Zukunft sinnvoll eingesetzt werden könnte. Es erfüllt bereits Kernvoraussetzungen, wie sie in Anbetracht der oben beschriebenen Entwicklung notwendig sind. Dem Nutzer wird Flexibilität in der Wahl über seine Interaktionswege gelassen, während die Implementation im Hintergrund durch geringe Anpassung mit verschiedenen Systemen zusammenarbeitet. Das

Konzept erhebt nicht den Anspruch für lange Zeit als besonders innovativ zu gelten, wenngleich es Potential für Erweiterungen bietet. So lange aber lernende Systeme fehlbar sind und die hier vorgestellten, verwandten Arbeiten durch verschiedenste Faktoren im Konzept-Status verweilen, hat dieser schlichte, aber effiziente Ansatz eine fundierte Rechtfertigung für seinen Einsatz. Technische Anpassungen sind dabei immer denkbar, das hier entwickelte Grundkonzept wird aber unter Berücksichtigung der oben beschriebenen Koexistenz vieler Interaktionskonzepte seine Gültigkeit behalten.

Anhang

Abbildungs Legende



Literaturverzeichnis

[Augusto u. a. 2010] Augusto, JuanCarlos ; Nakashima, Hideyuki ; Aghajan, Hamid: Ambient Intelligence and Smart Environments: A State of the Art. In: Nakashima, Hideyuki (Hrsg.) ; Aghajan, Hamid (Hrsg.) ; Augusto, JuanCarlos (Hrsg.): *Handbook of Ambient Intelligence and Smart Environments*. Springer US, 2010. – ISBN 978–0–387–93807–3, S. 3–31

[Basener 2013] Basener, Andreas: *Clusterbildung von Sensordaten aus einem Smarthome zur Aktivitäteninterpretation*, HAW Hamburg, Diplomarbeit, 2013

[Bitkom 2014] Bitkom: *Bekanntheit und Nutzungsbereitschaft von Smart Glasses und Smartwatches in Deutschland*. http://de.statista.com/infografik/2690/bekanntheit-und-nutzungsbereitschaft-von-smart-glasses-und-smartwatches-in-deutschland/. Version: 2014, Abruf: 23.10.2014

[Cleve u. Lämmel 2014] Cleve, J. ; Lämmel, U.: *Data Mining*. Gruyter, de Oldenbourg, 2014. – ISBN 9783486713916

[Costa u. a. 2009] Costa, Ricardo ; Carneiro, Davide ; Novais, Paulo ; Lima, Luís: Ambient Assisted Living. In: Corchado, Juan M. (Hrsg.) ; Tapia, Dante I. (Hrsg.) ; Bravo, José (Hrsg.): *3rd Symposium of Ubiquitous Computing and Ambient Intelligence 2008*. Springer Berlin Heidelberg, Januar 2009 (Advances in Soft Computing 51). – ISBN 978–3–540–85867–6, S. 86–94

[Eichler 2014] Eichler, Tobias: *Agentenbasierte Middleware zur Entwicklerunterstützung in einem Smart-Home-Labor*, HAW Hamburg, Diplomarbeit, Oktober 2014

[El-Sharif 2014a] El-Sharif, Yasmin: DeepMind: Google kauft Start-up für künstliche Intelligenz. (2014), Januar. http://www.spiegel.de/wirtschaft/unternehmen/

google-kauft-start-up-deepmind-fuer-kuenstliche-intelligenz-a-945668.html, Abruf: 01.09.2014

[El-Sharif 2014b] El-Sharif, Yasmin: Thermostathersteller: Google kauft Nest Labs für 3,2 Milliarden Dollar. (2014), Januar. http://www.spiegel.de/wirtschaft/unternehmen/google-kauft-nest-labs-fuer-3-2-milliarden-dollar-a-943362.html, Abruf: 01.09.2014

[Ellenberg 2011] Ellenberg, Jens: *Ontologiebasierte Aktivitätserkennung im Smart Home Kontext*, HAW Hamburg, Diplomarbeit, 2011

[Ellenberg u. a. 2011] Ellenberg, Jens ; Karstaedt, Bastian ; Voskuhl, Sören ; Luck, Kai v. ; Wendholt, Birgit: An Environment for Context-Aware Applications in Smart Homes. In: *International Conference on Indoor Positioning and Indoor Navigation*, IEEE, September 2011. – ISBN 978–1–4577–1803–8

[Erman u. a. 1980] Erman, Lee D. ; Hayes-Roth, Frederick ; Lesser, Victor R. ; Reddy, D. R.: The Hearsay-II Speech-Understanding System: Integrating Knowledge to Resolve Uncertainty. In: *ACM Comput. Surv.* 12 (1980), Juni, Nr. 2, S. 213–253

[Ertel 2011] Ertel, Wolfgang: *Introduction to Artificial Intelligence*. 2. überarb. Aufl. 2011. Springer-Verlag, 2011. – ISBN 978–0–85729–298–8

[Fielding 2000] Fielding, Roy T.: *REST: Architectural Styles and the Design of Network-based Software Architectures*, University of California, Irvine, Dissertation, 2000

[Google Inc. 2014a] Google Inc.: Android Wear. (2014). https://developer.android.com/wear/index.html, Abruf: 13.09.2014

[Google Inc. 2014b] Google Inc.: Building Apps for Wearables. (2014). https://developer.android.com/training/building-wearables.html, Abruf: 13.09.2014

[Google Inc. 2014c] Google Inc.: Design Principles for Android Wear. (2014). https://developer.android.com/design/wear/principles.html, Abruf: 13.09.2014

[Google Inc. 2014d] Google Inc.: Google Cloud Messaging Overview. (2014). https://developer.android.com/google/gcm/gcm.html, Abruf: 03.11.2014

[Kim u. a. 2012] Kim, David ; Hilliges, Otmar ; Izadi, Shahram ; Butler, Alex D. ; Chen, Jiawen ; Oikonomidis, Iason ; Olivier, Patrick: Digits: Freehand 3D Interactions Anywhere Using a Wrist-worn Gloveless Sensor. In: *Proceedings of the 25th*

Annual ACM Symposium on User Interface Software and Technology. New York, NY, USA : ACM, 2012 (UIST '12). – ISBN 978–1–4503–1580–7, S. 167–176

[Kolbaja 2012] Kolbaja, Viktor: *Generischer Smart Home Controller auf Android OS*, HAW Hamburg, Diplomarbeit, 2012

[LaViola 2013] LaViola, Joseph J.: 3D Gestural Interaction: The State of the Field. In: *ISRN Artificial Intelligence* Bd. 2013, 2013, S. 18

[LG Electronics 2014] LG Electronics: *G Watch - Ihr smarter Dauerbegleiter*. http://www.lg.com/de/wearables/lg-G-Watch. Version: 2014, Abruf: 01.10.2014

[Luck u. a. 2010] Luck, Prof. Dr. Kai von ; Klemke, Prof. Dr. Gunter ; Gregor, S. ; Rahimi, M. ; Vogt, M.: Living place hamburg – a place for concepts of it based modern living / Hamburg University of Applied Sciences. 2010. – Forschungsbericht

[Microsoft 2014] Microsoft: *Torque*. https://play.google.com/store/apps/details?id=com.microsoft.bing.torque. Version: Oktober 2014, Abruf: 27.10.2014

[Mistry u. Maes 2009] Mistry, Pranav ; Maes, Pattie: SixthSense: A Wearable Gestural Interface. In: *ACM SIGGRAPH ASIA 2009 Sketches*. New York, NY, USA : ACM, Dezember 2009 (SIGGRAPH ASIA '09), S. 11:1

[Neßelrath u. a. 2011] Nesselrath, Robert ; Lu, Chensheng ; Schulz, Christian ; Frey, Jochen ; Alexandersson, Jan: A Gesture Based System for Context – Sensitive Interaction with Smart Homes. In: *Ambient Assisted Living*. Springer Berlin Heidelberg, 2011. – ISBN 978–3–642–18166–5, S. 209–219

[Onnebrink u. Kuchen 2004] Onnebrink, Daniel ; Kuchen, Prof. Dr. H.: *Architekturmuster*. Seminar Ausarbeitung, Universität Münster, 2004

[Open Handset Alliance 2014] Open Handset Alliance: *Android Low-Level System Architecture*. https://source.android.com/devices/index.html. Version: 2014, Abruf: 28.10.2014

[Pai 2014] Pai, Aditi: ABI: 90M wearable devices to ship in 2014. (2014), February. http://mobihealthnews.com/29532/abi-90m-wearable-devices-to-ship-in-2014/, Abruf: 16.09.2014

[Pan u. a. 2010] Pan, Gang ; Wu, Jiahui ; Zhang, Daqing ; Wu, Zhaohui ; Yang, Yingchun ; Li, Shijian: GeeAir: a universal multimodal remote control device for

home appliances. In: *Personal and Ubiquitous Computing* 14 (2010), Nr. 8, S. 723–735. – ISSN 1617–4909

[Pearson 2013] Pearson, Ian: *You Tomorrow: The Future of Humanity, Gender, Everyday Life, Careers, Belongings and Surroundings*. Createspace Independent Pub, 2013. – ISBN 9781491278260

[Rahman u. a. 2009] Rahman, Asm ; Hossain, Anwar ; Parra, Jorge ; El Saddik, Abdulmotaleb: Motion-path Based Gesture Interaction with Smart Home Services. In: *Proceedings of the 17th ACM International Conference on Multimedia*. New York, NY, USA : ACM, 2009 (2009). – ISBN 978–1–60558–608–3, S. 761–764

[Spiegel 2011] Spiegel, Manager Magazin: *Wie viele Armbanduhren besitzen Sie, die Sie zumindest gelegentlich tragen?* http://de.statista.com/statistik/daten/studie/178379/umfrage/anzahl-der-eigenen-armbanduhren/. Version: 2011, Abruf: 23.10.2014

[Strategy Analytics 2014] Strategy Analytics: *Android Captured Record 85 Percent Share of Global Smartphone Shipments in Q2 2014*. http://blogs.strategyanalytics.com/WSS/post/2014/07/30/Android-Captured-Record-85-Percent-Share-of-Global-Smartphone-Shipments-in-Q2-2014.aspx. Version: 2014, Abruf: 03.11.2014

[Taylor Otwell 2014] Taylor Otwell: Laravel Documentation. (2014). http://laravel.com/docs/4.2, Abruf: 05.11.2014

[Torborg u. Simpson 2013] Torborg, Scott ; Simpson, Star: Google Glass Teardown. (2013), Juni. http://www.catwig.com/google-glass-teardown/, Abruf: 14.09.2014

[Waldner 2013] Waldner, Jean-Baptiste: *Nanocomputers and Swarm Intelligence*. New York : John Wiley & Sons, 2013. – ISBN 978–1–118–62415–9

[Witt 2011] Witt, Kristoffer: *Kontextabhängige multimodale Interaktion mit Schwerpunkt Spracherkennung im Smart-Home Umfeld*, HAW Hamburg, Diplomarbeit, 2011

Versicherung der Selbstständigkeit

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne fremde Hilfe selbständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, 10. November 2014 Björn Baltbardis