Technology
Arts Sciences
TH Köln

Entwicklungsprojekt interaktive Systeme

Wintersemester 2018/2019

Modelle und Modellierungsbegründungen Meet & Remind

Dozenten

Prof. Dr. Gerhard Hartmann

Prof. Dr. Kristian Fischer

Betreuer

Daniela Reschke

Markus Alterauge

Team

Johanna Mayer

Julian Schoemaker

Inhaltsverzeichnis

Einleitung

In diesem Dokument geht es um die Modelle und die Modellierungsbegründungen zum im Konzept beschriebenen System Meet & Remind. Es befasst sich mit dem gewählten Vorgehensmodell der Software-Entwicklung und beantwortet die Fragen zur Erstellung des User Interfaces. Desweiteren wird auf die Umsetzung der Systemkomponenten eingegangen. Dabei ist es wichtig die im Vorgehensmodell gezogenen Erkenntnisse zu beachten und für die Begründung mit einzubeziehen.

Vorgehensmodell

Zuerst wird auf die Wahl des Vorgehensmodells eingegangen, die bereits im Konzept begonnen wurde. Durch weitere Begründungen in den nächsten Abschnitten wird ersichtlich, warum die genannten Vorgehensmodelle eine gute Grundlage für die Modellierung und Implementierung des Systems Meet & Remind bieten.

DIN EN ISO 9241-210

Diese Norm befasst sich mit dem menschzentrierten Gestaltungsprozess. Dies ist ein Vorgehen, das bei der Entwicklung von interaktiven Systemen eingesetzt werden kann. Dabei soll sich unter Anderem auf die Verwendung des Systems und auf die Gebrauchstauglichkeit konzentriert werden.
Dieses Vorgehen passt zu dem zu entwickelnden System, da der Benutzer, und auch die Stakeholder im Allgemeinen, eine (im Alltag) unterstützende Anwendung erhalten sollen und ihre Aufgaben (und Ziele) im Bezug zum System effektiv, effizient und zufriedenstellend erledigen können sollen.

In Abbildung 1 wird der Prozess der menschzentrierten Gestaltungsaktivitäten gezeigt. Zuerst muss der Nutzungskontext verstanden und beschrieben werden. Dies geschah für “Meet & Remind” bereits im Konzept. Hier wurden die Domäne recherchiert, Probleme beschrieben, Ursachen gesucht und Stakeholder aufgelistet. Somit wurde der Nutzungskontext genauer untersucht, jedoch ist dieser Teil ein iterativer Prozess und wird ggf. nochmals aufgegriffen und überarbeitet.

Wenn der Nutzungskontext verstanden wurde folgt die Spezifizierung der Anforderungen. Hierbei wird in funktionale, organisationale und technische Anforderungen unterschieden. Diese sind Voraussetzung für den weiteren Prozess und den ersten Gestaltungslösungen.

Bei der Entwicklung von Gestaltungslösungen wird ein weiteres Vorgehensmodell, das “Usage Centered Design” angewendet, um den Verwendungszweck des Systems besser zu verstehen und damit die Benutzung des Systems zu vereinfachen.

Am Ende des Gestaltungsprozess steht die Evaluierung. Hier muss die Perspektive der Benutzer eingenommen werden oder ggf. Tests mit ihnen durchgeführt werden, um zu entscheiden, welche Teilaspekte des Prozess iteriert werden sollten, um am Ende ein gebrauchstaugliches Sytsem zu besitzen.

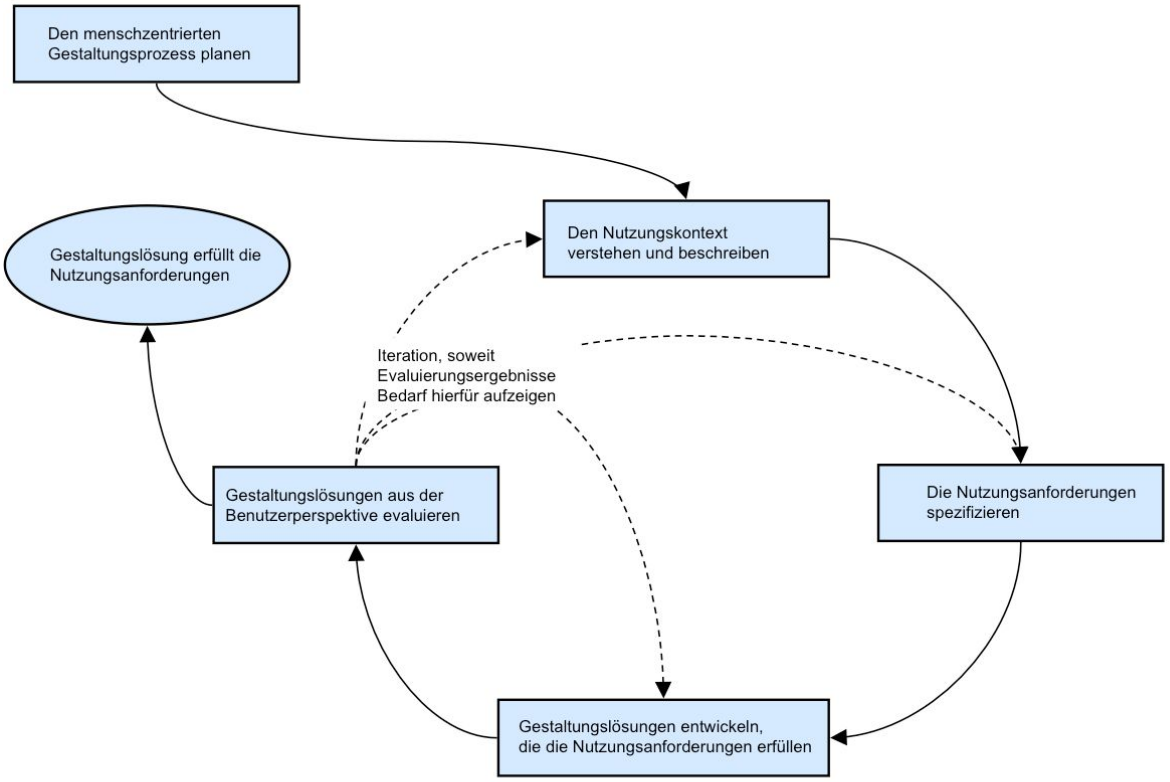


Abb. 1: aus DIN EN ISO 9241-210: Wechselseitige Abhängigkeit menschzentrierter Gestaltungsaktivitäten

Usage Centered Design

Beschreibung

Das Vorgehensmodell des Usage Centered Design basiert auf der modellgetriebenen Entwicklung in der Software-Entwicklung. Aus dem Namen lässt sich erkennen, dass hierbei vor allem die Einsatzmöglichkeit und Benutzung des Systems im Vordergrund steht. Die drei Modelle Role Model, Task Model, Operational Model und Content Model bilden die Grundlage für die Erstellung des Visual Design im Implementational Model. Außerdem liefern sie Erkenntnisse für die fachlichen Datenmodelle und beeinflussen damit nachhaltig die Systemarchitektur.

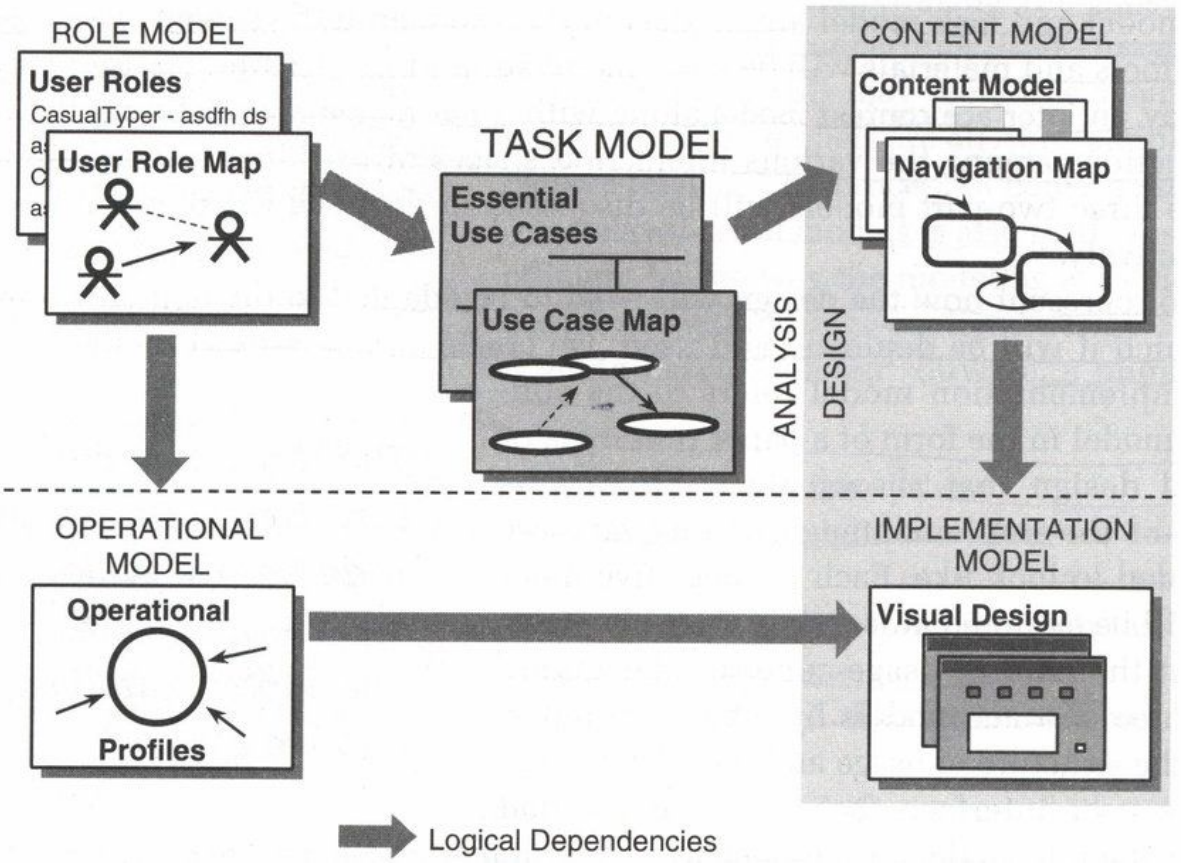


Abb. 2: aus Software for use: Essential models

Begründung

Im Konzept wurde bereits kurz beschrieben, warum dieses Vorgehensmodell positiv zur Entwicklung von Meet & Remind beitragen würde. Zudem lässt sich aus dem vorhergehenden Abschnitt erkennen, dass das Usage Centered Design für den Entwicklungsprozess förderlich ist.
Hier wurde auch das Scenario Based Usability Engineering als passend genannt, welches jedoch bei der weiteren Betrachtung der Modellierung als weniger geeignet eingeschätzt wurde. Die erarbeiteten Problemszenarien für dieses Vorgehensmodell wichen nicht weit genug voneinander ab, als dass man aus ihnen Schlüsse für das visuelle Design ziehen konnte.
Bei der ersten Stakeholderanalyse und dem Aufstellen der Kommunikationsmodelle konnte festgestellt werden, dass sich die Benutzer im Bezug auf das System nicht weit voneinander unterscheiden. Die Benutzer spielen also weniger eine Rolle für das System als die Art der Aufgaben, die sie mit dem System bearbeiten. Aus diesem Grund wurde auch das Vorgehensmodell des User Centered Design ausgeschlossen.

Einsatz

Beim Einsatz des Usage Centered Design wurde auf iterative Anwendung des Vorgehensmodells geachtet. Das bedeutet, dass alle Modellierungen mehrmals durchlaufen wurden und dabei Verbesserungen vorgenommen worden sind. Dadurch konnte sichergestellt werden, dass die Schlüsse aus späteren Modellierungen auch mit in der

Überlegungen für die vorhergehenden Modellierungen genommen wurden. Zur besseren Übersicht wurden die Iterationen in diesem Dokument zusammengefasst.

Collaborative Requirements Dialog

Der Prozess des Vorgehensmodell des Usage Centered Design beginnt mit dem Collaborative Requirements Dialog, der die Erfordernisse und Anforderungen der Stakeholder an das System aufzeigt. Hierbei wird sowohl die Sicht der Benutzer als auch der Entwickler und Administratoren des Systems beschrieben.

https://www.procontext.de/aktuelles/2012/06/nutzungskontext-erfordernisse-anforderungen-und-loesung-das-arbeitsmodell-des-usability-engineering.html

Erfordernisse

“Eine notwendige Voraussetzung, die es ermöglicht, den in einem Sachverhalt des Nutzungskontexts enthaltenen Zweck effizient zu erfüllen” [QUELLE https://www.dakks.de/sites/default/files/71_sd_2_007_leitfaden_usability_1.3_0.pdf]

Der Cash Flow Manager muss wissen, welche Rechnungen zu welchen Zeitpunkten bezahlt werden sollten, um das Saldo des Firmenkontos über Null zu halten. (Voraussetzung + Zweck)

1. Als Benutzer muss man alle verbundene Kontakte als Liste verfügbar haben, um Erinnerungen für den jeweiligen Kontakt erstellen zu können.
2. Als Benutzer muss man ein Textfeld verfügbar haben, um Erinnerungen zu erstellen.
3. Als Benutzer muss man an erstellte Erinnerungen beim Aufeinandertreffen des jeweiligen Kontakts erinnert werden können, um diese beim Gesprächspartner ansprechen zu können.
4. Als Benutzer muss man Labels zu Erinnerungen hinzufügen können, um passende Gesprächsthemen vorgeschlagen zu bekommen.
5. Als Benutzer muss man mobil und ohne dauerhafte Internetverbindung für erstellte Erinnerungen benachrichtigt werden, um die Sicherheit zu haben, keine Erinnerung zu verpassen.

Anforderungen

Die Anforderungen werden von den Erfordernissen abgeleitet und werden im späteren Verlauf eingesetzt um bei der Evaluierung des System zu prüfen, ob die gestellten Anforderungen auch erfüllt wurden.
“Erfordernis oder Erwartung, das oder die festgelegt, üblicherweise vorausgesetzt oder verpflichtend ist. ” [QUELLE https://www.dakks.de/sites/default/files/71_sd_2_007_leitfaden_usability_1.3_0.pdf]
Anforderungen können bspw. in funktionale, organisationale, qualitative und technische Anforderungen unterteilt werden. Dabei ist bei den funktionalen Anforderungen das folgende Schema zu beachten:

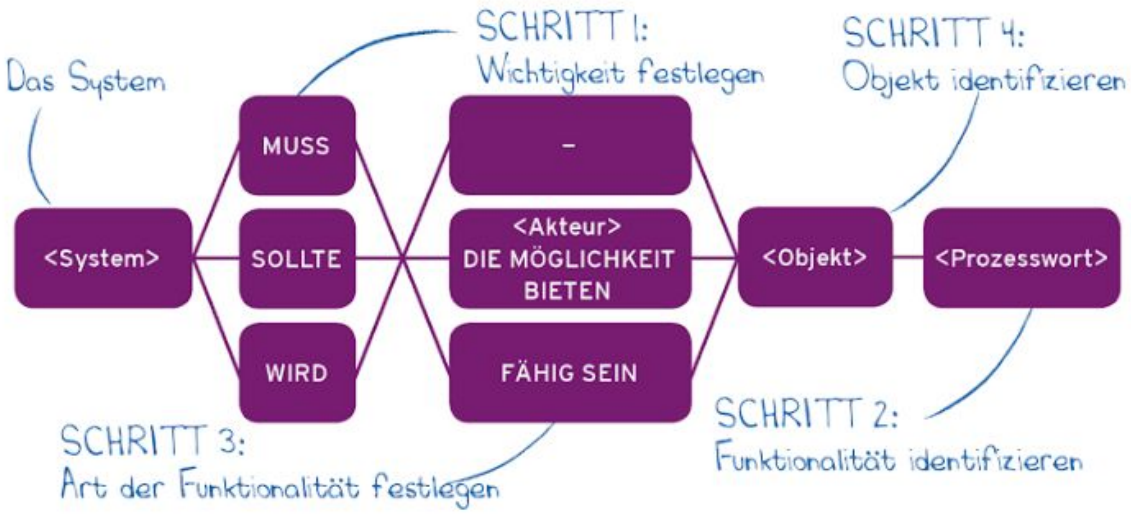


Abb. 3: Anforderungsschablone

Hierbei ist die Unterscheidung von Muss, Sollte und Wird wichtig. “Muss” sind Mindestanforderungen an das System, “Soll” bedeutet, dass die Funktion nicht zwingend notwendig ist und “Wird” stellt einen Ausblick dar.

Außerdem müssen bei den Anforderungen die Qualitätsanforderungen beachtet werden:
Vollständig, wenn sie umfangreich Information liefern.
Notwendig ist eine Anforderung, wenn sie zur Zielerreichung beiträgt.
Atomar, wenn eine Anforderungsformulierung ein Vollverb besitzt.
Verfolgbar, wenn die Anforderungen nummeriert sind.
Technisch lösungsneutral, damit im Vorfeld keine Einschränkungen gemacht werden.
Realisierbar, also innerhalb der möglichen Grenzen.
Konsistent, also widerspruchsfrei.
Eindeutig, also nicht missverständlich.
Prüfbar, also messbar oder es können Tests durchgeführt werden.

Funktionale Anforderungen

1. Das System muss dem Benutzer die Möglichkeit bieten sich mit einem anderen Benutzer koppeln zu können.
   1.1. Das System muss eine Liste aller verfügbaren Geräte zum koppeln anzeigen.
   1.2. Das System muss dem Benutzer die Möglichkeit bieten ein Gerät zum Koppeln auszuwählen.
   1.3. Das System muss auf beiden Geräten einen Dialog starten, ob die Kopplung erwünscht ist.
2. Das System muss fähig sein zu erkennen, wenn sich zwei bereits gekoppelte Geräte in der Nähe befinden.
3. Das System muss eine Übersicht aller bereits gekoppelten Geräte bieten.
4. Das System muss dem Benutzer die Möglichkeit bieten ein bereits gekoppeltes Gerät auszuwählen, um für dieses Gerät eine neue Erinnerung zu erstellen.
5. Das System muss dem Benutzer die Möglichkeit bieten Erinnerungen zu erstellen.
   5.1. Das System muss mindestens ein Textfeld zur Erstellung bieten.

    5.2. Das System muss die Möglichkeit bieten, Themen-Labels zu der jeweiligen Erinnerung auszuwählen.
6. Das System muss fähig sein dem Ersteller die jeweilige Erinnerung als Benachrichtigung anzuzeigen, sobald er sich der zur Erinnerung zugehörigen Person genähert hat.
7. Das System muss erkennen, wenn die Gesprächspartner in einem gewissen Zeitraum dieselben Themen-Labels gesetzt haben.
8. Das System muss bei gleichen Themen-Labels beiden Gesprächspartnern denselben informativen Text (Artikel etc.) zur Verfügung stellen.

Organisationale Anforderungen

1. Das System muss im Play Store für alle Android Geräte ab Version 7.0 verfügbar sein.
2. Das System muss sowohl auf Tablet als auch auf Smartphones nutzbar sein.

Qualitative Anforderungen

1. Das System muss problemlos das Koppeln ermöglichen.
2. Das System muss in jeder Situation Reaktionszeiten von unter einer Sekunde haben.
3. Das System darf nicht unerwartet abstürzen oder einfrieren.
4. Das System muss die Datensicherheit beachten.
5. Das System soll gebrauchstauglich im Sinne der Benutzer sein.

Anforderungen an die Benutzungsoberfläche

1. Das System muss auch für die Zielgruppe der älteren vergesslichen Leute gut lesbar sein.

Technische Anforderungen

1. Das System muss einen Zuweisungs-Algorithmus der Themen-Labels auf Serverseite besitzen.
2. Der Server soll eine REST Schnittstelle besitzen.
3. Der Server soll Zugriff auf eine externe Datenhaltung besitzen.
    3.1. Die externe Datenhaltung soll über Googles Firebase realisiert werden.
    3.2. Die externe Datenhaltung soll unabhängig vom Client sein, sodass bei Umstellung des externen Datensystems nur die Serverseite angepasst werden muss.
4. Die Kommunikation zwischen Client-App und Server, sowie zwischen Server und externer Datenhaltung, muss sicher sein.

Fazit aus Anforderungen

Die Erfordernisse und Anforderungen geben an, was vorausgesetzt wird, damit der Benutzer seine Ziele mit dem System bewerkstelligen kann. Sie bilden die Grundlage für die weiteren Modellierungen und liefern einen tieferen Einblick über den Nutzungskontext des Systems.

Aus den Anforderungen können zudem die ersten Schlüsse für Systemarchitektur und Datenstruktur gezogen werden, die durch die weiteren Schritte des Vorgehensmodells überprüft werden.

Domain Model

Bei der Vorgehensweise des Usage Centered Design wird die Prozess des Domain Model für die weitere Modellierungsbegründung genutzt. Hier kommen unsere Domänenrecherchen aus dem Konzept zu tragen.

Domänenmodell

Um einen besseren Überblick für den Modellierungsschritt Role Model zu erhalten, haben wir uns das Domänenmodell zur Domäne “Gespräch” wieder einbezogen. Innerhalb der Erarbeitung des Konzeptes wurde das Domänenmodell bereits iteriert.

//begründete Iteration
Eine weitere ausführliche Iteration ist unserer Meinung nach nicht notwendig, da die Domäne wie in der Abbildung gezeigt bereits genügend Umfang bietet, um ein verteiltes System mit Anwendungslogik zu entwickeln.

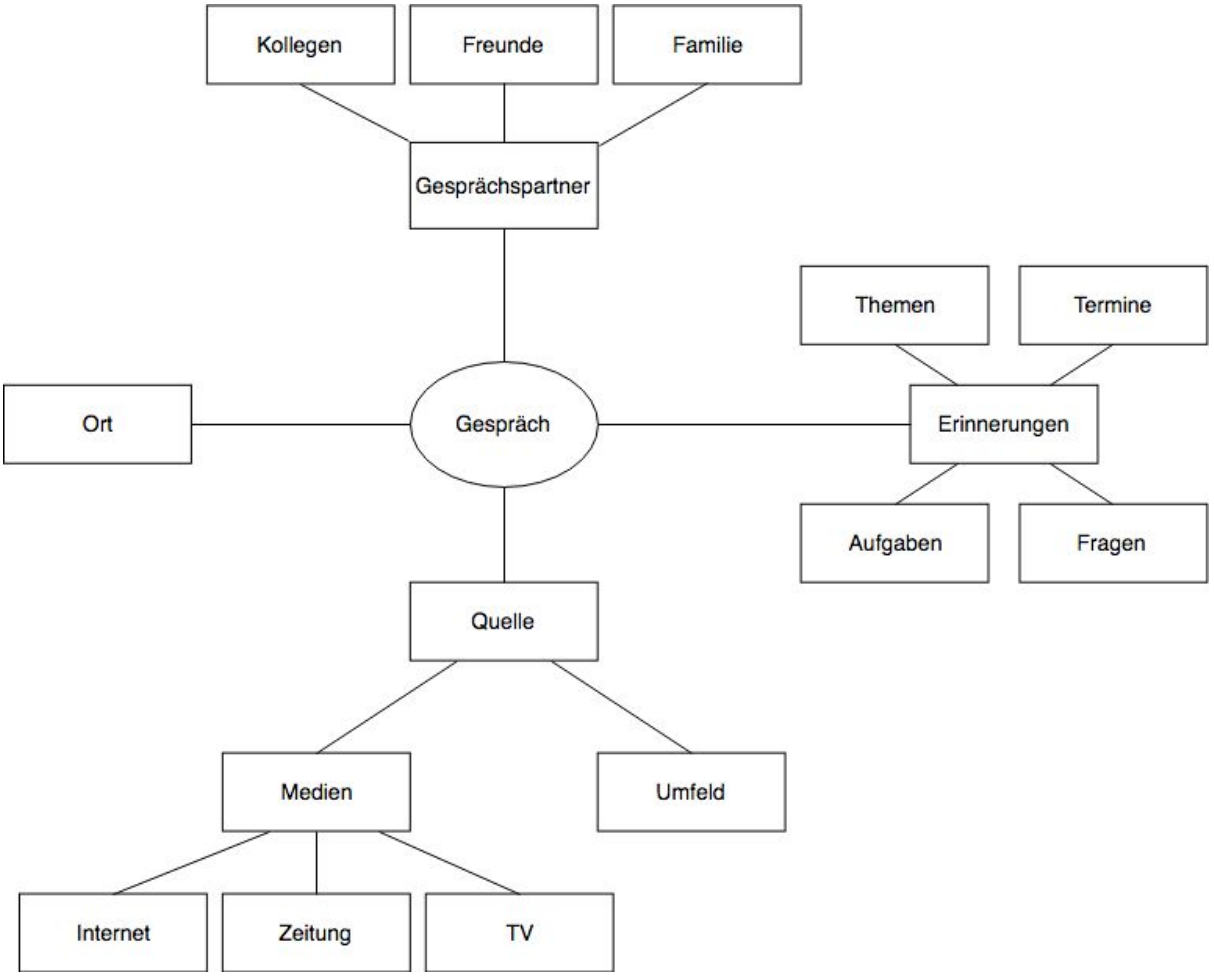
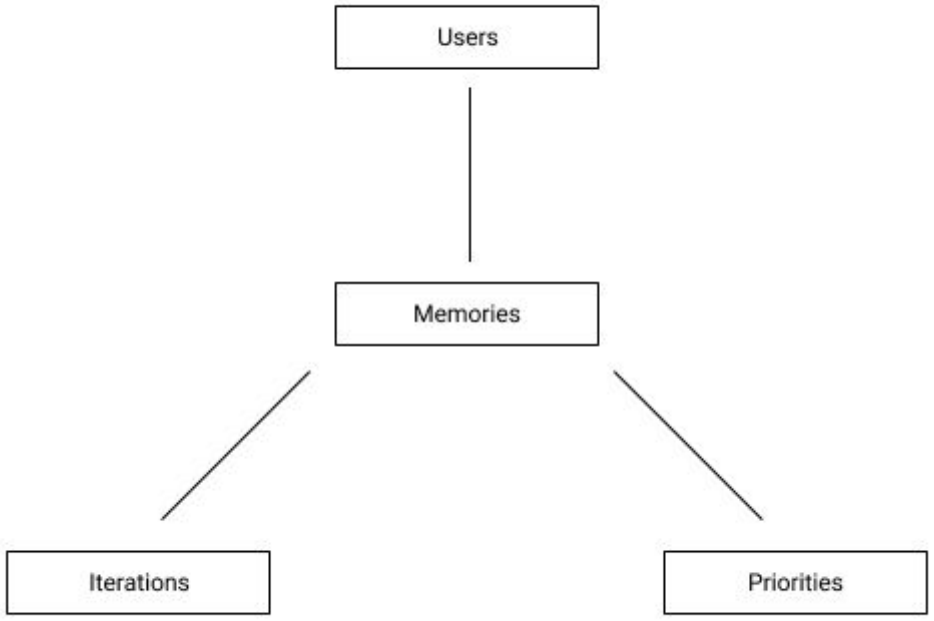


Abb. 3: Domänenmodell zur Domäne “Gespräch”

Klassendiagramm

//Abgeleitetes Modell vom Domänenmodell
// Ungefähr das, was wir im Konzept zunächst als Domänenmodell abgeben wollten



Stakeholderanalyse

Desweiteren bringt uns die Stakeholderanalyse aus dem Konzept Einblicke in die Rollen des Systems. Zusammen mit den Anforderungen und dem Domänenmodell sind hiermit die Grundkenntnisse über die Domäne und die Beziehung zum System vorbereitet um sie im Role Model auszuarbeiten.

Tabelle: Stakeholderanalyse

Bezeichnung	Beschreibung	Beziehung zum System	Priorität für das Projekt
Menschen im Arbeitsumfeld	Benutzer die viel unterwegs sind und dadurch oft Kontakten begegnen. Sowohl innerhalb eines Unternehmens, als auch außerhalb (Außendienstmitarbeiter, Berater).	Interesse	sehr hoch
Menschen mit vielen	Benutzer, die durch einen	Interesse	niedrig

Kontakten	gewissen Umstand (Vermögen, Berühmtheit) viele Kontakte haben und diese Kontakte pflegen wollen.		
Gestresste Menschen	Benutzer, die aufgrund von privatem Umfeld (Eltern, Alleinstehende) oder hohen Verantwortungen (Geschäftsführer, Ärzte) im Alltag viele Dinge im Kopf haben müssen.	Interesse	sehr hoch
Vergessliche Menschen	Benutzer, die wegen ihrem Alter und einer Erkrankung (Demenz, Alzheimer)	Interesse	hoch
Schüchterne Menschen	Benutzer, denen es schwer fällt Gespräche aufzubauen, da sie keinen Einstieg, zum Beispiel in Form von Themengebieten kennen.	Interesse	mittel
Ersteller von Medienprodukten (Wissenschaftler, Journalisten usw.)	Personen, die durch das Veröffentlichen von Medien (z.B. Zeitungsartikel) Themen für Gespräche liefern.	Interesse	mittel
Alle	Datensicherheit: Daten müssen abgesichert sein. Wie Daten gehalten und benutzt werden muss transparent sein.	Anrecht	hoch

Fazit aus Domain Model

// Wenn uns das zur Argumentation reicht, genügt das. Also hier aufschreiben, dass unsere Recherche aus dem Konzept ausreicht und wir sie ja auch noch über die weitere Modellierungen erweitern.

Role Model

Für das Usage Centered Design liegt der Fokus eher auf dem Task Model, aber die Perspektive der Benutzer spielt trotzdem eine Rolle. Es ist wichtig, dass Entwickler und Benutzer in Interaktion treten, damit beide Perspektiven berücksichtigt werden. Nur so kann ein gutes System gewährleistet werden. Diese Perspektiven werden mit dem Role Model analysiert.

Das Role Model ist in simpler Form eine Liste von Benutzerrollen, beschrieben in Bedürfnisse, Interessen, Erwartungen, Benehmen und Verantwortlichkeiten.

Eine Benutzerrolle ist eine abstrakte Klasse, definiert durch eine bestimmte Beziehung zum System. Es wäre zu grob gesagt, dass sie eine Gruppe von Benutzern zusammenfassen. Denn User Roles sind eine Abstraktion und stellen keine echten Benutzer dar. [QUELLE]

User Roles

User Roles Vorgehen

Das Vorgehen bei User Roles sieht folgendermaßen aus:

1. Compile: Zuerst wird Brainstorming verwendet, wobei dabei nicht diskutiert werden darf. Alle Ideen sind akzeptiert und es werden Aspekte der User Roles gesammelt, also einzelne Eigenschaften, Bedürfnisse etc.
2. Organize: Der erste Schritt wurde sorgfältig überprüft und nun geht es darum die gesammelten Aspekte zu sortieren und mit aussagekräftigen Titeln zu gruppieren.
3. Detail: Da es nun eine Gruppierung gibt und die entsprechenden Merkmale zugeordnet wurden, können diese nun verfeinert werden und Lücken geschlossen werden.
4. Refine: Nun wird das organisierte und detaillierte Model verfeinert und komplettiert. Hier wird es genauestens überprüft und Kritik geäußert.

Bei den User Roles sollten folgende Fragen beantwortet werden:

1. Wer würde oder könnte das System benutzen?
2. Welcher Klasse oder Gruppe gehören sie an?
3. Was zeichnet sie aus, wie sie das System benutzen?
4. Was charakterisiert ihre Beziehung zum System?
5. Was brauchen sie typischerweise vom System?
6. Wie verhalten sie sich gegenüber dem System und was erwarten sie vom Verhalten des Systems?

User Role Tabelle

Abstrakte Klasse und Wer? (1., 2.)	Wie ist die Benutzung des Systems? (3.)	Beziehung zum System (4.)	Bedürfnisse (5.)	Verhalten gegenüber System (6.)	Erwartetes Verhalten vom System (6.)
Durch Alltagsstress Vergessliche (Berufstätige, Familien, Freunde)	Schnelle und häufige Benutzung, viele Kontakte, viele Erinnerungen	Funktion der Erstellung der Erinnerungen sowie die Funktion der Themen-vorschläge wichtig	Effiziente Bedienung, lieber Symbole als Text	schnell, oft, kurzlebig	Zuverlässigkeit, Schnelligkeit, Gebrauchs-tauglichkeit

Durch Arbeitsstress Vergessliche	Schnelle und häufige Benutzung, Arbeitskollegen als Kontakte, viele Erinnerungen	Funktion der Erstellung der Erinnerungen wichtig, Themenvorschläge unwichtig, da sonst nur Themen über die Arbeit	Effiziente Bedienung	schnell, oft, kurzlebig	Zuverlässigkeit, Schnelligkeit, Gebrauchstauglichkeit
Durch Haushalt Vergessliche	Häufige Benutzung, eher Familie als Kontakte, viele Erinnerungen, zu Hause in Ruhe Erinnerungen erstellen	Funktion der Erstellung der Erinnerungen sowie die Funktion der Themen-vorschläge wichtig	Effiziente Bedienung	in Ruhe, oft, kurzlebig	Zuverlässigkeit, Gebrauchstauglichkeit
Organisatorische Benutzer (Außendienst)	Wohl überlegte Benutzung, ausgewählte Kontakte, Erinnerungen spezifisch.	Funktion der Erstellung der Erinnerungen wichtig, Themenvorschläge eher nicht	Effiziente Bedienung	in Ruhe, wohl überlegt, oft überprüfend	Zuverlässigkeit, Gebrauchstauglichkeit
Krankheits-bedingt Vergessliche (Alte Menschen)	langsame Benutzung, wenig Kontakte, simple Erinnerungen	Nur Funktion der Erstellung der Erinnerungen wichtig.	Viel Text, wenig Symbole, große, gut lesbare Buchstaben mit hohem Kontrast. Verständliche, einfach zu erklärende Bedienung.	langsam, ausgewählte Erinnerungen	Zuverlässigkeit, Einfachheit

Introvertierte Menschen (Schüchterne, ruhige Personen)	Ausgewählte Kontakte, wenige aber ausführliche Erinnerungen	Funktion der Themen-vorschläge besonders hilfreich, da gemeinsame Interessen schnell gefunden werden können.	Fachliche Informationen der Themenvorschläge, keine kritischen “Fake News”	schnell, oft, kurzlebig	Zuverlässigkeit, Schnelligkeit, Gebrauchs-tauglichkeit
Extrovertierte Menschen (viele Freunde, viele Gesprächs-themen)	Viele Kontakte, schnelle und häufige Benutzung, viele Erinnerungen	Funktion der Erstellung der Erinnerungen sowie die Funktion der Themen-vorschläge wichtig, da sie so neue interessante Themen ansprechen, die dem Gegenüber auch wirklich interessieren	Gute Übersicht aller Kontakte, keine Belästigung wegen zu häufigen wieder-kehrenden Erinnerungen, effiziente Bedienung, lieber Symbole als Text	schnell, oft, kurzlebig	Zuverlässigkeit, Schnelligkeit, Gebrauchs-tauglichkeit

User Role Maps

Die User Role Map verbildlicht die Beziehungen der einzelnen User Roles zueinander. Somit ist auf einem Blick zu erkennen, wer die Benutzer des Systems sind und wie sie dieses nutzen. Die User Roles können auf 3 unterschiedliche Weisen verbunden sein [QUELLE]:

Affinity

Die User Roles besitzen viele Gemeinsamkeiten. Sie besitzen ähnliche Interaktionen, Erwartungen und Charakteristiken. (Dargestellt durch eine gestrichelte Linie und dem Stichwort “resembles”) .

Classification

Manche User Roles stellen eine Unterklasse einer generellen User Role dar. Diese sind eine spezialisierte Version der Oberklasse und besitzen spezifische Eigenschaften, Interaktionen oder Bedürfnisse etc. (Dargestellt durch Pfeil von Unterklasse zur Oberklasse und dem Stichwort: “specializes”).

Composition

Manche User Roles kombinieren die Eigenschaften und Charakteristiken von zwei oder mehr anderen Rollen. Das heißt, dass ein User die Rolle von zwei anderen User Roles einnehmen kann, auch wenn sich diese grundlegend unterscheiden. (Dargestellt durch Pfeil von kombinierter Klasse zu den speziellen Klassen und dem Stichwort “includes”).

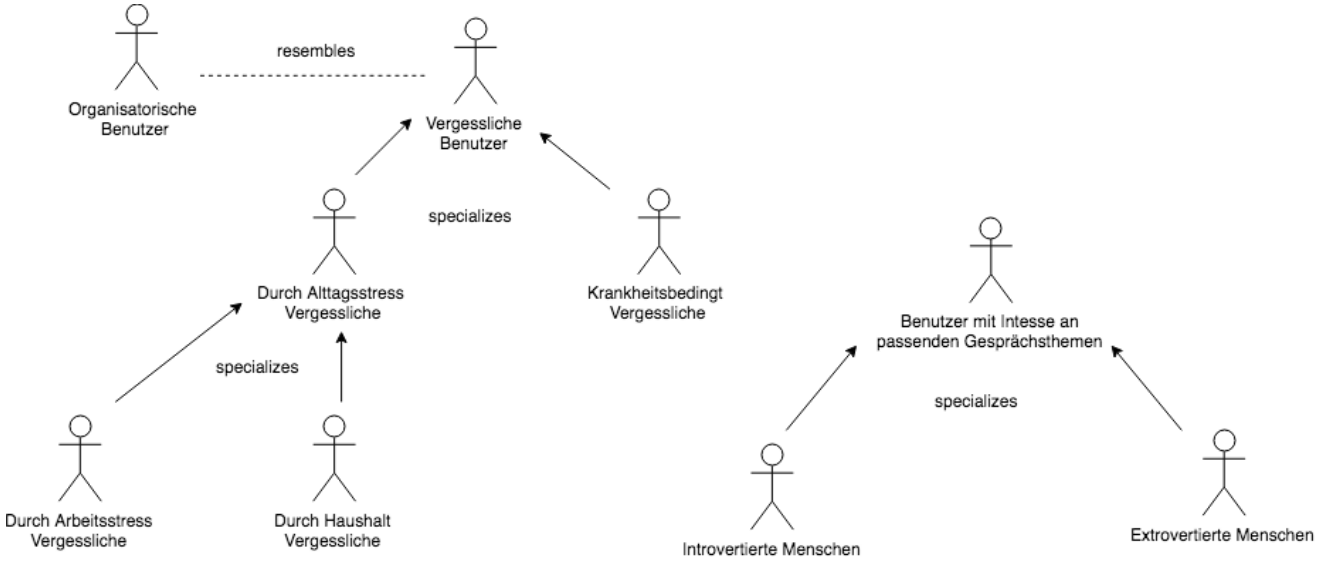


Abb 2: User Role Map

Fazit aus User Roles

Die User Roles ähneln sich alle dahingehend, dass es normale Benutzer des Systems sind. Es gibt keine Personen mit besonderen Rechten oder Pflichten, wie bspw. bei Planungssystemen.

Grundlegende Unterscheidungen gibt es bei den Benutzern, die vergesslich sind und denjenigen, für die das Gesprächsthema eine wichtige Rolle spielt. Dies spiegelt sich auf die zwei Hauptfunktionen des Systems.

Ein besonderer Fokus muss auf die krankheitsbedingten Vergesslichen Benutzern gesetzt werden. Diese haben als einzige Benutzerrolle wichtige zu berücksichtigende Bedürfnisse. Das System muss für ältere Personen, die nicht oft mit Technik umgehen, verständlich sein und das User Interface muss für Personen mit Sehschwächen angepasst sein.

Allgemein profitiert das System stark vom Netzwerkeffekt [QUELLE Netzwerkeffekt]. Je mehr Benutzer das System nutzen und Erinnerungen anlegen, desto höher ist der Nutzen für die Benutzer. Dies kann sowohl einen positiven Effekt bewirken, wenn viele Benutzer das System nutzen und somit auch gute Themenvorschläge geliefert werden, als auch einen negativen Effekt geben, falls wenige Benutzer das System für Erinnerungen nutzen oder nur eine der beiden Kommunikationspartner, sodass das System dieser Person keine guten Vorschläge liefern kann.

Task Model

Aufbauend auf den Rollen der Benutzer kann die Modellierung über das Task Model vorgenommen werden. Hierin wird beschrieben, wie der Benutzer vorgeht um seine Aufgaben zu erledigen. Bei komplexen Aufgaben (“Tasks”) werden sie in kleine “Subtasks” zerlegt. Dadurch ist es möglich auch große und komplizierte Aufgaben verständlich abbilden zu können. In den folgenden Abschnitten werden diese Aufgaben mittels Essential Use Cases modelliert und sie zueinander in Verbindung gesetzt. [QUELLE Buch Seite 99-100]

Use Cases

Zunächst werden in diesem Modellierungsschritt Use Cases erarbeitet, anders als bei Szenarien, die die Probleme der Benutzer erzählend umschreiben, geht es bei bei den Use Cases um die konkrete Verwendung des Systems und der Interaktionen mit diesem. “Jeder Use Case beschreibt eine Interaktion, die vollständig, klar definiert und für einige Benutzer von Bedeutung ist” [QUELLE Seite 101]. Hierbei wird der komplette Verlauf der Interaktionen mithilfe einer Liste beschrieben. Folgende Use Cases gibt es:

- Kontakt anlegen
- Erinnerung hinzufügen
- Benachrichtigung zu einer Erinnerung erhalten
- Themenvorschläge nutzen

Essential Use Cases

Bei den Essential Use Cases geht es dabei noch mehr um den Zweck und Absichten des Benutzers und weniger um die sequentielle Reihenfolge der Aufgabenlösung [QUELLE Seite 103].
Essential Use Cases werden mit hilfe einer Tabelle dargestellt. Über der Tabelle befindet sich der Name des Essential Use Case. Die linke Spalte zeigt die User Intentions, also die

Absichten, die der Benutzer bei der Handlung hat. In der rechten Spalte befinden sich die System Responsibilities, die die Antwort vom System aufzeigt, die der Benutzer erwartet [QUELLE Seite 104].

Kontakt anlegen

User Intention	System Responsibility
Ein Gerät in der Nähe finden	
	Nahe Geräte werden aufgelistet
Gerät auswählen	
Mit diesem Gerät koppeln	
	Kopplung wird vollzogen
Name für Kontakt eintragen	
Kontakt speichern	

Erinnerung hinzufügen

User intention	System responsibility
Kontaktliste einsehen	
	Kontaktliste ausgeben
Kontakt auswählen	
	Kontaktdetailseite (Erinnerungen und Themenvorschläge) anzeigen
Erinnerung hinzufügen	
Erinnerungstext einfügen	
Themen-Label setzen	
speichern	

Benachrichtigung zu einer Erinnerung erhalten

User intention	System responsibility
Zwei Kontakte nähern sich	
	Benachrichtigung senden
Benachrichtigung öffnen	

Benachrichtigung lesen	
Mit Kontakt darüber sprechen	
Erinnerung als erledigt markieren	
	Benachrichtigung wird nicht erneut versendet

Themenvorschläge nutzen

User intention	System responsibility
App vor einem Treffen öffnen	
	Kontaktliste wird angezeigt
Kontakt auswählen	
	Kontaktdetailseite (Erinnerungen und Themenvorschläge) anzeigen
Themenvorschlag ansehen	

Use Case Maps

// Beziehungen der Elemente untereinander als Karte modelliert

Fazit aus Use Cases

// wir sollten möglichst aus jedem Teilabschnitt ein Fazit ziehen um einen roten Faden zu bilden

Einbeziehung der Technologie

Begründung für das Einbeziehen

// Begründung warum wir jetzt die Technologie brauchen -> für content model

Schlüsse aus der Modellierung der Systemkomponenten

// Aus dem WBA2 Teil der Modellierung der Systemkomponenten geht hervor, dass Bluetooth der beste Weg ist, da wir Smartphone als Endgeräte/Clients erwähnt haben.
// Außerdem bietet die Android Entwicklung hier schon viele Funktionen mit sich, die für unsere Ergebnisse aus dem Task Model zutreffen.

Content Model

Das Content Model zielt darauf ab eine abstrakte Repräsentation der Inhalte der verschiedenen Interaktionsräumen zu erstellen. In diesen Interaktionsräumen geschieht die Ausführung der Benutzeraufgaben.
Die Navigation Map bildet ab, wie sich der Benutzer zwischen den einzelnen Interaktionsräumen bewegen kann.

Im Buch “Software for use” werden die Begriffe Content (Inhalt) und Context (Zusammenhang) gleichgestellt. Der Nutzungskontext ist also der Interaktionsraum und in diesem befindet sich der Content, also Funktionen und Materialien.

Content Model

Ein abstrakter Prototyp unterstützt die Designer zuerst darüber nachzudenken, was im User Interface benötigt wird, bevor sich damit auseinandersetzt wird, wie es aussehen oder wie es sich Verhalten wird.

Die Technik die von Constantine und Lockwood vorgeschlagen wird, umfasst ein Blatt Papier sowie Post-Its. Somit ist gewährleistet, dass sich der Designer noch keine bildlichen Vorstellungen macht. Lediglich die Funktionen und Materialien werden abgebildet, sodass Designentscheidungen erst später getroffen werden müssen. Dies hilft dabei das User Interface einfach zu halten und sicherzustellen, dass wirklich nur die erforderlichen Inhalte Bestandteil sind.

Das Vorgehen beginnt mit einem leeren Papier. Hier wird der Interaktionsraum aussagekräftig benannt. Namen wie “Main Screen” sollten vermieden werden.
Danach werden die verschiedenfarbigen Post-Its beschriftet. Die Funktionen (Tools) sind die aktiven, vom Benutzer steuerbaren Elemente. Diese können, wenn gewünscht, nach der im Buch beschriebenen Konvention in warmen Farben, wie pink, orange und gelb dargestellt werden. Dies ist jedoch keine Vorschrift, sondern wird vorgeschlagen, um ein einheitliches Schema zu haben.
Die Materialien (Materials) sind Daten, Container oder simple Anzeigen, mit denen der Benutzer passiv handelt. Diese können in kalten Farben wie blau oder grün dargestellt werden.

Mit diesem Vorgehen werden die verschiedenen Interaktionsräume, die sich aus den Use Cases ergeben, abstrakt dargestellt. Das Content Model bietet die Grundlage für spätere Designentscheidungen und stellt einen gewissen Grad der Gebrauchstauglichkeit sicher.

Content Models für Meet & Remind

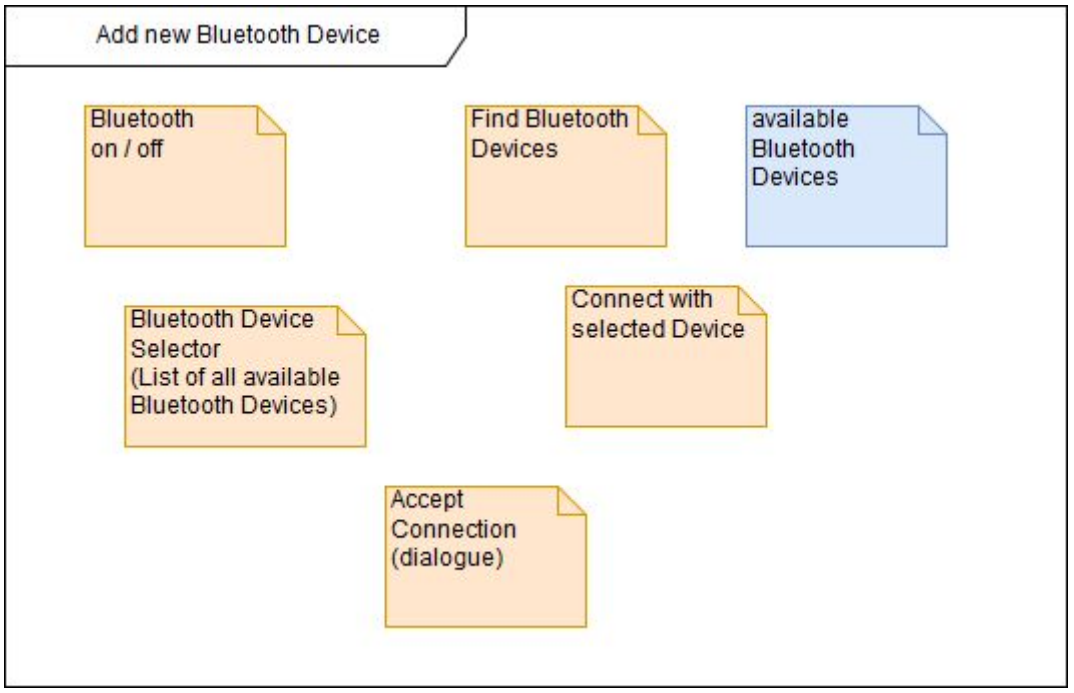


Abb. X

Dieser Interaktionsraum wurde bereits im Proof of Concept implementiert. Dabei ging es nur um die Funktionalität der Bluetooth Kopplung. Im PoC gab es eine Funktion, die die Bluetooth Sichtbarkeit des eigenen Geräts ein- oder ausschaltet. Beim Gestalten des Models ist aufgefallen, dass diese Funktion nicht zwangsweise benötigt wird, da der Benutzer beim Einschalten von Bluetooth davon ausgehen wird, dass sein Gerät für Andere sichtbar ist. In der Praxis ist dies nicht automatisch gewährleistet und muss daher im Code berücksichtigt werden, damit kein zusätzlicher Aufwand für den Benutzer entsteht.

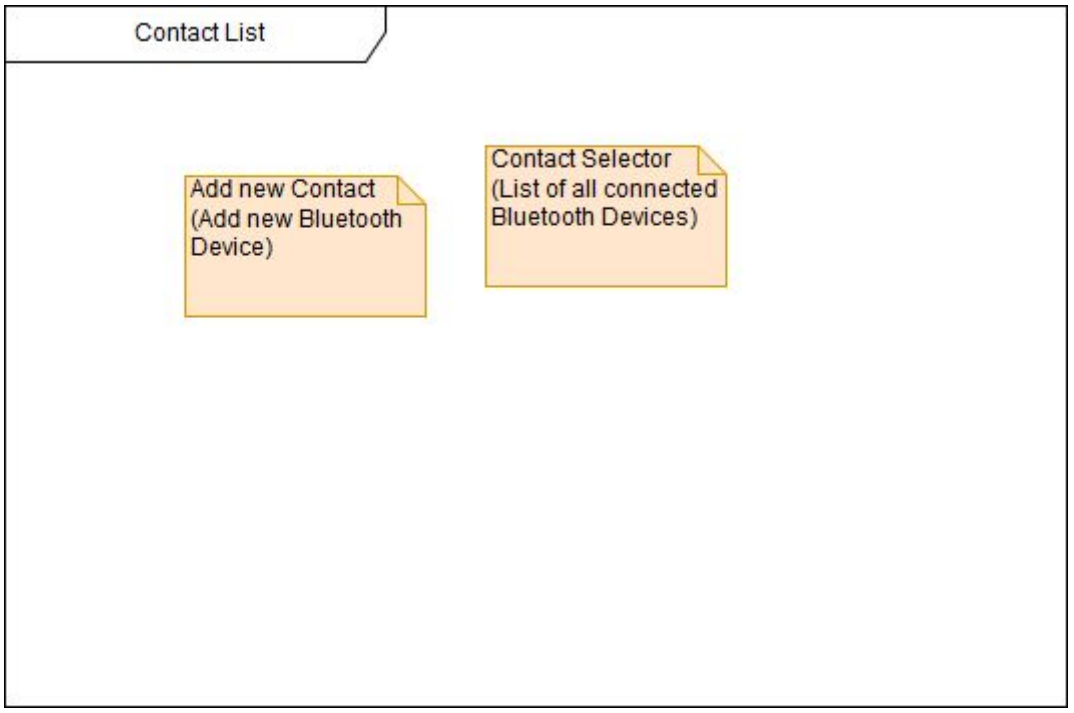


Abb. X

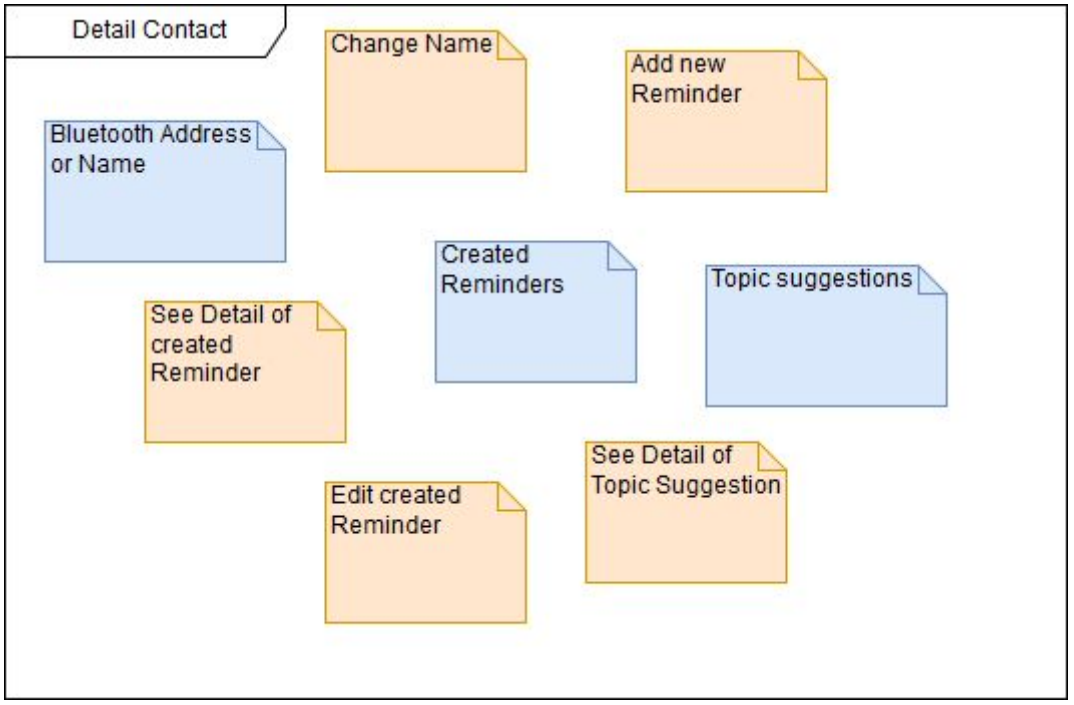


Abb. X

Hier ist eine neue Anforderung hinzugekommen: Das System muss dem Benutzer die Möglichkeit bieten der Bluetooth Adresse einen Namen hinzuzufügen. Andernfalls müsste sich der Benutzer merken, welche Person zu welcher kryptischen Adresse gehört.

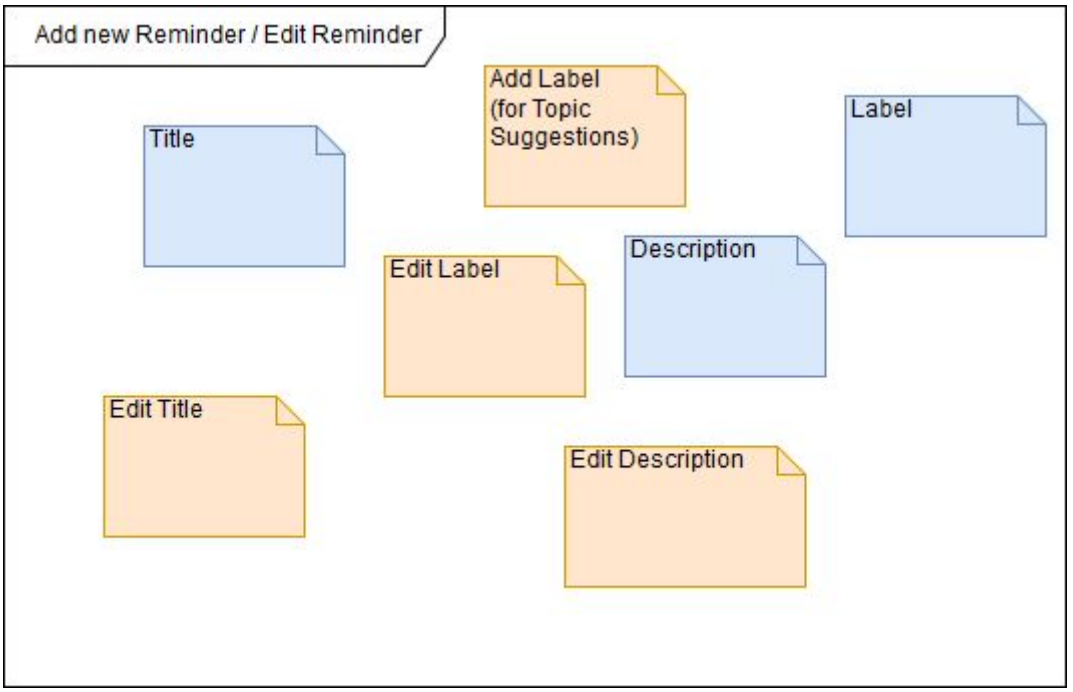


Abb. X

Erst bei der Erstellung des Models ist aufgefallen, dass in den Anforderungen auch noch die Funktionen des Editierens hinzugefügt werden müssen.

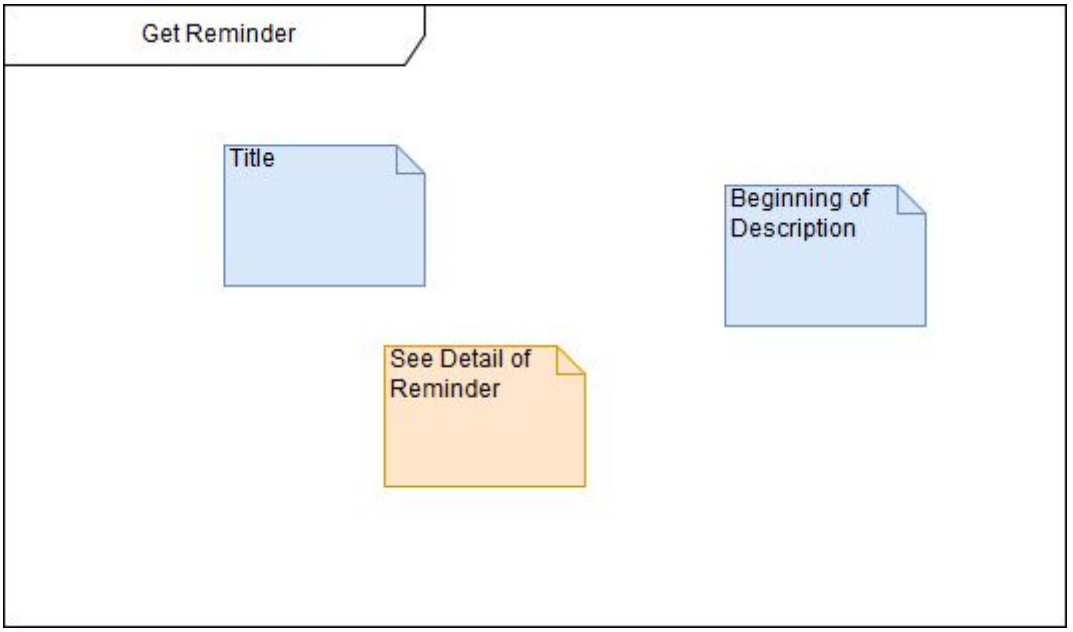


Abb. X

Dieser Interaktionsraum befindet sich nicht direkt in der Applikation, sondern als Benachrichtigungsfenster auf dem Smartphone. Von dort aus kann jedoch direkt auf die App zugegriffen werden.

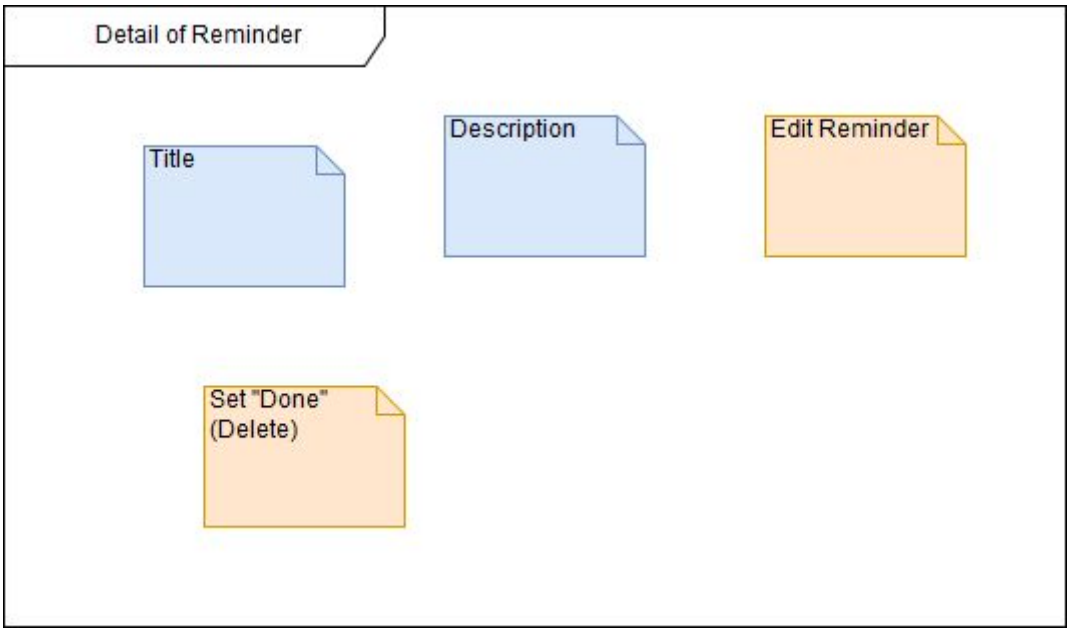


Abb. X

Hier gibt es die zusätzliche Funktion die Erinnerungen als “Erledigt” zu markieren. Damit wird die Erinnerung gelöscht. Wird dies nicht markiert, setzt die Anwendungslogik ein, die Erinnerung beim nächsten Mal erneut zu schicken.

Context Navigation Map

Das User Interface sollte einen einfachen und effizienten Workflow bieten. Daher wird in der Context Navigation Map modelliert, wie sich die Benutzer von einem Interaktionsraum zum anderen navigieren, um ihre Aufgaben, die in den Use Cases beschrieben sind, zu erfüllen. Die komplette Struktur der User Interface Architektur soll abstrakt dargestellt werden.

Bei den Interaktionsräumen soll beachtet werden, dass der Kontext zwar klein und einfach gestaltet sein sollte, dies aber im Endeffekt komplex werden kann, da es mehr Interaktionsräume gibt, zwischen denen navigiert werden muss.
Andererseits sind wenige, komplexe Interaktionsräume auch nicht zielführend.
Hier muss ein gutes Mittelmaß gefunden werden.

Für die Context Navigation Map wird ein Diagramm erstellt, in dem Rechtecke die Kontexte abbilden und Pfeile die Beziehung zwischen ihnen, also die möglichen Übergänge, die ein Benutzer (oder das System) auslösen kann.

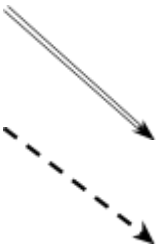
Für unser System sind folgende Konventionen, die zum Teil von uns ergänzt wurden, zu gebrauchen:

Jeder Interaktionsraum

Screen oder Anzeige

Dialogfenster oder Nachricht

Übergang zwischen Interaktionsräumen ausgelöst durch “Aktion”.
Unterscheidungen: Menü Auswahl: View | Toolbars
Button oder Listenauswahl: [Apply], [Kontakt]
Icon oder Tool Auswahl: <page width>
Vom System ausgelöst: “Erfolg”



Übergang mit impliziertem “Return”.
Kurzschreibweise, damit nicht zwei Pfeile gemacht werden müssen, wenn eine Rückkehr durch Return oder Ereignisse wie OK oder Cancel möglich sind.
Aus technischen Gründen ist im untenstehendem Diagramm der Pfeil gestrichelt.

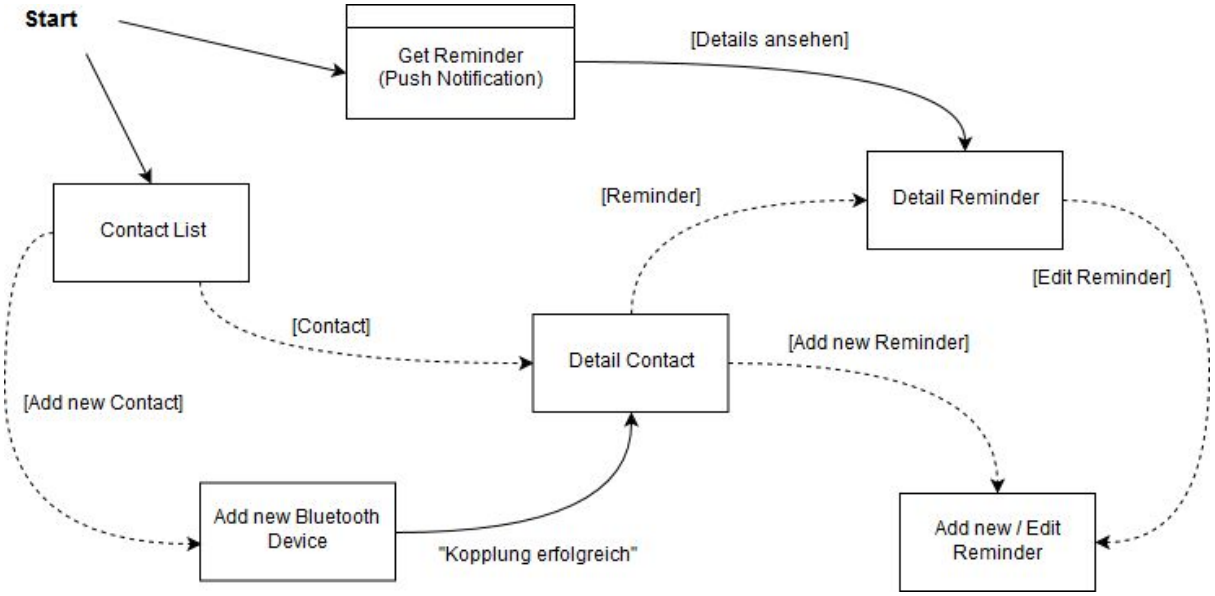


Abb. X:

// Eventuell aus MoCo:
Navigation Stack Abbildung und unsere spezifische Reihenfolge/ Abweichungen des Stacks?

Activity Lifecycle? Wann sollen noch welche Funktionen ausführbar sein? Z.b. dass Bluetooth aktiviert bleibt, nach Ausschalten des Geräts… Oder es cool wäre, wenn man die Benachrichtigung auch bekommt, wenn die App nicht im Hintergrund geöffnet ist...

Fazit aus Content Model

// wir sollten möglichst aus jedem Teilabschnitt ein Fazit ziehen um einen roten Faden zu bilden

Operational Model (optional)

// Modelliert den Kontext in dem das System benutzt wird

Operational Profiles

Fazit aus Operation Profiles

// wir sollten möglichst aus jedem Teilabschnitt ein Fazit ziehen um einen roten Faden zu bilden

Implementational Model

// Beinhaltet das visuelle Design des User Interface sowie die Beschreibungen seiner Operationen

Visual Design

Usability Inspection

Evaluierung

Systemkomponenten

Komponenten

// Nur beschreiben der einzelnen Komponenten
// Wo liegen die Komponenten

Dienste im Web

// Firebase beschreiben
// Externer Webservice Alternativen und Begründung

Architektur

// Architekturdiagramm
// Beziehungen zwischen den Komponenten

Kommunikation zwischen den Komponenten

// asynchron /synchron in Bezug auf das Architekturdiagramm

Programmiersprachen

Die Auswahl der Programmiersprache ist ein wichtiger Schritt bei der Konzeption eines interaktiven Systems. Hier gibt es mehrere Kriterien, die für die Auswahl ausschlaggebend sind, zu beachten. Zunächst sollte geprüft werden, welche Programmiersprache bereits bekannt ist und ob die geforderten Systemkomponenten mit dieser umsetzbar sind. Hier ergeben sich die Vorteile durch Kompaktheit des Codes und bereits vorhandener Frameworks oder Libraries, die für die Umsetzung genutzt werden können. Die Erfahrung der Entwickler spielt bei der Wahl der Programmiersprache auch eine große Rolle. Die Einarbeitung in eine unbekannte Sprache kann aufgrund des Syntax und der Kompaktheit viel Zeit in Anspruch nehmen.

Clientseitige Programmiersprache

Da für den Client eine native mobile Anwendung in Form einer Android App implementiert wird, ist die Wahl der Programmiersprache stark eingegrenzt. Die Unterstützung von Java ist hier am größten [QUELLE für ANDROID APP Sprachen].
In den Modulen Algorithmen und Programmierung 1 und 2 wurde der Umgang mit der objektorientierten Entwicklung mit Java gelernt und in Praktika vertieft. Auch in den weiteren Modulen haben die Entwickler Java mit importierten Libraries und Frameworks genutzt, um Projekte zu bearbeiten.

Serverseitige Programmiersprache

// NodeJS wegen WBA2, dem Zusammenspiel mit Firebase , der Kompaktheit

Entwicklungsumgebung

Aufbauend auf der Wahl der Programmiersprache wird die Entwicklungsumgebung (auch “Integrierte Entwicklungsumgebung”, kurz “IDE” [QUELLE]) ausgewählt. Auch hier gibt es wie bei bei der Programmiersprache mehrere Kriterien, die zu beachten sind. Auf der einen Seite stehen die Kriterien der Umgebung selbst. Ein Kriterium kann die Funktionalität mit der gewählten Programmiersprache genannt werden. Die IDE sollte zum Beispiel Code-Autovervollständigung und ein gutes Syntax-Highlighting für die Programmiersprache besitzen. Mit diesen Hilfsmitteln wird es für den Entwickler leichter den Code zu schreiben und zu verstehen.
Dadurch wird auch die andere Seite der Kriterien klar. Hier stehen die Erfahrungen und Vorlieben der Entwickler mit der IDE selbst. Kennen die Entwickler sich bereits mit einer Umgebung aus, die die ersten Kriterien erfüllen? Haben sie bereits in anderen Projekten gute Erfahrungen machen können?

Clientseitige Umgebung

Unser Client besteht aus einer Android App, die mit Java entwickelt wird und für das Layout der Interaktionselemente XML verwendet. Google bietet für ihre Android App eine offizielle IDE Namens Android Studio [QUELLE] an.
Betreffend der im oberen Abschnitt genannten Kriterien bietet es eine sehr gute Code-Autovervollständigung und ein gutes Syntax-Highlighting an. Da sie genau für die Entwicklung von Android Applikationen aufgebaut ist, ist auch die Integration von anderen Java Klassen unkompliziert. Gerade bei der Entwicklung der Bluetooth Komponenten bringt dies viele Vorteile mit sich. Da wir außerdem für die Datenhaltung Firebase nutzen, dass ebenfalls von Google kommt, ist auch hier die Integration gut umgesetzt.
Des weiteren sprechen die Kriterien der Entwickler für den Einsatz dieser IDE. Im Wahlpflichtfach Mobile Computing haben beide Teammitglieder den Umgang mit Android Studio gelernt und bereits in einem Projekt angewendet.

// Alternative Eclipse oder Intellij???

Serverseitig Umgebung

Da Serverseitig zur Entwicklung NodeJS genutzt wird, wird eine IDE benötigt, die Javascript-fähig ist. Hier bietet die IDE IntelliJ von Jetbrains [QUELLE] gutes Syntax-Highlighting und eine gute Code-Autovervollständigung an. Zudem gibt es für NodeJS Plugins, die genutzt werden können, um den kompletten Entwicklungsprozess über diese IDE handhaben zu können.
Auch hier sprechen die Kriterien der Entwickler für IntelliJ. Im Modul Web-basierte Anwendungen 2 wurde im Rahmes eines Projektes zum Aufsetzen eines Servers IntelliJ genutzt. Die Vorgänge zum Kompilieren und Ausführen des Servers sind also bereits bekannt.

// Alternativen ??

Zusammenspiel der beiden Umgebungen

Da Android Studio auf Jetbrains IntelliJ basiert, spielen die beiden Umgebungen sehr gut zusammen und weisen ein sehr ähnliches Interface auf. Eine Umstellung bei der Entwicklung von Server und Client also nicht nötig. Auch die Integration von Git ist in beiden standardmäßig vorhanden und verhält sich identisch, sodass es zu keinen Konflikten bei der Versionsverwaltung kommt.

Arten von Endgeräten

// Einleitung

Begründung für Android

// Warum Android Geräte (Verteilungsdiagramm zu Geräte-Verteilung iOS/Android

Verfügbarkeit

// Android ab 7.0

Verteilte Anwendungslogik

Datenstruktur

Persistente Datenhaltung

Datenformat

PoCs

//kritisch reflektieren und verbessern.

Fazit

Quellen

1. Beispielquelle: https://www.paulwatzlawick.de/axiome.html (abgerufen am 08.11.2018)
2. https://www.hanser-elibrary.com/doi/book/10.3139/9783446443136 Alternative: https://books.google.de/books?id=pJKqBAAAQBAJ&pg=PA1&dq=requirements+engineering&hl=de&sa=X&ved=0ahUKEwiHrOaOjonfAhVJUlAKHaNJBm4Q6AEIZDAI#v=onepage&q&f=false
3. Software for use: ….