Fachhochschule Köln
Cologne University of Applied Sciences

DiaMon

Ein Diabetes-Management System für Kinder und Jugendliche

Meilenstein 2 - Konzept

EIS
Entwicklung Interaktiver Systeme

ausgearbeitet von

Edgar Gellert
Marcel Holter

im Studiengang
Medieninformatik

Dozenten : Prof. Dr. Gerhard Hartmann
Fachhochschule Köln

Prof. Dr. Kristian Fischer
Fachhochschule Köln

Betreuer : B. Sc. Robert Gabriel
Fachhochschule Köln

Gummersbach, 26. Juni 2015

Kurzfassung

Die Dokumentation zum Modul EIS - Entwicklung Interaktiver Systeme beschreibt das Vorgehen bei der Entwicklung des Diabetes Management Systems *Diagotschi*. Es werden MCI spezifische Vorgehensweisen, sowie WBA2 technische Aspekte des Projektes aufgelistet und erläutert.

Inhaltsverzeichnis

1 Konzept

1.1 Zielhierarchie

Die Zielhierarchie dient zur Definition der langfristigen (strategischen), der mittelfristigen (taktischen) und der kurzfirstigen (operativen) Ziele, die wir mit unserem System verfolgen.

Strategische Ziele:

- Diabetiker müssen viel Zeit in die Erfassung ihrer Vitalwerte investieren, was oft mit Einschnitten in der Lebensqualität einhergeht. Für Kinder ist dies eine besonders schwere Einlern-Phase. Es soll also ein System entwickelt werden, welches die Lebensqualität von an Diabetes erkrankten Kindern steigert.
- Das System soll die Eigenständigkeit und die Selbstsicherheit beim Umgang mit der Krankheit fördern.

Taktische Ziele:

- Kindern soll die kontinuierliche Erfassung aller relevanten Informationen erleichtert werden.
- Das System soll Kindern helfen die Durchführung der Insulintherapie zu erleichtern.
- Die Erfahrung mit Diabetes umzugehen entwickelt sich erst im Verlauf der Jahre. Dem Kind soll mit der Applikation geholfen werden, die nötige Erfahrung auszugleichen und kontinuierlich zu steigern.
- Die Eltern sollen Einsicht auf das Messverhalten der Kinder aus der Ferne haben.
- Diabetologen sollen Zugriff auf die für die Anpassung der Insulintherapie notwendigen Informationen haben.
- Das System soll Eltern die Möglichkeit bieten Erfahrungen auszutauschen und sich gegenseitig zu helfen.

Operative Ziele:

- Es soll eine Vision des Projektes in Form eines umfangreichen Architekturdiagramms realisiert werden.
- Es soll ermittelt werden, auf welche Art und Weise Kindern die Präsentation von relevanten Informationen zugänglich gemacht werden kann.
- Es soll ermittelt werden auf, welche Art und Weise Kinder motiviert werden können am Ball zu bleiben. Dienlich kann hier beispielsweise die Motivationspsychologie sein.
- Es soll ermittelt werden, zu welchem Grad die Unterstützung des Kindes ins Negative ausschlagen kann. Genauer gesagt, ab welchem Maß kann Unterstützung schädlich sein. Dies würde der Steigerung der Erfahrung entgegenwirken.
- Es soll ermittelt werden, auf welche Art und Weise detaillierte Statistiken des Kindes repräsentativ visualisiert werden können, damit Eltern diese auf unterschiedlichen Endgeräten perzipieren können.
- Es soll ermittelt werden, auf welche Art und Weise detaillierte Statistiken des Kindes repräsentativ visualisiert werden können, damit Diabetologen diese zur Ferndiagnose nutzen können.
- Es soll ermittelt werden auf welche Art und Weise verschiedene Parteien verknüpft (Matchmaking) werden können. Was oft in Spielen implementiert wird, um faire Partien zu ermöglichen oder auf Partnerbörsen, um zwei Menschen mit Gemeinsamkeiten einander näher zu bringen, könnte ebenfalls in diesem Anwendungsfall funktionieren.

1.2 Marktrecherche

Auf dem Markt werden viele Applikationen angeboten, die Teilfunktionen zur Lösung des Nutzungsproblems zur Verfügung stellen. Diese werden im Folgenden vorgestellt.

1.2.1 mySugr Diabetes Tagebuch + Junior

Diese Applikation (mySugr) bietet die Möglichkeit Blutzucker-Messungen mit einigen Zusatzinformationen wie zum Beispiel Bilder der Mahlzeiten festzuhalten. Die Daten werden ausgewertet und in verschiedener Form zur Analyse dargestellt. Das digitale Tagebuch kann exportiert werden, um es dem Diabetologen vorzulegen. Kleine "Challenges" und Erinnerungsfunktionen sollen bei fehlender Motivation helfen das Messverhalten zu optimieren. Die Zusatzapplikation "Junior" ist speziell für Kinder entwickelt. Durch ein Belohnungssystem und eine kleine Kreatur sollen Kinder mehr Spaß an den kontinuierlichen Messungen haben. Die Applikation ist mit dem Client der Eltern verbunden. Diese bekommen eine Nachricht für jede Messung des Kindes. Weiterhin können Kinder ihren Eltern Fragen stellen, unterstützt durch eine Foto-Funktion. Wissen sie nicht, wie viele Kohlenhydrate in einer Mahlzeit enthalten sind, können Eltern direkt aus der Ferne helfen.

Stärken:

- Zwei unterschiedliche Klienten für die Nutzergruppen "Eltern" und "Kinder"
- Einfache Logbuchführung
- Erinnerungsfunktion bei nachlässigem Messverhalten
- Belohnungssystem für Kinder zur Motivationssteigerung
- Eltern können Kinder aus der Ferne durch Nachrichten helfen

Schwächen:

- Keine Unterstützung bei Berechnung der Insulinmenge
- Keine Plattform, um Fragen zu stellen

1.2.2 Sugar Sense

Sugar Sense (Medhelp) ist ebenfalls ein digitales Tagebuch für Blutzucker-Messungen. Die Applikation bietet die Möglichkeit Zusatzinformationen wie Sportaktivitäten und Gefühlslage den Tagebucheinträgen hinzuzufügen. Über eine Community lassen sich Fragen zur Krankheit stellen.

Stärken:

- Sehr detailierte Tagebucheinträge
- Community zur gegenseitigen Unterstützung

Schwächen:

- Keine weiteren Funktionen
- Nicht speziell für Kinder

1.2.3 Insulin Calculator

Mit Insulin Calculator (Bowley) lässt sich die Insulindosis berechnen. Man gibt nach Messung den aktuellen Blutzuckerwert, die Kohlenhydrate der bevorstehenden Mahlzeit in Broteinheit und eventuelle sportliche Aktivitäten ein.

Stärken:

- Berechnung der Insulindosis

Schwächen:

- Kohlenhydrate in Mahlzeiten müssen bekannt sein
- Nicht speziell für Kinder

1.2.4 BE-Finder

BE-Finder (Cougar-Media) sucht in einer sich ständig vergrößernden Datenbank nach den Kohlenhydraten in Broteinheit für eine Mahlzeit.

Stärken:

- Große Datenbank mit Broteinheiten in Mahlzeiten

Schwächen:

- Keine weiteren Funktionen
- Nicht speziell für Kinder

1.2.5 Mission T1D

Mission T1D (Sanofi-Aventis) ist ein Spiel vom Entwickler Sanofi. Es hat keinerlei Funktionen zum Erfassen von Blutzuckerwerten oder der Berechnung einer optimalen Insulinmenge. Das Spielt dient lediglich der Unterstützung im Verständnis um die Krankheit Diabetes Mellitus Typ 1, was der am häufigst verbreiteste Diabetes-Typ bei Kindern ist. Der Schauplatz des Spieles ist in einer Schule angesiedelt und liefert lehrreiche Nachrichten mit einem besonderen Schwerpunkt auf Kinder, die selbst zur Schule gehen. Das Spiel besteht aus mehreren Leveln. Um in einem Level aufzusteigen, ist das Kind gezungen sich Lehrvideos über Diabetes anzuschauen. Des Weiteren sollen Quizfragen dem Verinnerlichen von Wissen dienen.

Stärken:

- fördert Wissen und Verständnis von Diabetes
- Gamification-Paradigma wirkt motivierend
- speziell für Kinder entwickelt

Schwächen:

- kein Diabetes Management System
- keine Möglichkeit des Erfassens von Blutzuckerwerten
- keine Berechnung einer Insulinmenge
- Informationen werden nicht abgespeichert
- Eltern werden nicht mit einbezogen
- Diabetologen werden nicht mit einbezogen

1.2.6 Monster Manor

Monster Manor (Ayogo-Health-Inc) ist eine mobile device Applikation und wurde ebenfalls von Sanofi entwickelt. Im Gegenteil zu Mission T1D ist Monster Manor kein Spiel, sondern ein Diabetes Management System für Kinder. Es soll Kindern durch das Gamification Paradigma das Testen des Blutes und das Festhalten der Blutzuckerwerte erleichtern. Kinder werden dazu angehalten selbstständig ihre Werte zu erfassen, indem man sie mit Belohnungen ködert. Nach jedem Erfassen erhält das Kind eine Piñata, die ein virtuelles Geschenk enthält. Es lassen sich dadurch Monster und allerhand Utensilien freischalten. Eltern haben die Möglichkeit sämtliche Erfassungen ihrer Kinder

einzusehen. Des Weiteren können sie die Gamification Aspekte selbst nutzen, indem sie Herausforderungen an das Kind stellen oder dieses bspw. durch Belohnungen unterstützen.

Stärken:

- Diabetes Management System
- speziell für Kinder entwickelt
- Gamification Paradigma wirkt motivierend
- Ermöglicht das Erfassen von Blutzuckerwerten
- selbstständiges Vorgehen durch Aufforderung seitens des Systems
- Eltern haben eigenen Client
- Eltern können Kinder durch Gamification motivieren
- Eltern können sämtliche Informationen des Kindes einsehen

Schwächen:

- Diabetologen werden nicht mit einbezogen
- Eltern haben keine Möglichkeit sich untereinander zu informieren bzw. auszutauschen

1.3 Alleinstellungsmerkmal

Mit den vorgestellten Applikationen lassen sich jeweils Teilaspekte des Nutzungsproblems lösen. Digitale Tagebücher ersetzen ein handschriftliches Logbuch, die Insulindosis lässt sich berechnen, ein Katalog zur Suche nach der Menge der Kohlenhydrate in Mahlzeiten wird angeboten, eine Community um Fragen zu stellen und eine Verbindung zwischen Eltern und Kindern ist verfügbar. Manche dieser Applikationen versuchen Kinder durch spielerische Elemente zu motivieren. Sportliche Aktivitäten lassen sich in manchen Applikationen zwar manuell den Tagebucheinträgen hinzufügen, über eine automatische Erfassung über die Sensorik der Smartphones oder zusätzliche Hardware (Bsp.: Fitness-Armband) hat sich bisher niemand Gedanken gemacht. Jedoch existiert keine Lösung, die alle diese Funktionen in einem System vereint. Deutlich wird das, wenn man die Diabetes-Manager genauer betrachtet. Keiner dieser Applikationen hat einen integrierten Insulinrechner. Der Benutzer muss in den meisten Fällen die korrekte

Insulinmenge selbst berechnen. Kindern, denen es noch an Erfahrung mangelt, müssen hierfür ihre Eltern zu Rate ziehen, da sie es selbst nicht können. Eine Plattform, auf der sich Eltern austauschen können, ist meist nur in Form eines Forums realisiert. Fragen werden willkürlich gepostet und ein jeder kann darauf antworten, wenn er es möchte. Diabetologen gelangen an die Daten lediglich über einen Besuch des Patienten.
Das zu entwickelnde System soll nicht nur als zentrales System aller genannten Funktionalitäten dienen, es soll sich auch durch eine zentrale elektronische "Krankenakte" (Profil) der Kinder und eine automatische Erfassung der sportlichen Aktivitäten von anderen System absetzen. Diabetologen sollen ebenfalls einbezogen werden, um die Profile zu erweitern und Ferndiagnosen erstellen zu können. Auch ein Austausch zwischen Eltern soll stattfinden können. Hierfür werden die Profile der Kinder verglichen, um Gemeinsamkeiten zu nutzen, die eine Vermittlung vereinfacht. Dadurch können Eltern untereinander Problemsituationen bewältigen, um die Lebensqualität des Kindes zu steigern. Dem Kind selbst soll die Berechnung der Insulinmenge abgenommen werden, da ihnen die BE-Werte von Lebensmitteln noch nicht geläufig sind. Aus diesem Grund soll auch die Auswahl der Lebensmittel vereinfacht werden, damit ein Kind alle Werte manuell eintragen kann. Anschließend kann die optimale Insulindosis automatisiert berechnet werden. Folgend sollen die Alleinstellungsmerkmale nochmals konkret aufgelistet werden.

- Integrierter Insulinrechner, um die mangelnde Erfahrung der Kinder auszugleichen.
- Matchmaking System für die Eltern, wodurch Eltern mit ähnlichen Kind-Profilen die Fragen anderer Eltern besser beantworten können.
- Diabetologen sind Teil des Systems zur Erstellung der Profile und um Ferndiagnosen aufstellen zu können.
- Über die Sensorik und/oder externe Hardware (Bsp.: Fitness-Armband) werden sportliche Aktivitäten automatisch erfasst.
- Alle essentielle Funktionen werden in einem System vereint.

1.4 Kommunikationsmodelle

Kommunikationsmodelle sollen sowohl den momentanen als auch den zukünftigen Kommunikationsweg zwischen den Instanzen beschreiben. So visualisiert Abbildung 1.1 den Ist-Zustand und Abbildung 1.2 den Soll-Zustand der Kommunikationswege.

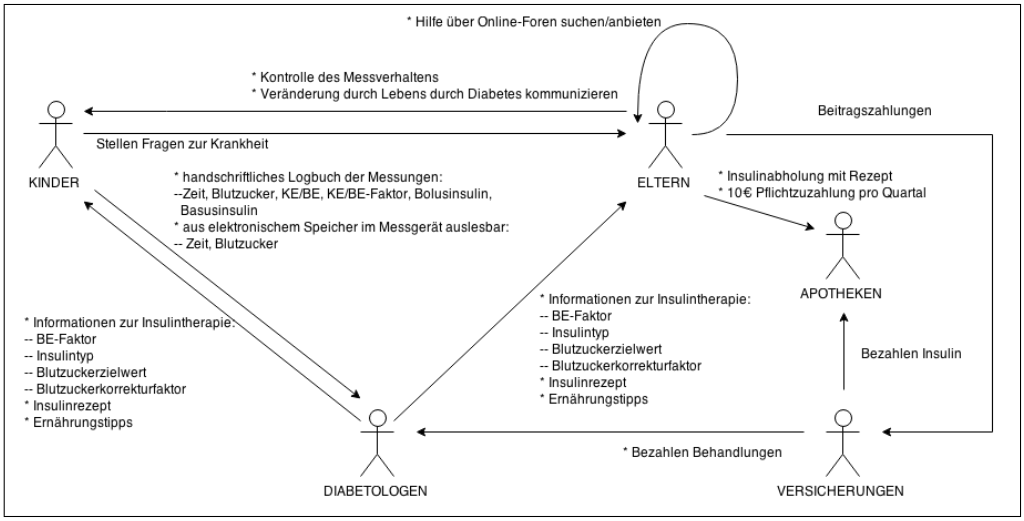

Abbildung 1.1: Das deskriptive Kommunikationsmodell

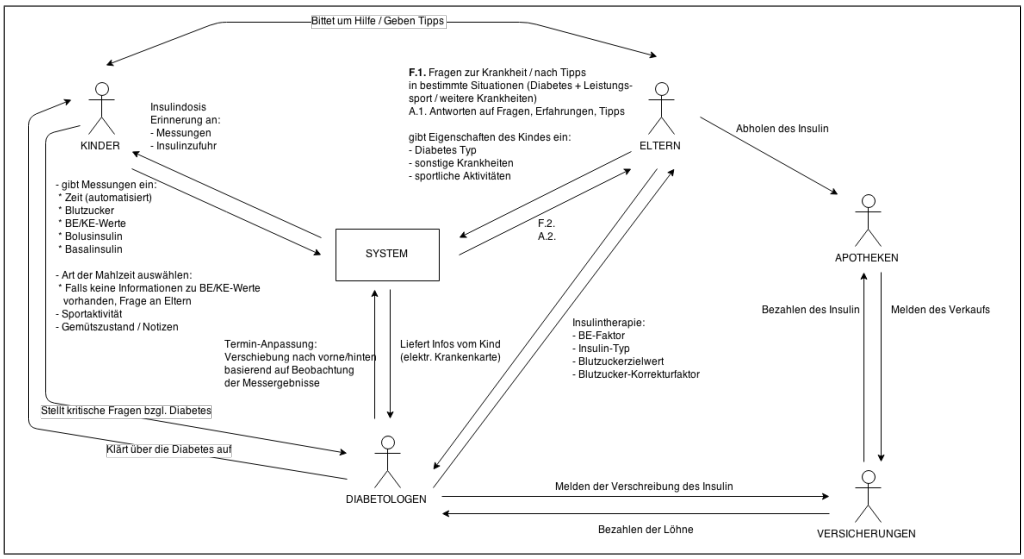

Abbildung 1.2: Das präskriptive Kommunikationsmodell

1.5 Methodischer Rahmen (MCI)

1.5.1 Nutzungskontext

Für die Abwägung eines Vorgehensmodell ist die Analyse des Nutzungskontextes wichtig. Es ist wichtig klarzustellen, welche Domäne mit dem zu entwickelnden System bedient wird. In diesem Fall soll eine Diabetes Management Applikation für Kinder im Alter zwischen acht und sechzehn Jahren entwickelt werden. Unter Beachtung dieser Aspekte soll im Folgenden eine Abwägung des gewählten Vorgehensmodells beschrieben werden.
Besonderer Wert liegt auf folgenden Kriterien:

- Skalierbarkeit, da das Projekt im Rahmen der Veranstaltung umfangreichen Restriktionen unterliegt, wie beispielsweise nur einem 2-Mann Team, muss das Vorgehensmodell nach unten skalierbar sein.
- User-Centered Design, da durch die Wahl der Domäne (Diabetes) ein besonderer Bereich gewählt wurde, der durch diese Wahl eine größere Qualität gewährleistet bekommt.
- Klare Struktur, da dies eine korrekte Vorgehensweise gewährleistet und das Team im Prozess gut unterstützt. Da es an praktischer Erfahrung fehlt solch ein Vorgehensmodell durchzuführen, ist ein Vorgehen, das durch das Vorschlagen von Techniken Hilfestellung leistet erstrebenswert. Des Weiteren hilft dies den Zeitaufwand im Vorfeld einschätzen zu können.

1.5.2 Usage-Centered Design

Das Usage-Centered Design stellt den Verwendungszweck eines Systems in den Vordergrund. Hierbei sind die Merkmale der Benutzung Ausgangspunkt der Modellierung des interaktiven Systems. Es ist also die Aufgabe, die der Benutzer erledigen möchte, die im Zentrum der Entwickler zu stehen hat. Da in der Auflistung zuvor bereits erwähnt wurde, dass durch die Wahl der Domäne ein stärkerer Fokus auf dem Benutzer liegt, wird hier nicht weiter auf das Usage-Centered Design eingegangen.

1.5.3 Szenariobasiertes Vorgehensmodell

Das Szenariobasierte Vorgehensmodell ist in seiner Anwendung äußerst aufwändig und im Rahmen des Projektes zu überdimensioniert. Es hat im Allgemeinen definitv seine Existenzberechtigung und kann für den ein oder anderen Projektinternen Anwendungsfall genutzt werden. Jedoch soll das Szenariobasierte Vorgehensmodell nicht als Hauptmodell genutzt werden.

1.5.4 Disount-Usability Engineering

Das Discount-Usability Engineering nach Nielsen (Nielsen 1994) beschreibt ein kostengünstiges, weniger zeitaufwendiges und komplexes Vorgehensmodell, das sich aus vier Techniken zusammensetzt:

- User and task observation
- Scenarios
- Simplified thinking aloud
- Heuristic evaluation

Das Discount-Usability Engineering kann im Rahmen solch einer Veranstaltung eine sehr gute Anbindung zu einem Projekt finden. Da die Techniken in dem Modell explizit festgelegt sind, könnte dies jedoch den Umgang mit der Domäne einschränken. So ist es im Rahmen des Projektes leider nicht möglich einen Diabetologen zu konsultieren, der Expertenwissen zur Verfügung stellt, worauf konkrete Design-Aspekte diskutiert werden können. Des Weiteren ließen sich nur bedingt an Diabetes erkrankte Kinder ausfindig machen, die an einem think aloud teilnehmen könnten, um so erste Schritte der Evaluation durchzuführen. Abschließend lässt sich sagen, dass die Evaluation anhand der 10 Heuristiken von Nielsen, keine Gewichtung auf eine Domäne im eHealth-Bereich setzt. Viel mehr ließe sich mit Hilfe dieser Heuristiken ein Evaluation im Usage-Centered Design abwickeln.

1.5.5 User-Centered Design

Das User-Centered Design stellt den Benutzer eines Systems in den Vordergrund. Hierbei werden die Merkmale der Benutzer als Ausgangspunkt der Modellierung des interaktiven Systems verwendet. Wie bereits zuvor beschrieben, soll durch die Wahl der Domäne ein besonderer Fokus auf dem Benutzer liegen. Aus diesem Grund fällt die Wahl des Vorgehens auf das User-Centered Design.
Prädestiniert für dieses Vorgehen ist das Vorgehensmodell der *DE EN ISO 9241-210:2010*. Dieses Modell beschreibt ein iteratives Vorgehen im Menschzentrierten Entwicklungsprozess. Eines der wesentlichen Merkmale dieses Modells ist seine generische Vorgehensweise und im Rahmen der Veranstaltung ein Ausschlusskriterium. Es existieren keine expliziten Vorschläge für Techniken oder mögliche Alternative. Da bisher keine praktische Erfahrung bei der Anwendung solcher Vorgehensmodelle existieren kann solch ein generisches Vorgehensmodell auch von Nachteil sein.

1.5.6 Abwägung der Vorgehensmodelle

Nach sorgfältiger Analyse und Abwägung wurde im Rahmen des Projekts der Usability Engineering Lifecycle von Deborah Mayhew (Mayhew (1999)) gewählt. Das Vorgehensmodell beschreibt wesentliche Phasen eines User-Centered Designs. Dieses Modell unterteilt sich in drei klar strukturierte Phasen und erlaubt somit ein kontrollierbares und weniger risikobehaftetes Vorgehen im Entwicklungsprozess. User-Centered Design deshalb, da sich durch die Diabetes-Domäne ein klar abgetrennter Entwicklungsbereich beschreiben lässt. Selbstverständlich ist der Nutzen eines Systems wichtig und erstrebenswert umzusetzen, wodurch ein Vorgehensmodell im Usage-Centered Design seine Existenzberechtigung hat. Jedoch ist mit der Wahl dieser Domäne ein besonderes Augenmerk auf die Benutzer zu richten und der Usability Engineering Lifecycle von Deborah Mayhew erlaubt genau dies zu gewährleisten. Des Weiteren erlaubt es das Vorgehensmodell eine Skalierung auf kleinere Projekte vorzunehmen, was von besonderem Vorteil im Rahmen dieses Projektes ist. Deborah Mayhew beschreibt dieses Vorgehen als "Customizing the Usability Engineering Lifecycle" und liefert mit ihren sogenannten "shortcuts" passende Skalierungen für große, kleine und alle dazwischen liegende Projektgrößen. Für simple Projekte mit nur geringen Ressourcen könnte der Lifecycle folgendermaßen aussehen (Mayhew (1999) Seite 24, frei übersetzt durch den Autor):

- "Quick and dirty"-Versionen von User Profiles und Contextual Task Analysen können verwendet werden.
- Designer können sich verstärkt auf Design-Grundprinzipien und Guidlines stützen, die in der Literatur beschrieben werden.
- Die drei Design-Level können zu einem einzelnen iterativen Designprozess zusammengefasst werden. Aber es sollte beachtet werden, dass es immer noch wichtig ist, sorgfälltige Überlegungen anzustellen und bewusst auf allen drei Level zu entwickeln.
- Das Dokumentieren von Design-Standards im Style-Guide des Produktes kann übersprungen werden.

Für jede Phase liefert Deborah Mayhew Vorschläge für die bereits erwähnten "shortcuts". So ließen sich z. B. für die Anfertigung der User Profiles Interviews mit den Probanden durchführen, anstelle einer Befragung mittels Fragebögen. Ein weiteres Beispiel für "shortcuts" sind die Mock-ups. Anstelle von interagierbaren Prototypen können animierte Mock-ups (z. B. Paper-based prototyping) erstellt werden. Das Projekt kann also je nach dem wieviel Ressourcen zur Verfügung stehen dynamisch angepasst werden.

Ein weiterer positiver Aspekt des Usability Engineering Lifecycle ist die mitgelieferte detaillreiche Sammlung an Usability Task Plänen, Usability Projekt Plänen, User Profile Fragebögen und weiteren Bestandteilen des Vorgehensmodells. Dies erleichtert insbesondere für weniger erfahrene Entwickler das Fortschreiten im Designprozess erheblich.
In Abbildung 1.3 wird der strukturierte Ablauf nochmals visualisiert. Die gestrichelten Pfeile zeigen mögliche Shortcuts für einen groben Projektverlauf; aber auch in jeder einzelnen Phase ließe sich eine Verkürzung des Aufwandes herstellen, indem bestimmte Vorgehensweisen verwendet werden.

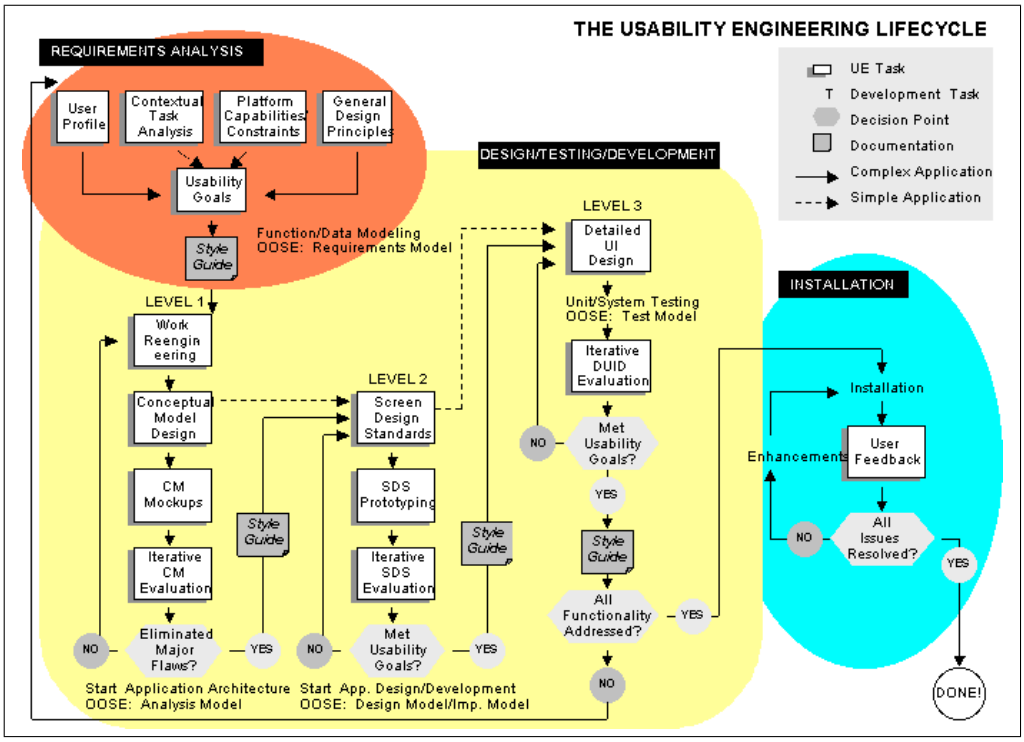

Abbildung 1.3: Der Usability Engineering Lifecycle nach Deborah Mayhew (Mayhew)

1.6 Risiken

Die ermittelten Risiken sollen in den folgenden Passagen genauer beleuchtet und um Maßnahmen der Minimierung und Behandlung erweitert werden.

1.6.1 Krankheit

Ein krankheitsbedingter Ausfall eines Teammitgliedes, oder sogar beider, kann zu einem sofortigen Beenden des Projektes führen. Bei einem Ausfall ließe sich die verlorene Zeit kaum noch aufholen. Da dieses Risiko jedoch auf dem Willen der Projektteilnehmer beruht nicht leichtfertig mit der Gesundheit umzugehen, wird hier eine erhöhte Aufmerksamkeit auf gesünderen Umgang mit dem Körper gesetzt.

1.6.2 Schlechtes Zeitmanagement

Durch die vielen Nebenveranstaltungen besteht die Gefahr, dass das eigentliche Projekt durch schlechtes Zeitmanagement in Verzug gerät. Auch durch allgemein falsches Planen kann es dazu kommen, dass für einige Meilensteine weniger Zeit zur Verfügung steht. Dadurch entsteht das Risiko, dass das Projekt scheitert oder nur unbefriedigend fertiggestellt wird.
Um das Risiko zu minimieren, ist eine sorgfälltige Planung und Dokumentation der einzelnen Voränge notwendig. Hierfür wird ein detaillierter Projekt- und Stundeplan angelegt, um die einzelnen Projektphasen und -vorgänge zu protokollieren und zu planen. Sollte das Risiko zu einem Ereignis werden, so ist eine Analyse der Planung und der einzelnen Vorgänge erforderlich. Nach der Analyse lässt sich dann eine Umstrukturierung des gesamten Prozesses oder auch nur von einzelnen Vorgängen durchführen.

1.6.3 Unzureichende Programmierkenntnisse

Damit Eltern ihre Erfahrungen untereinander austauschen und sich bei Fragen gegenseitig helfen können, müssen die Fragen und somit die Eltern anhand des Profils ihres Kindes zusammengeführt werden. Diesem Verfahren soll ein matchmaking Algorithmus zugrunde liegen. So ein Algorithmus stellt das Team vor eine große Herausforderung, da man sich mit solch komplexen Algorithmen noch nicht konfrontiert sah.
Die Android-Programmierung ist mangels Erfahrung ebenfalls sehr herausfordernd. Hierfür muss einige Zeit für die Einarbeitung eingeplant werden, da kaum Wissen über den allgemeinen Aufbau und dem Prozessablauf einer Android-Applikation bekannt ist. Um das Risiko zu minimieren wurde bereits vor Beginn des Projektes Zeit investiert, um sich in die Materie einzuarbeiten. Sollte dies dennoch nicht genügen, so sind

Überstunden in die Einarbeitung in gewisse Bereiche von Algorithmen und Android-Programmierung zu leisten.

1.6.4 Korrekte Berechnung der Insulinmenge

Die Berechnung der korrekten Insulinmenge ist wichtig, insbesondere für ein Kind, das bei der Ermittlung der einzelnen Werte noch keine Erfahrung hat. Um eben diesem Mangel an Erfahrung auszugleichen, muss die Berechnung fehlerfrei sein. Etwaige Fehler können zu einer Hyperglykämie (Überzuckerung) oder einer Hypoglykämie (Unterzuckerung) führen. Hierfür ist wichtig, dass das Kind die korrekten BE/KE-Werte in das System eingibt. Die Problematik stellt eine exakte Auswahl der Nahrungsmittel dar, die anschließend eine korrekte Berechnung erlauben. So gibt es beispielsweise eine Vielzahl an Brotsorten. Welches Brot das Kind nun vor sich hat ist ihm nicht immer klar, wodurch die berechnete Insulinmenge stets variiert. Um dieses Risiko zu minimieren wurde bei einem Diabetologen um ein Interview gebeten, damit unter anderem Fragen bzgl. Schwankungen von Insulinmengen erörtert werden können. Sollten Schwankungen in einem annehmbaren Intervall akzeptabel sein, so kann in gewissen Bereichen vielleicht ein Mittelmaß genutzt werden. Wird das Risiko zu einem Ereignis, so muss über eine Lösung nachgedacht werden, wie man dem Kind eine genauere Eingabe der Nahrungsmittel ermöglicht, um so eine genauere Berechnung der Insulinmenge zu gewährleisten.

1.6.5 BE/KE-Datenbank

Nach längerer Recherche wurde keine offene Schnittstelle gefunden, die eine Abfrage von BE/KE-Werten erlaubt. Es existiert jedoch ein Bundeslebensmittelschlüssel (BLS) vom Bundesministerium für Ernährung, Landwirtschaft und Verbraucherschutz. Dieser BLS enthält eine Datensammlung mit einer Zusammensetzung sämtlicher Nährstoffe. Der Zugriff auf diese Datensammlung ist jedoch nur mit dem Kauf einer Lizenz verbunden. Der Erwerb solch einer Lizenz steht im Rahmen des Projekt außer Frage. Um das Risiko eines Ausfalls solch einer Datenbank zu minimieren muss eine Alternative durch weitere Recherchen ermittelt werden. Ist das Risiko zu einem Ereignis geworden, so ist die Erstellung eines Dummy-Datensatzes notwendig. Dieser Datensatz ist dann in der Lage, die im Rahmen des Projektes notwendige Grundfunktionalität der Berechnung der optimalen Insulinmenge zu gewährleisten.

1.6.6 Matchmaking

Eine der größten Risiken stellt das Matchmaking dar. Solche Algorithmen können sehr komplex ausfallen, was die Entwicklung eines eigenen Algorithmus schwierig macht. Des

Weiteren muss hierbei die Beachtung der richtigen Parameter gewährleistet sein, damit Elternpaare korrekt zusammengeführt werden. Um das Risiko zu minimieren muss eine weitreichende Recherche erfolgen. Hier können beispielsweise Matchmaking-Systeme aus Online-Spielen, falls diese offen zugänglich sind, genutzt werden, um Eltern nach bestimmten Eigenschaften ihrer Kinder zu verknüpfen. In Spielen werden hierfür Statistiken der Siege, Niederlagen und sonstigen Spielereigenschaften von Spiele-Sessions genutzt, um faire Partien zusammenzufügen. Weitere Matchmaking-Algorithmen werden in Partnerböresen verwendet, um Partnersuchende, zumeist basierend auf Gemeinsamkeiten, einander näher zu bringen. Sollte das Risiko eintreffen, so sind die Benutzer selbst für die Suche nach Fragen zuständig, auf die sie eine sinnvolle Antwort geben können. Hier für kann eine manuelle Filterfunktion hilfreich sein, damit die Benutzer nicht gezwungen sind sämtliche Fragen durchzuforsten.

1.6.7 Arbeitsaufwand unterschätzt

Bei all den zuvor beschriebenen Risiken ist das größte Risiko das Unterschätzen des gesamten Arbeitsaufwands. Die einzelnen Risiken beschreiben sehr gut die komplexen Vorstellungen, auf denen das Projekt aufgebaut wird. Wenn sich durch die einzelnen Projektphasen zu viel aufgebürgt wird, so ist das gesamte Projekt in Gefahr. Um dieses Risiko zu minimieren, sollen die regelmäßig angebotenen Gespräche mit den Betreuern wahrgenommen werden. Des Weiteren sollen Experten-Interviews bzgl. Implementierungsfragen oder MCI-Prozessen angestrebt werden.

1.7 Proof of Concept

Für das Proof of Concept wurden kritische Punkte der Systementwicklung ermittelt. Diese müssen zwingend getestet werden, um eine erfolgreiche Umsetzung zu ermöglichen.

1.7.1 BE/KE-Werte

Um eine korrekte Insulinmenge berechnen zu können, muss der Zugang zu einer Datenbank gewährleistet sein, die die nötigen BE/KE-Werte enthält. Diese Werte müssen abgerufen werden können, um sie anschließend dem Benutzer zur Verfügung zu stellen. Um zu definieren wie viele Datenbankabfragen als "Proof of Concept" – notwendig sind, müssen folgende Statistiken betrachten werden:

- Ca. 70% aller Kinder in Deutschland sind zwischen sechs und 16 Jahren alt ((Statista a))

- „*In der Altersgruppe von 0 bis 19 Jahren sind etwa 30500 Kinder und Jugendliche von einem Typ-1-Diabetes betroffen.*“ [Diabetes-Hilfe, S. 118]
- Unter Berücksichtigung der Smartphone-Nutzung durch Kinder und Jugendliche nach Altersgruppen ((Statista b)) ergeben sich insgesamt grob 13.000 Kinder, die sowohl von einem Typ-1-Diabetes betroffen sind, als auch ein Smartphone nutzen.

Geht man nun davon aus, dass die Hälfte dieser Kinder das System nutzt und täglich durchschnittlich drei Mahlzeiten (morgens, mittags und abends) zu sich nimmt, und schätzt die Zeitintervalle mit den meisten Datenbankabfragen auf zwei Stunden (07:00 - 09:00 Frühstück, 12:00 - 14:00 Mittagessen, 17:00-19:00 Abendbrot), ergeben sich 6.500 Abfragen/2 Stunden = ca. 55 Abfragen/Minute.

Exit: Es lassen sich erfolgreich 50 KE/BE-Werte pro Minute aus der Datenbank auslesen.

Fail: Lassen sich die BE/KE-Werte weniger als 50 mal pro Minute abrufen, gilt das PoC als gescheitert. Dies trägt weitere Konsequenzen mit sich; unter anderem beeinträchtigt ein eingeschrängter Zugriff auf die BE/KE-Werte die User Experience eines Benutzers. Dies kann letztlich dazu führen, dass das System vom Benutzer abgestoßen wird.

Fallback: Bei einem Fallback muss der Benutzer die Werte manuell eingeben. Um einer stetigen manuellen Eingabe entgegenzuwirken, kann das System im Vorfeld genutzte Nahrungsmittel in einer lokalen Datenbank speichern. So kann immer auf einen breiten Datesatz zurückgegriffen werden, ohne eine gesamte Lebensmittel-Datenbank auf dem Endgerät anlegen zu müssen.

1.7.2 Matchmaking

Das Matchmaking ist eines der wesentlichen Funktionen des Systems. Es gewährleistet die Zusammenführung zweier Elternpaare, deren Kinder ein ähnliches Krankheitsprofil mit anderweitigen Eigenschaften (bspw. bestimmte sportliche Aktivitäten) aufweisen. Dadurch sollen Eltern unter anderen Anregungen erhalten, wie sie ihre Kinder besser zum Blutzuckermessen motivieren können.

Exit: Das Zusammenbringen von Elternpaaren unter Berücksichtigung der Kind-Profile lässt sich ohne Probleme durchführen. Konkret bedeutet das, dass der Matchmaking

Algorithmus fähig ist unter Berücksichtigung des Diabetes-Typs des Kindes, der sportlichen Aktivität (welcher Sport wird genau ausgeübt?), der parallelen Erkrankungen (die zumeist chronischer Natur sind, oder vielleicht sogar durch Diabetes bedingt sind) eine Zusammenführung der Eltern zu gewährleisten. Diese Auflistung entspricht nicht einer endgültigen Liste. Es können im Verlauf des Projektes weitere Merkmalsausprägungen ermittelt werden. Da ein Matchmaking auf hundertprozentiger Übereinstimmung der Parameter schwer zu bewerksteligen ist, oder vielleicht sogar unvorteilhaft ist, sollte man hier auf eine Übereinstimmung von 75% hinarbeiten. Es kann sich auch im Verlauf des Prozesses herausstellen, dass absolute Werte keine qualitative Lösung darstellen. Darauf aufbauend muss dann eine anderweitige Lösung entwickelt werden, beispielsweise in Form von Gewichtungen einzelner Parameter.

Fail: Ein Zusammenbringen von Elternpaaren unter Berücksichtigung der Kind-Profile ist nicht möglich. Dies kann mehrere Gründe haben. Ein negativer Anwendungsfall wäre, wenn ein Matchmaking positiv verläuft, die gestellte Fragen jedoch in genau jenen Bereich fällt, in denen sich keine Übereinstimmung finden lässt. Genauer gesagt, es lässt sich nicht verhindern, dass eine Frage in die 25% nicht Übereinstimmung fällt. Unterbinden ließe sich das, indem die Eltern lediglich ihr Anliegen als Freitext formulieren können. Die Parameter, die für das Matchmaking relevant sind, werden dem Profil des Kindes entnommen und können nur über eine vordefinierte Auswahl getätigt werden. Auch die eigentliche Frage kann über das Integrieren von Schlagwörtern ein Matchmaking-Verfahren vereinfachen.

Fallback: Sollte ein Matchmaking nicht implementiert werden können, bleibt das Forum dennoch funktionsfähig. Fragen können weiterhin publiziert und beantwortet werden. Es besteht jedoch keine Möglichkeit der Verknüpfung von Elternpaaren mit ähnlichen Kind-Profilen. Dies resultiert darin, dass augenscheinlich auf bestimmte Eltern zugeschnittene Fragen diesen nicht mehr priorisiert präsentiert werden.
Eine Alternative kann in Form von Topics umgesetzt werden. Hierbei wird kein automatisiertes Matching vollzogen, sondern die Eltern wären angehalten Topics zu abbonieren, um bestimmte Themenbereiche priorisiert präsentiert zu bekommen.

1.7.3 Unzureichende Kenntnisse in der Android Entwicklung

Das Team hat keine Vorkenntnisse in der Entwicklung von Applikationen für das Android Betriebssystem. Da mindestens ein Client als Teil des Systems für das Android Betriebssystem implementiert wird, muss das Team sich in die Android-spezifische Entwicklung einarbeiten.

Exit: Bis zum 11.05.2015 (Vorstellung der PoCs) müssen

- fünf Activities und deren Interaktion(Aufruf einer neuen Activity, Datenverkehr zwischen Activities) und
- die persistente Datenspeicherung im lokalen Speicher eines Android-Endgeräts

implementiert werden können.

Fail: Ist es dem Team bis zum oben genannten Termin nicht möglich grundlegende Teile des Android Betriebssystem (die oben genannten Kriterien) implementieren zu können, gilt das PoC als gescheitert.

Fallback: Als Fallback kann der Client als Java Desktop-Anwendung oder Browser-Applikation implementiert werden, um die Funktionalität vorstellen zu können. Zur Erreichung der strategischen Ziele ist weder die Lösung einer Desktop-Applikation, noch einer Browser-Applikation sinnvoll.

1.7.4 Client/Server Kommunikation

Die Kommunikation zwischen den Android Clients und dem NodeJS Server muss gegeben sein, damit das System funtkionieren kann. Ohne diese Kommunikation kann lediglich eine lokale Applikation auf den Clienten implementiert werden, die für die Erreichung der strategischen Ziele nicht ausreicht.

Exit: Vom Android Client muss per GET und POST Methoden via HTTP auf den Server zugeriffen und Daten vom Server angenommen werden können.

Fail: Wenn keine Kommunikation zwischen Clients und Server hergestellt bzw. implementiert werden kann, gilt das PoC als gescheitert.

Fallback: Im Falle eines Fails gibt es kein Fallback und das Projekt gilt als gescheitert, denn die Implementierung des Client/Server-Paradigma ist für das Erreichen der strategischen Ziele unabdingbar.

1.8 Dokumentation der Proof Of Concepts

1.8.1 Fehlende Kenntnisse in der Android-Entwicklung

Die Einarbeitung in die Entwicklung von Android Applikationen in Java wurde mit diesen Tutorials (Google a) begonnen. Für den Einstieg und die ersten Schritte wurde

eine Beispiel-Applikation implementiert. Über diesen Weg wurde sich mit der Erstellung von Layouts via *XML*, Implementierung von *Activities* und deren Kommunikation über *Intents* auseinandergesetzt. Insgesamt wurden fünf *Activities* implementiert, die teilweise als Grundlage für die anderen Proof Of Concepts dienen. Hinzu kommt, dass für das Proof Of Concept *Client/Server-Kommunikation* (Siehe Kapitel 1.8.2) letztlich mehr Wissen über die Entwicklung von Android Applikationen notwendig war. Daher kann man es auch als Teil dieses Proof Of Concept verstehen.
Folgende fünf *Activities* wurden implementiert:

- Main
  - Besteht aus mehreren Buttons, die auf weitere *Activities* führen. Dadurch wurde der Umgang mit dem Start neuer *Activities* mittels Intents (Google f) klar.
- ClientServerCommunication
  - In dieser *Activity* wird das Proof Of Concept *Client/Server-Kommunikation* dargestellt. Der Zugriff auf Textfeldinhalte und das ändern dieser Inhalte über vordefinierte IDs der Elemente war dafür nötig. Weitere Beschreibung der Implementation im Abschnitt *Client/Server-Kommunikation* (Siehe Kapitel 1.8.2).
- DataStorage
  - Hier wird die persistente Datenspeicherung im internen Speicher geprüft. Zusätzlich wird diese zur Überprüfung auch ausgelesen und ausgegeben. Dafür wurden die Klassen *FileInputStream* (Google c) und *FileOutputStream* (Google d) genutzt.
- ActivityCommunication
  - Um Daten zwischen *Activities* zu übertragen wurde ein Textfeld implementiert, dessen Inhalt nach Klick auf den Send Button an eine weitere *Activity* gesendet und dort angezeigt wird.
- DisplayMessage
  - Diese *Activity* zeigt den Inhalt des Textfelds aus der *ActivityCommunication Activity*, um die Kommunikation zwischen den *Activities* deutlich zu machen.
- InsulinCalculatorActivity
  - Hier findet die Berechnung der Insulineinheiten statt. Über Textfelder vom Typ *Number* lassen sie alle nötigen Werte zur Berechnung sowohl des Basis-, als auch des Bolusinsulin berechnen.

1.8.2 Client/Server-Kommunikation

Server

Für die Kommunikation zwischen Client und Server wurde erst ein Server mit *Node.js* (Joyent) implementiert. Dafür musste im Vorfeld die *Node.js* Plattform inklusive dem Package Management System *npm* (Node Package Manager) installiert werden. Zur Auswahl stehen der Download über die Homepage oder eine Installation über einen Package Manager wie *Homebrew* (Howell) oder *MacPorts* (TheMacPortsProject). Zur Vereinfachung der Serverimplementierung wurde das Framework *Express* (ExpressJS) gewählt. *Express* bietet das Paket *express-generator* zur Initialisierung an. Damit lässt sich das Grundgerüst einer Applikation automatisch generieren. Anschließend wurde die Ressource */poc* mittels einer Route implementiert, die auf die Methoden *GET* und *POST* via *HTTP* reagiert.

- GET
  - Der Server sendet ein Test-JSON-Objekt zurück. Dieses JSON-Objekt soll vom Client dargestellt dargestellt werden. Somit lässt sich überprüfen, ob die Kommunikation erfolgreich war. Zudem kann man erkennen, ob Daten auf dem Kommunikationsweg verloren gegangen sind, indem man das empfangene Objekt auf dem Client mit dem versandten Objekt auf dem Server vergleicht. Dafür wird das gesendete Objekt auf der Konsole ausgegeben.
- POST
  - Die empfangenen Daten im JSON-Format werden auf der Konsole ausgegeben. Auch hier lässt sich überprüfen, ob die Kommunikation erfolgreich war und ob die Daten vollständig und unverfälscht angekommen sind. Außerdem sendet der Server eine Nachricht über den Erfolg als String per *response* zurück.

Client

Im *Fehlende Kenntnisse in der Android-Entwicklung* Proof Of Concept (Siehe Kapitel 1.8.1) wurde die Implementierung eines Android Clients mit mehreren *Activities* beschrieben. Dieser dient als Grundlage für dieses Proof Of Concept. Für die Überprüfung des Proof Of Concepts wurde eine *Activity ProofOfConceptActivity* implementiert, die aus zwei Buttons, einem TextView und zwei EditText-Views besteht. Über den Get-Button wird per *HTTP* die Methode *GET* auf der Ressource */poc* auf dem Server ausgeführt. Die Antwort des Servers wird im darunter liegenden TextView angezeigt. Der Put-Button führt per *HTTP* die Methode *POST* auf der Ressource */poc* auf dem Server

aus und sendet den Inhalt der beiden EditText-Views als *JSON* Objekt. Im TextView wird die Antwort des Servers angezeigt. Für die Implementierung der *HTTP* Methoden war die Einarbeitung in asynchrone Tasks notwendig, damit das Interface während der Client/Server-Kommunikation nicht blockiert wird. Dafür wurde die abstrakte Klasse *AsyncTask* (Google b) genutzt. Für die einzelnen Methoden wurde jeweils eine Helferklasse implementiert, die die abstrakte Klasse *AsyncTask* erweitern. Für die eigentliche Kommunikation zum Server wurde die abstrakte Klasse *HttpURLConnection* (Google e) genutzt. Da die Daten bei der Kommunikation in Form eines *ByteStream* übertragen werden, wurde die Helferklasse *Stream* implementiert, mit der sich aus dem *ByteStream* lesen und schreiben lässt. Für die Erzeugung von Objekten im *JSON* Format wurde die Klasse *JSONObject* (Google g) genutzt.

1.8.3 BE/KE-Werte

Für die Überprüfung dieses Proof Of Concepts wurde ein weiterer *Node.js* Server mit dem Framework *Express* implementiert. Weiterhin wurde ein dokumentenbasierte Datenbank mit *MongoDB* (MongoDB b) erstellt und dort die Testdatensätze abgespeichert. *Mongoskin* (kissjs.org) dient als *Middleware*, um die Datenbankzugriffe zu vereinfachen. Als Test bzw. Überprüfung wurde ein Javascript Programm implementiert, welches durch die *setInterval()* Methode und das *request* (Request) Modul 50 Anfragen in einer Minute an den Server stellt. In der Konsole wird jede Anfrage ausgegeben und über einen Zähler iteriert. Somit lässt sich nachvollziehen, ob das Proof Of Concept erfolgreich war.

1.8.4 Berechnung der Insulineinheiten

Zur Berechnung der Insulineinheiten wurden zwei Berechnungen implementiert:

- Basisinsulin
  - Über zwei Textfelder lassen sich die Werte für den BE-Faktor und die Broteinheiten in der Mahlzeit eintragen. Durch den Button *Calculate* wird das Ergebnis der Berechnung in einem Textfeld angezeigt.
- Bolusinsulin
  - Für die Berechnung des Bolusinsulin sind nun drei Werte erforderlich. Der aktuelle Blutzuckerwert, der Zielwert und die Korrekturzahl. Auch hier lässt sich das Ergebnis der Berechnung durch den Button *Calculate* in einem Textfeld anzeigen.

1.8.5 MatchMaking Algorithmus

Es wurden 150 Testdatensätze erstellt, die per Algorithmus mit einen Testdatenobjekt verglichen werden. Da durch den fehlenden Kontakt zu den Eltern als Stakeholder die Gewichtung der Vergleiche der einzelnen Merkmale sehr unklar war, wurden eigene Kriterien entworfen, um die Funktionalität des Algorithmus zu testen. Der Algorithmus verglich ein Array mit zwei Strings des Testdatenobjekts mit den Arrays der anderen 150 Testdatensätze. Wurden diese zwei Strings in den Arrays der Testdatenobjekte gefunden, hat der Algorithmus die ID dieser Testdatenobjekte zurückgegeben. Dafür musste eine Aggregation-Pipe zur Datenbankabfrage modelliert werden, da sich herausstellte, dass diese Art von Abfrage in nicht-relationalen Datenbanken komplizierter zu modellieren ist:

```
1  var a = {'$match': {'$or': [{'krankheiten': 'Stinken'},
2          {'krankheiten': 'Grippe'}]}};
3  var b = {'$unwind': '$krankheiten'};
4  var c = {'$project': {'c': {'$concat':
5                              [{'$cond':[{'$eq' : ['$krankheiten', '
                                   Stinken' ]},'Stinken','']},
6                              {'$cond':[{'$eq' : ['$krankheiten', 'Grippe'
                                   ]},'Grippe','']}]}}};
7  var d = {'$group': {'c': {'$addToSet': '$c'}, '_id': '$_id'}};
8  var e = {'$match': {'c': {'$size': 3}}};
9  var f = {'$project': {'_id': '$_id'}};
10
11 db.collection('sick').aggregate([a,b,c,d,e,f], function(err, result) {
12   //console.log(result);
13   var ids = result.map(function(obj) { return obj._id; });
14   //console.log(ids);
15   db.collection('sick').find({'_id': {'$in': ids}}).toArray(function(err
         , docs) {
16     console.log(docs);
17   });
18 });
```

Listing 1.1: MatchMaking Proof Of Concept

1.9 Systemarchitektur

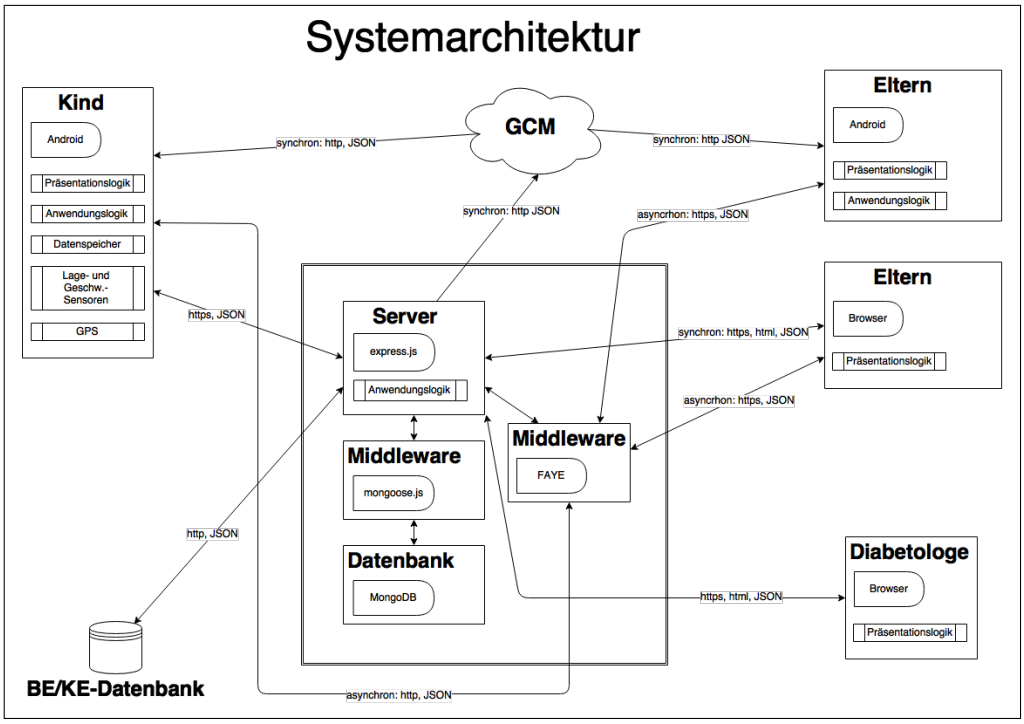

Abbildung 1.2: Architekturdiagramm erste Version

Für die Wahl der Architektur müssen die einzelnen Komponenten und deren Kommunikation untereinander betrachtet werden. Im folgenden wird eine Abwägung der gewählten Technologien beschrieben.

1.9.1 Server

Der Server muss über eine angebundene Datenbank verfügen, in der die Profile der Kinder inklusive der Logbucheinträge und alle Inhalte des Forums gespeichert werden können. Außerdem sollen die Daten der externen Nahrungsmittel-Datenbank archiviert und ggf. mit weiteren Informationen (Bsp.: Bilder der Nahrungsmittel) angereichert werden.

Technische Umsetzung

Für die technische Umsetzung des Servers können viele Programmiersprachen genutzt werden (z.B.: Java, Ruby, PHP, Python, Javascript). Mit allen genannten Programmiersprachen lassen sich die Anforderungen erfüllen. Das Team hat sich für die Implementierung mit JavaScript via dem Framework NodeJS entschieden, da diese Plattform durch ihre unkomplizierte Struktur und Skalierbarkeit sich sehr gut für das System

eignet. Zudem hat unser Team bereits Erfahrung mit JavaScript/NodeJS im Modul „Web-basierte Anwendungen 2: Verteilte Systeme“errungen, wodurch Zeit in der Einarbeitung gepart werden kann.
Im NodeJS Ökosystem werden viele Frameworks zur Serverimplementation angeboten. Koa, Hapi und Express sind drei der meist verbeitesten Frameworks. Sie bieten alle die notwendigen Funktionen, um den Server nach oben genannten Anforderungen zu implementieren. Da das Team im Modul „Web-basierte Anwendungen 2: Verteilte Systeme“bereits mit Express gearbeitet hat, wurde sich für die Implementation des Servers für dieses Framework entschieden.

Datenbank

Für die persistente Datenspeicherung serverseitig wurde die dokumentenbasierte Datenbank MongoDB gewählt. Grund dafür ist einerseits ihre Performanz und andererseits die Syntax des Datenformats (BSON), in der die Dokumente in der Datenbank gespeichert werden. Wie in Abbildung 1.3 zu sehen ist, haben Dokumente in MongoDB die gleiche Syntax wie JavaScript Objekte. Da unser BackEnd mit JavaScript realisiert wird bietet sich MongoDB an, da durch die gleiche Syntax Daten nicht noch in ein anderes Format umgewandelt werden müssen. Eine relationale Datenbank kommt aufgrund ihrer eingeschränkten Skalierbarkeit und der starken Indexierung in diesem Projekt nicht in Frage.

Integrität der Daten

Da alle Daten personenbezogen sind, müssen wir in der REST API alle Endpoints vor Zugriffen von nicht berechtigten Nutzern schützen. Die in Browser-Applikationen üblich verwendeten Sessions sind nicht zu gebrauchen, da sie gegen die Zustandslosigkeit einer REST-Architektur verstoßen, somit also nicht REST-konform sind. Eine Alternative sind Tokens. Bei jedem Request von Clients an den Server wird ein eindeutiger und verschlüsselter Token mitgesendet, anhand dem serverseitig überprüft werden kann, ob der Client die Berechtigung hat, geforderte Ressource zu erhalten. Dafür werden JSON Web Tokens(Auth0) implementiert.

1.9.2 Datenformat

Die zwei meist verbreitesten Datenformate in verteilten Systemen sind JSON und XML. Die Wahl fiel auf JSON, da JSON bei gleichem Informationsinhalt eine geringere Datenmenge vorweist. Außerdem lässt sich, wie auch bei der Wahl der Datenbank, JSON sehr gut durch die gleiche Syntax wie Javascript Objekte (wie der name Javascript Object Notation auch zeigt) in Javascript weiterverarbeiten. Clientseitig, auf Java ba-

Abbildung 1.3: BSON Syntax (MongoDB (a))

sierend, lässt sich JSON ebenfalls gut verarbeiten, da es diverse Bibliotheken gibt, die aus JSON Java-Objekte erzeugen lassen.

1.9.3 Mobiler Client

Zur Wahl stehen die drei bekanntesten und meist genutzten Plattformen für Smartphones: Android, iOS und Windows Phone. Unser Team hat keine Erfahrung mit der Implementierung aller genannten Plattformen doch sind bereits Java Kenntnisse aus den Modulen „Algorithmen und Programmierung“und „Computergraphik und Animation“vorhanden. Zudem wurde die Entwicklung in Java für Android im Rahmen von „Entwicklungsprojekt interaktiver Systeme “vorgegeben. Aus diesen Gründen wird der Client für Android in Java implementiert.

1.9.4 Browser-Applikationen

Die Applikationen im Browser werden mit einer Kombination aus HTML, CSS und JavaSCript realisiert. Es existieren unzählige Frameworks zur Implementation von Browser-Applikationen. Die Realisierung des Mobilen Client steht im Vordergrund, da die Zeit im Rahmen vom Modul „Entwicklungsprojekt interaktiver Systeme “nicht ausreicht.

1.9.5 Protokolle

Als Übertragungsprotokoll zur Kommunikation zwischen den Systemkomponenten wurde HTTP/HTTPS (HyperText Transfer Protocol) gewählt, da auf dem Server eine REST-Architektur vorgesehen ist und sich JSON-Objekte via HTTP/S übertragen lassen. Auch der Zugriff auf den externen Server zur Abfrage der Kohlenhydrate geschieht per HTTP.

1.9.6 Synchrone Kommunikation

Da die Daten der Nutzer von mehreren Akteuren (Kinder, Eltern und Diabetologen) manipuliert werden, sind eindeutig identifizierte Ressourcen von Vorteil. Das gilt ebenfalls für das Forum, in denen die Eltern sich austauschen können. Dies führt zu der Entscheidung die Service-Architektur als synchrone Kommunikation im REST-Stil zu realisieren. Als Alternative kann SOAP genannt werden. SOAP nutzt jedoch XML an Stelle von JSON als Datenformat. Zudem ist der Implementierungsaufwand von SOAP deutlich höher und die sehr enge Bindung zwischen Client und Server ist kritisch zu sehen. Jede Änderung beim Server muss auch auf dem Client angepasst werden. Weiterhin entsteht bei SOAP ein großer Overhead wodurch die Skalierbarkeit bei einer REST-Architektur deutlich höher ist.

1.9.7 Asynchrone Kommunikation

Das Kommunikationsparadigma der asynchronen Kommunikation wird im System dazu verwendet die Logbucheinträge der Kinder im Browser der Eltern anzuzeigen, damit die Ressource nicht manuell wärhend der Nutzung abgerufen werden muss. Des weiteren dient es dem gleichen Zweck bei Benutzung des Forums. Neue Topics und Kommentare werden während der Nutzung durch sowhl durch Publich/Subscribe, als auch GCM (Google Cloud Messaging) angezeigt.

1.9.8 Änderung am 17.06.2015

Eltern sollen über die Logbucheinträge ihrer Kinder asynchron, sowohl auf dem Client als auch im Browser, informiert werden. Zu Beginn wurde diese Kommunikation mit dem Publish/Subscribe Messaging System Faye für den Browser und Google Cloud Messaging für den Client modelliert. Weiterhin sollen Eltern im Forum über diese Kommunikationsarten auf Forenbeiträge und abbonierte Topics des Forums benachrichtigt werden. Die Implementation zweier Technologien zur Erreichung eines Ziels ließ sich einerseits durch die Kompatibilitätsprobleme von Faye und Betriebssystemen mobiler Endgeräte (Android, iOS) und andererseits fehlender Funktionalität von Google Cloud Messaging (Multicast über Topics, Browsersupport) nicht vermeiden. Während der Entwicklung erschien eine neue Version von Google Cloud Messaging, die die Unterstützung von Browsersupport, iOS Betriebssystemen und Multicast an Topics eingeführt hat. Dadurch lassen sich die Anforderungen durch Implementation einer Technologie erfüllen. Aus diesem Grund wurde Faye aus der Systemarchitektur entfernt. Daraus resultiert ein neues Architekturdiagramm 1.4

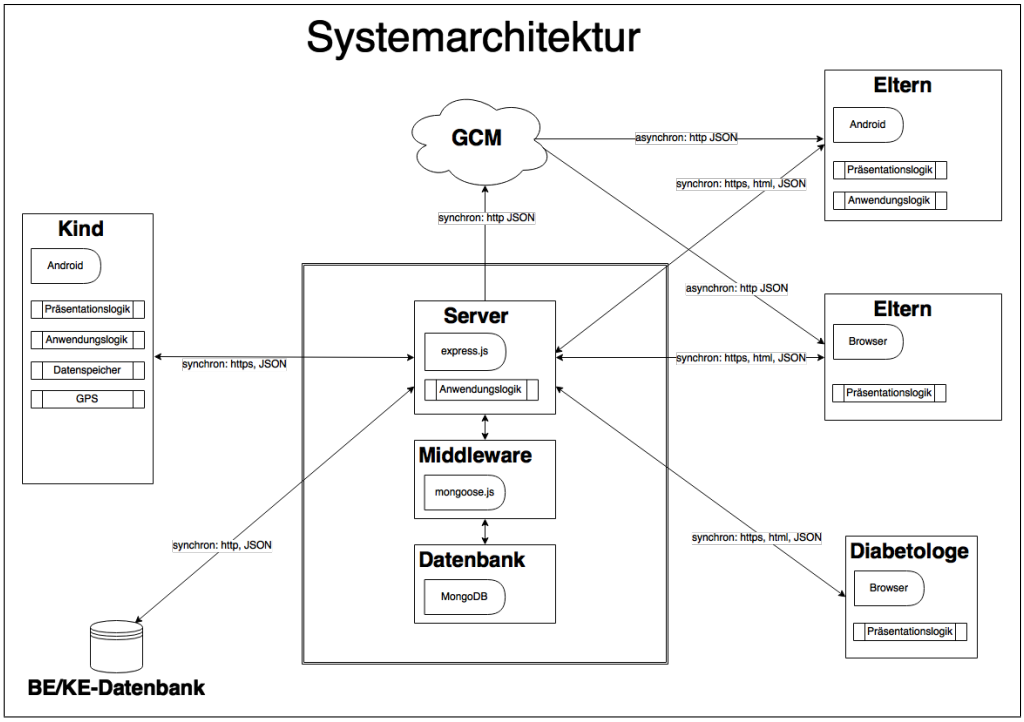

Abbildung 1.4: Architekturdiagramm überarbeitete Version

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

Auth0
AUTH0: *JSON Web Tokens*. http://jwt.io/. – zuletzt gesichtet am 18.06.2015

Ayogo-Health-Inc
AYOGO-HEALTH-INC: *Monster Manor*. http://ayogo.com/blog/monster-manor/. – zuletzt gesichtet am 17.04.2015

Bowley
BOWLEY, Chris: *Insulin Calculator*. https://itunes.apple.com/de/app/insulin-calculator/id320616301?mt=8. – zuletzt gesichtet am 17.04.2015

Cougar-Media
COUGAR-MEDIA: *BE Finder*. https://itunes.apple.com/de/app/be-finder/id684244169?mt=8. – zuletzt gesichtet am 17.04.2015

Diabetes-Hilfe
DIABETES-HILFE, DiabetesDE D.: *Deutscher Gesundheitsbericht Diabetes 2015: Die Bestandsaufnahme*. http://www.diabetesde.org/fileadmin/users/Patientenseite/PDFs_und_TEXTE/Infomaterial/Gesundheitsbericht_2015.pdf. – zuletzt gesichtet am 28.04.2015

ExpressJS
EXPRESSJS: *Express*. http://expressjs.com/. – zuletzt gesichtet am 08.05.2015

Google a
GOOGLE: *Android Training*. http://developer.android.com/training/index.html. – zuletzt gesichtet am 08.05.2015

Google b
GOOGLE: *AsyncTask*. http://developer.android.com/reference/android/os/AsyncTask.html. – zuletzt gesichtet am 08.05.2015

Google c
GOOGLE: *FileInputStream*. http://developer.android.com/reference/java/io/FileInputStream.html. – zuletzt gesichtet am 08.05.2015

Google d
GOOGLE: *FileOutputStream*. http://developer.android.com/reference/java/io/FileOutputStream.html. – zuletzt gesichtet am 08.05.2015

Google e
GOOGLE: *HttpURLConnection*. http://developer.android.com/reference/java/net/HttpURLConnection.html. – zuletzt gesichtet am 08.05.2015

Google f
GOOGLE: *Intent*. http://developer.android.com/reference/android/content/Intent.html. – zuletzt gesichtet am 08.05.2015

Google g
GOOGLE: *JSONObject*. http://developer.android.com/reference/org/json/JSONObject.html. – zuletzt gesichtet am 08.05.2015

Howell
HOWELL, Max: *Homebrew*. http://http://brew.sh/. – zuletzt gesichtet am 08.05.2015

Joyent
JOYENT, Inc.: *Node.js*. https://nodejs.org/. – zuletzt gesichtet am 08.05.2015

kissjs.org
KISSJS.ORG: *Mongoskin*. https://github.com/kissjs/node-mongoskin. – zuletzt gesichtet am 08.05.2015

Mayhew
MAYHEW, Deborah J.: *The Usability Engineering Lifecycle*. http://www.thinkbrownstone.com/wp-content/uploads/2010/12/usability-engineering-lifecycle_v3-copy2.jpg. – zuletzt gesichtet am 24.04.2015

Mayhew 1999
MAYHEW, Deborah J.: *The Usability Engineering Lifecycle: A Practitioner's Guide to User Interface Design*. 1999

Medhelp
MEDHELP: *Sugar Sense - Diabetes App*. https://itunes.apple.com/de/app/sugar-sense-diabetes-app-glucose/id880725347?mt=8. – zuletzt gesichtet am 17.04.2015

MongoDB a
MONGODB: *BSON Syntax*. http://docs.mongodb.org/manual/_images/crud-annotated-document.png. – zuletzt gesichtet am 24.04.2015

MongoDB b
MONGODB, Inc.: *MongoDB*. https://www.mongodb.org/. – zuletzt gesichtet am 08.05.2015

mySugr
MYSUGR: *mySugr Diabetes Tagebuch*. https://mysugr.com/de/apps/. – zuletzt gesichtet am 17.04.2015

Nielsen 1994
NIELSEN, Jakob: *Usability engineering*. Elsevier, 1994

Request
REQUEST: *Request*. https://github.com/request/request. – zuletzt gesichtet am 08.05.2015

Sanofi-Aventis
SANOFI-AVENTIS: *Mission T1D*. https://itunes.apple.com/gb/app/mission-t1d/id912339081?mt=8. – zuletzt gesichtet am 17.04.2015

Statista a
STATISTA: *Anzahl der minderjährigen Kinder in Deutschland in den Jahren 2000 und 2010 nach Alter (in Tausend)*. http://de.statista.com/statistik/daten/studie/197780/umfrage/minderjaehrige-kinder-in-deutschland-nach-alter/. – zuletzt gesichtet am 28.04.2015

Statista b
STATISTA: *Smartphone-Nutzung durch Kinder und Jugendliche in Deutschland im Jahr 2014 nach Altersgruppen*. http://de.statista.com/statistik/daten/studie/1104/umfrage/smartphone-nutzung-durch-kinder-und-jugendliche-nach-altersgruppen/. – zuletzt gesichtet am 28.04.2015

TheMacPortsProject
THEMACPORTSPROJECT: *MacPorts*. https://www.macports.org/. – zuletzt gesichtet am 08.05.2015