Technology
Arts Sciences
TH Köln

Projektbegründungen

“MDKS”

im Rahmen der Veranstaltung
Entwicklung Interaktiver Systeme

Veranstaltung betreut von
Prof. Dr. Kristian Fischer | Prof. Dr. Gerhard Hartmann

im
WS 2015/2016

verfasst von
Kevin Apitz | Alexander Miske

Gruppe betreut von
Sheree Saßmannshausen | Ngoc-Anh Dang

Inhaltsverzeichnis

1. Einleitung
   - 1.1. Problemstellung
   - 1.2. Erläuterung der Inhalte bzw. Bezug auf dieses Dokument
2. Zielhierarchie
   - 2.1. Strategische Ziele
   - 2.2. Taktische Ziele
   - 2.3. Operative Ziele
3. Marktrecherche
4. Alleinstellungsmerkmale
5. Betrachtung des Nutzungskontextes
   - 5.1. Prozessdarlegung
6. MCI-Vorgehen
   - 6.1. Abwägung der MCI-Vorgehensmodelle
   - 6.2. Abwägung der Methoden
   - 6.3. Verstehen und Beschreiben des Nutzungskontextes
   - 6.4. Spezifizieren der Nutzungsanforderungen
   - 6.5. Entwerfen der Gestaltungslösungen
     - 6.5.1. papierbasierte vs. computerbasierte Prototypen
     - 6.5.2. Techniken des Prototyping
   - 6.6. Testen und Bewerten der Gestaltung
   - 6.7. Benutzermodellierung
   - 6.8. Aufgabenmodellierung
   - 6.9. Problemszenario
   - 6.10. Use Cases
   - 6.11. Entwerfen der Prototypen
     - 6.11.1. Gestaltungsgrundsätze
   - 6.12. Evaluierung
7. Architektur
   - 7.1. Systemarchitektur
   - 7.2. Komponenten

7.3. Server
7.4. Middleware
7.4.1. Middlewaresysteme
7.4.2. Faye vs GCM vs RabbitMQ
7.5. Datenbank
7.5.1. realtional vs dokumentenorientiert
7.6. Webservice
7.7. Synchrone und asynchrone Nachrichten
7.8. Anwendungslogik
7.9. Ressourcen
7.10. Topics
7.11. Datenrepräsentation
7.12. Interaktionen zwischen den Komponenten
8. Risiken
8.1. Asynchrone Kommunikation über das System nicht möglich
8.2. Benachrichtigungen zu Ereignissen im System nicht möglich
8.3. Implementierung eines redundanten Serversystems
8.4. Datenaustausch zwischen Client und Server ist nicht möglich
8.5. MCI-Vorgehensmodell sieht Iterationen vor
8.6. Datenschutz
8.7. Datensicherheit
8.8. Umsetzung der Medikationsüberprüfung
8.9. Ausfall eines Teammitglieds
9. Proof of Concepts

Einleitung

Problemstellung

Erläuterung der Inhalte bzw. Bezug auf dieses Dokument

Zielhierarchie

Strategische Ziele

Taktische Ziele

Operative Ziele

Marktrecherche

Alleinstellungsmerkmale

Betrachtung des Nutzungskontextes

Um das Projekt zu konzipieren muss eine erste Domänenrecherche durchgeführt werden. Dabei wurden alle Stakeholder, Vorgänge, Metaphern sowie Paradigmen recherchiert. Die Ergebnisse sollen als Grundlage für weitere projektspezifische Entscheidungen dienen.

Das zu entwickelnde System soll in Krankenhäusern eingesetzt werden und die dort angewendete Medikation unterstützen. Dem Medikationsprozess liegt eine hohe Komplexität zugrunde, weshalb die Arzneimitteltherapie viele Risiken birgt (siehe Abbildung 4) und als "der fehleranfälligste Teil der medizinischen Versorgung"(Ärztliche Direktor der Frankfurter Uniklinik, Jürgen Schölmerich[1]) gilt. Dies resultiert aus den unterschiedlichen Personen, die in diesem Prozess involviert sind, der Vielzahl an Vorgängen, die für die Medikation notwendig sind, dem Daten- und Informationsaustausch unter den beteiligten Personen, sowie der Vielfalt an Medikamenten, die im Krankenhaus eingesetzt werden.

Die Etablierung eines Systems im Krankenhaus birgt außerdem noch rechtliche Risiken. Daher müssen weitere Untersuchungen im Bereich Datenschutz und Datensicherheit unternommen werden.

[1] http://www.n-tv.de/wissen/Medikamente-sind-grosse-Fehlerquelle-article14803246.html
Sichtung: 17.10.2015 18:15

(siehe Domänenrecherche)

Prozessdarlegung

Die Vorgänge des Medikationsprozesses wurden im Hinblick auf die Kommunikation der Stakeholder untersucht. Das Kommunikationsdiagramm baut auf den in der Domänenrecherche beschriebenen Medikationsprozess auf (siehe Domänenrecherche, Kommunikationsdiagramm).
Ausser der verbalen Kommunikation dienen Datensammlungen wie die Patienakte dem Personal als Kommunikationsmedium. Daher wurden Daten als weitere Instanz in dem Diagramm hinzugefügt, um die Kommunikation konkreter zu beschreiben. Des weiteren wurden Datenzugriffe des Arztes auf Medikamentendaten berücksichtigt.
Die Instanz Einrichtungen wurde herangezogen, um Schritte, die von dem Kommunikationsablauf abhängig sind, beschreiben zu können. So muss zum Beispiel das Pflegepersonal den Bestand in der Station prüfen bevor eine Kommunikation zum Apotheker stattfindet.
Aus dem deskriptiven Kommunikationsmodell wurde das präskreptive Modell entwickelt. Da in dem zu entwickelnden System die benötigten Patienten-, Verordnungs, und Medikamentendaten elektronisch vorliegen sollen, ist eine Kommunikation über die Patientenakte nicht mehr nötig. Der Kommunikationsprozess findet über das System statt. Dennoch können aus rechtlichen Gründen (siehe Domänenrecherche) nicht alle Kommunikationsprozesse über das System stattfinden. Daher müssen Apotheker und Ärzte weiterhin persönliche Gespräche mit den Patienten über Verordnungen von Medikamenten führen.
Die erarbeiteten Alleinstellungsmerkmale (Querverweis) sind in der Kommunikation zwischen Patienten und System ersichtlich. Der Patient kann Informationen zu seiner Medikation erhalten und Selbstmedikationen (siehe Domänenrecherche) über das System bestätigen.

MCI-Vorgehen

Das Vorgehen aus der MCI Sicht ist für die Konzeption des Projekts grundlegend. Deswegen muss die Wahl des Vorgehensmodells ausreichend abgewägt werden. Die Argumente für die Wahl eines Vorgehensmodells müssen immer in Bezug zu dem vorhanden Problemraum stehen. Aus diesem Grund müssen die Ergebnisse aus der Untersuchung des Nutzungskontexts an dieser Stelle herangezogen werden.

Abwägung der MCI-Vorgehensmodelle

Aus dem Nutzungskontext ist ersichtlich, dass viele Benutzertypen im Medikationsprozess involviert sind. Diese haben verschiedene komplexe Aufgaben und Perspektiven, die maßgeblich entscheidend für die Entwicklung des Systems sind. So hat der Patient ein völlig anderes Interesse an dem System als das Pflegepersonal. Ein besonderer Fokus im Vorgehensmodell sollte daher der Benutzer mit seinen Aufgaben sein. Dennoch muss das System ein gutes Werkzeug sein, da der Medikationsprozess wichtige und risikoreiche Vorgänge beinhaltet.

Das Usage Centered Design, dass von Lockwood und Constantine 1996 entwickelt wurde, fokussiert sich auf den Verwendungszweck des zu entwickelnden interaktiven Systems. dieses System soll ein gutes Werkzeug für die Benutzer darstellen, wodurch die Betrachtung des eigentlichen Benutzers des Systems in den Hintergrund rückt.
Das User-Centered Design fokussiert sich dagegen auf den Benutzer und seine Bedürfnisse. Demnach muss der Benutzer des zu entwickelnden Systems stetig in den Entwicklungsprozess miteinbezogen werden.
Aus der Argumentation heraus sollte also von einem benutzungszentrierten Ansatz abgesehen werden und ein benutzerzentriertes Vorgehensmodell gewählt werden. Grund dafür ist vorwiegend die inhomogene Zielgruppe des Systems.

Einige Modelle geben dabei konkrete Methoden vor. Im Usability-Engineering nach Rosson und Carrol wird hauptsächlich mit Szenarien gearbeitet. Diese werden als Methode für jegliche Aktivitäten bei der Entwicklung des Systems vorgesehen.

Das Vorgehensmodell “Usability engeneering lifecycle” nach Deborah Mayhew ist iterativ und skalierbar und besteht aus drei Prozessbestandteilen. Zunächst müssen die Anforderungen der Stakeholder analysiert werden, die sich aus den User profiles und der Aufgabenanalyse ableiten lassen. Der nächste Schritt ist es aus den Anforderungen Prototypen zu entwickeln und Style-Guides festzulegen. Dieser Schritt muss solange evaluiert und getestet werden, bis die Prototypen valide sind und alle Probleme im Hinblick auf die präskreptive Aufgabenmodellierung beseitigt sind. Die Prototypen müssen abschließend dahingehend überprüft werden, ob alle Anforderungen adressiert werden. Der letzte Bestandteil des Vorgehensmodells ist es, das System zu installieren. Auch dieser Schritt sieht Iterationen vor. Das System muss mit Zielbenutzern evaliert und gegebenenfalls weiterentwickelt werden.
Auffallend bei diesem Vorgehensmodell ist besonders die Iterationen in allen qualitäts-relevanten Aktivitäten. Dies kann im Hinblick auf den Problemraum sehr

zeitaufwendig werden und ein echtes Risiko für die Realisierung des Projekts darstellen.

Das "Discount useability engeneering" Vorgehensmodell nach Nielson beschreibt, wie man mit Probanden durch die Methode des think alouds sein System mit kostengünstigem Aufwand Gebrauchstauglicher gestalten kann. Dabei werden Beobachtung angestellt, während die Probanden ihre Aufgaben mit einem Prototypen des Systems lösen. Die Evaluierung wird durch die 10 usability-heuristiken nach Nielson durchgeführt. Dadurch kann ein für die Benutzer und ihren Fähigkeiten angepasstes System konzipiert werden, wodurch die Gebrauchstauglichkeit erhöht wird und gleichzeitig das Risiko vermindert wird, dass die Anforderungen der Stakeholder nicht erfüllt werden. Vorteilhaft ist vorallem an diesem Modell, dass die Gebrauchstauglichkeit durch wenig aufwand gesteigert werden kann. Dazu werden konkrete Techniken vorgegeben, die im Entwicklungsprozess verwendet werden müssen.[2]

Aus dem Problemraum wird ersichtlich, dass die Aufgaben sehr komplex werden können und verschiedene Anwendungsfälle betrachtet werden müssen. Dabei ist es wichtig Unteraufgaben und Kommunikationsnetze aufzuzeigen. Durch Szenarien kann dies beschrieben werden. Dennoch ist es sinnvoll weitere Methoden zu den Szenarien zu verwenden, um die Aufgaben zu beschreiben. Das Vorgehensmodell sollte daher keine konkreten Methoden vorschreiben und modifizierbar sein. Dadurch kann nach Bedarf weiteres Wissen durch andere Methoden erarbeitet werden. Demnach sollten auch Iterationen im Vorgehensmodell einplanbar sein. Die Wahl fällt aufgrund der beschriebenen Umstände auf das ISO 9142 Teil 210 Vorgehensmodell. Dieses wird mit einigen Methoden modifiziert.

(siehe MCI-Rahmen)

Abwägung der Methoden

Die Methoden dienen uns für das Erarbeiten von Wissen für die Konzeption des Systems. Die ISO Norm nach der das Vorgehensmodell erstellt werden soll, legt keine Methoden fest. Daher müssen mögliche Methoden für die einzelnen Aktivitäten aus dem MCI-Rahmen gewählt werden. Die ersten Recherchen aus dem Nutzungskontext bieten uns auch an dieser Stelle Argumente zur Wahl von Methoden.

[2] http://www.nngroup.com/articles/discount-usability-20-years/

Verstehen und Beschreiben des Nutzungskontextes

Nach der Norm bestimmen die Benutzermerkmale, Arbeitsaufgaben und die organisatorische, technische und physische Umgebung den Kontext. Daher müssen zuerst die Benutzer identifiziert und analysiert werden. Eine geeignete Methode dafür ist die Stakeholderanalyse. Der Benutzer werden in primäre, sekundäre und teriäre Benutzergruppen aufgeteilt. Vorteilhaft ist hierbei, dass sofort ersichtlich ist welche Benutzer im Fokus stehen und das System häufig nutzen werden. Trotzdem können Nutzer, die zwar in Verbindung mit dem System stehen, aber es nicht häufig oder garnicht benutzen mit in den Entwicklungsprozess einbezogen werden.

Anschließend können User Profiles für diese Benutzer angelegt werden, die die Benutzermerkmale aufzeigen. Dies ist für unseren Problemraum sehr wichtig, da hier die Fähigkeiten und Einschränkungen der Benutzer ersichtlich werden, die für die Gestaltung eines barriefreien Systems von Bedeutung sind.

Aus den User Profiles können Personae entwickelt werden, die die Benutzerperspektive zum System verdeutlichen können.

Die Aufgaben der Benutzer sind verschieden. Diese unterscheiden sich in Komplexität und Häufigkeit der Durchführung. Des weiteren können mehrere Benutzer bei den Aufgaben involviert sein. Es bietet sich an Problemszenarien als Methode zur Beschreibung für die komplexen Aufgaben zu verwenden. Die Beschreibungen sind deskriptiv. Aus zeitlichen Gründen können nicht alle Aufgabenbereiche durch Szenarien abgedeckt werden, da das Schreiben von Szenarien zeitaufwendig ist.
Anschließend können Fehleranfällige Aufgabenbereiche spezifischer durch Use Cases betrachtet werden. Use Cases beschreiben Aufgaben genauer als Hierachical Task Analysen. Dabei sollen die Use Cases in narrativer Form erstellt werden, wodurch verschiedene Aspekte besser beschrieben werden können. Die Use Cases sollten vorallem auf folgende Fragestellungen eingehen:

- Wie häufig wird die Aufgabe durchgeführt?
- Welche Risikofaktoren gibt es bei der Durchführung?
- Welche Fertigkeiten und Fähigkeiten werden benötigt?
- Ist der Vorgang kontinuierliche oder wird dieser temporär immer wieder aufgegriffen?
- Welche Faktoren sind kritisch bei dem Prozess (Zeit, Informationen)?

- Welche Benutzer sind beteiligt?

Spezifizieren der Nutzungsanforderungen

Es ist wichtig, dass das deskriptive Model der Aufgaben ausführlich betrachtet wird, um funktionale sowie nicht-funktionale Anforderungen zu ermitteln. Die Use Cases als präskreptives Modell sollen die Anforderungen beinhalten. Eine mögliche Iteration wäre hierbei, den Nutzungskontext nochmals zu betrachten, um mögliche weitere Anwendungsfälle zu finden. Die Anforderungen werden demnach aus den Problemszenarien ermittelt. Allerdings sollte die Domänenrecherche auch für die Ermittlung der Anforderungen herangezogen werden, da eventuell nicht alle Aufgaben modelliert werden können oder wichtige Aspekte nicht in den Szenarien betrachtet wurden.

Entwerfen der Gestaltungslösungen

Es gibt zwei verschiedene Methoden zur Erstellung von Prototypen[3], die im folgenden diskutiert werden sollen.

papierbasierte vs. computerbasierte Prototypen

Der Vorteil von papierbasierten Prototypen sind, dass sie in Hand-Skizzen oder -Zeichnungen erstellt werden und deshalb weniger zeitaufwendig sind. Zudem sind sie leicht änderbar und auch zeitliche Abfolgen können über Storyboards dargestellt werden. Der Nachteil ist, dass keine Interaktionsmöglichkeit bestehen, wie bei den computerbasierten Prototypen. Im Hinblick auf den zeitlichen Rahmen des Projekts eignen sich daher papierbasierte Prototypen mehr. Die Erstellung von computerbasierten Prototypen könnte ein echtes zeitliches Risiko für das Projekt darstellen, da alle Prototypen immer wieder geändert und validiert werden müssen. Des weiteren können durch papierbasierte Prototypen besonders wichtige Aspekte im Interface hervorgehoben werden, indem andere Aspekte nicht in der Gestaltung miteinbezogen werden.

Techniken des Prototyping

Requirements Animation Zielt darauf ab, funktionale Anforderungen im frühen Projektphasen im Prototypen zu gestalten und diese mit den Stakeholdern zu validieren. Für unser Projekt eignet sich diese Variante sehr gut, da ein komplexes System zu erstellen ist. Daher sollten Fehler der Prototypen schon im frühen Stadium

[3] nach MCI-Draft von Gerhard Hartmann, S 500 20.2.2 Prototypen in der Mensch-Computer Interaktion

der Entwicklung erkannt und beseitigt werden. Um die Ergebnisse im Rahmen des Projektes im vollen Funktionsumfang präsentieren zu können, müssen die Prototypen vertikal entwickelt werden.

Testen und Bewerten der Gestaltung

Die erste Evaluierung der Prototypen soll mithilfe von Gestaltungsregeln durchgeführt werden. Es soll überprüft werden, ob die Prototypen Mängel aufweisen, die durch für das Projekt erstellten Gestaltungsregeln identifiziert werden sollen. Anschließend müssen die Mängel beseitigt werden und die Evaluierung muss erneut durchgeführt werden. Dieser Prozess soll wiederholt werden, bis keine Mängel mehr vorhanden sind und die Prototypen valide sind.
Der nächste Schritt ist es, die Prototypen mit einem Benutzer des Systems zu evaluieren. Ziel ist es die Evaluation formativ durchzuführen, um weitere Anforderungen zu erheben und Fehler zu beseitigen. Dabei soll ein qualitatives Verfahren verwendet werden, damit der Benutzer seine Meinung beim Testen frei äußern kann.

Benutzermodellierung

Der nächste Schritt ist zunächst die Stakeholder konkret zu benennen und sie in dem Nutzungskontext einzuordnen. Dazu werden die Ergebnisse aus der Domänenrecherche verwendet (siehe Stakeholder).

Um aus den ermittelten Stakeholdern Personae zu generieren, müssen User Profiles angelegt werden. An dieser Stelle kann der Aufwand immens werden, da gerade zu unzählige User Profile erstellt werden können. Daher muss hier eine Entscheidung getroffen werden, auf welche Benutzer wir uns besonders in diesem Projekt fokussieren wollen. Für die Benutzung des Systems sind bisher Patienten, Ärzte, Pflegepersonal und Apotheker vorgesehen. Die Hauptaufgaben im Medikationsprozess liegen bei den Ärzten und dem Pflegepersonal. Der Patient soll, wie in den Alleinstellungsmerkmalen beschrieben, mit in den Prozess einbezogen werden. Demnach soll bei dem weiteren Vorgehen der Benutzermodellierung der Fokus auf folgende Benutzertypen fallen:

- Patienten
- Ärzte
- Pflegepersonal

Die User Profiles wurden zunächst grob mit einigen Benutzermerkmalen und Ausprägungen angelegt. Nach weiterer Betrachtung des Nutzungskontexts wurden die User Profiles verfeinert. Es wurden nun weitere Ranges für das Alter der Benutzer angelegt, da die Merkmale sich dahingehend teilweise unterscheiden. Diese Änderung lieferte uns auch weitere Ausprägungen der Merkmale sowie Sichtweisen der Benutzer. Oft ist es der Fall, dass ältere Menschen Einschränkungen wie eine Sehschwäche aufweisen und eine andere Einstellung zu elektronischen Systemen und Technologien haben als jüngere Benutzer.

Um diese Unterschiede und Auswirkungen der Merkmale konkret in Szenarien zu beschreiben wurden zunächst Personas aus bestimmten Altersgruppen erstellt. Dabei wurde ein besonderer Fokus auf kritische Benutzermerkmale, wie die Einschränkung der Fähigkeiten oder Technologieängste gelegt. Um die Unterschiede der Patienten aufzuzeigen, wurden zunächst zwei Personas geschrieben, die als Patienten in den Szenarien verwendet werden sollen.

Aufgabenmodellierung

Grundlage für die deskreptive Aufgabenmodellierung sind die Personas. Um konkrete Probleme zu skizzieren wurde die Domänenrecherche als Quelle hinzugezogen. Die dort beschriebenen Risiken in der Medikation konnten für die Erstellung der Szenarien verwendet werden.

Problemszenario

todo wahl der szenarien

Use Cases

todo wahl der use cases und vorgehen

Entwerfen der Prototypen

Gestaltungsgrundsätze

Der Gestaltungsentwurf muss auf Regelwerken[4] basieren, die im Vorfeld festgelegt werden müssen und aus dem Problemraum heraus zu begründen sind.

[4] nach MCI-Draft von Gerhard Hartmann, S 442 19.1.3 Prinzipien

Evaluierung

Architektur

Systemarchitektur

Das System soll durch eine Multitierarchitektur realisert werden. Die Architektur setzt sich aus mehreren Schichten zusammen.

Der Client-Tier ist generell für die Realisierung der Benutzerschnittstellen zuständig. Über diese Schnittstellen sollen die Benutzer Zugriff auf die Systemfunktionen haben. Da die Benutzer aus Sicht der MCI unterschiedliche Merkmale, Aufgaben und Anforderungen besitzen müssen mehrere Schnittstellen implementiert werden. Die Zugriffsrechte sind je nach Benutzertyp eingeschränkt. Dementsprechend können Geschäftobjekte auch nur zum Teil bearbeitet werden. Die mobilen Clients sollen im gegensatz zu den lokalen Clients einen Teil der Anwendungslogik beinhalten.

Über die Steuerungsschicht oder auch Middleware genannt werden die Anfragen der Clients sowie die Nachrichten des Anwendungssystems zu den Clients verwaltet. Die Middleware dient der Übermittlung von Daten und asynchroner Nachrichten an die Komponenten des Systems.

Das Anwendungssystem oder auch die Logikschicht ist für die Bearbeitung der Geschäftobjekte zuständig. Automatische Prozesse werden hier ausgeführt. Dabei greift das System auf Informationen der Datenschicht zu.

Die Datenschicht speichert persistent alle Daten, die für die Medikation relevant und notwendig sind.

Komponenten

Die Systemarchitektur muss nun weiter definiert werden. Dazu ist es nötig, die Systemkomponenten im einzelnen zu betrachten und zu spezifizieren. Dafür müssen verschiedene Technologien abgewägt werden und Entscheidungen mit den bisherigen Ergebnissen aus dem MCI-Vorgehen herangezogen werden.

Server

Der Server kann durch verschiedene Technologien realisiert werden. Da wir im Verlauf des Studiums Erfahrungen mit NodeJS gesammelt haben, soll der Server mit NodeJS realisiert werden. Die Einarbeitung in andere Technologien könnte für das Projekt ein weiteres Risiko darstellen. Außerdem wird NodeJS ständig weiter entwickelt und die Technologie ist weit verbreitet. Dadurch ist die Anbindung verschiedener Middleware- und Datenbank-Systeme möglich. NodeJS schränkt uns daher in keiner Weise ein.

Clients

Die ermittelten Anforderungen zeigen, dass sowohl mobile als auch lokale Clients entwickelt werden müssen.

Als mobiles Endgerät soll ein Tablet eingesetzt werden. Alternativ könnten auch Smartphones verwendet werden. Allerdings bietet ein Tablet ein größeres Display, welches die Darstellung von vielen Informationen erleichtert.
Die meist verbreitetesten Betriebsysteme für mobile Endgeräte sind Googles Android OS und Apples iOS. Die native Programmiersprache von Android ist Java, wohingegen für iOS in Objective-C programmiert wird. Zusätzlich sind Frameworks vorhanden, die es ermöglichen die Apps in Javascript und Html zu programmeren. Die programmierten Anwendungen können dann durch das Framework für beliebige Betriebssysteme exportiert werden.
Eine wesentliche Rahmenbedingung im Projekt ist es, die Anwendungslogik in Java zu realisieren. Da das Team auch Erfahrungen in der Programmierung mit Java und dem Android SDK hat, werden die mobilen Clients für ein androidbasiertes Endgerät programmiert.

Die lokalen Clients sollen in Html/Javascript entwickelt werden.
todo… irgendiwe noch was

Middleware

Nach den Anforderungen soll entschieden werden, welches Middleware-System für das Projekt verwendet werden soll.

Middlewaresysteme

Grundsätzlich lassen sich die Middleware-Technologien in zwei Arten unterscheiden. Zum Einen gibt es kommunikationsorientierte Middleware-Technologien, die sich

durch entfernte Aufrufe durch z.B. Webservices oder nachrichtenorientierte Middlewares durch komplexe Warteschlangen klassiefiezieren. Zum Anderen gibt es anwendungsorientierte Middlware, die sich dadurch auszeichnen, dass sie die kommunikationsorientierte Middleware um eine Laufzeitumgebung erweitern[5].
Aus den Anforderungen ist ersichtlich, dass vorallem Nachrichten zwischen den Komponenten ausgetauscht werden müssen. Ein wesentlicher Aspekt ist, dass das MDKS asynchrone Nachrichten verteilen muss. Die Nachrichten richten sich dabei an verschiedene Empfänger, die je nach Inhalt der Nachricht addressiert werden sollen. Entfernte Aufrufe oder eine Laufzeitumgebung innerhalb der Middleware sind nicht nötig. Daher soll eine nachrichtenorientierte Middleware gewählt werden, die ein Publish-Subscribe-System unterstützt. Die Nachrichten sollen durch einen Broker verwaltet und an die Message Queues weitergeleitet werden, die wiederum die Nachrichten an die jeweiligen Empfänger weiterleiten. Die Nachrichten werden dann übermittelt, wenn die Empfänger erreichbar sind. Daher gehen die Nachrichten bei Verbindungsabbrüche mit den Clients nicht verloren. Sender und Empfänger können dadurch räumlich wie auch zeitlich entkoppelt werden.

Es gibt eine Reihe von Technologien, die ein Publish-Subscribe ermöglichen. Drei der Technologien sollen im folgenden im Hinblick auf das zu entwickelnde System abgewägt werden.

Faye vs GCM vs RabbitMQ

Faye ist ein Modul von NodeJS. Nachrichten können durch ein Pub-Sub-System asynchron per http-Protokoll übermittelt werden. Allerdings lassen sich die Message Queues nicht selbst verwalten und insgesamt sind die Möglichkeiten für eine individuelle Einrichtung der Middlware eingeschränkt. Ein weiterer negativer Aspekt ist, dass es keinen offiziellen Client für Java gibt, der die Nachrichten aus der Queue entgegen nimmt. Dies stellt für uns ein Problem dar, da die Clients in Java programmiert werden sollen.

Google Messaging Cloud ist eine weitere Lösung für ein Publish-Subscribe-System. die Daten werden in einer Message Queue, die sich in einer Cloud im Web befindet, aufbewahrt und von dort an die clients verschickt. Service ist einfach nutzbar und

[5] https://www.google.de/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&cad=rja&uact=8&ved=0CCAQFjAAahUKEwj8kbfKi47JAhUGliwKHQHHBoM&url=http%3A%2F%2Fwww4.in.tum.de%2Flehre%2Fseminare%2Fhs%2FWS0506%2Fmvs%2Ffiles%2FAusarbeitung-Krivoborodov.doc&usg=AFQjCNFv-bzMKHJ_YOY4dN99MuntIrCFzg&sig2=RlCoqwiv6p1xnWiwFxCu1w
Sichtung: 2.11.2015, 20:55

besonders praktisch für Androidapplikationen. Es ist aber zu beachten, dass die Daten aus dem MDKS dadurch an einen Drittanbieter verschickt werden müssen. Da die patientenbezogene Daten einem strengen Datenschutz unterstehen, ist die Weitergabe der Daten an dritte Personen kritisch zu betrachten.

Eine weitere Technologie ist RabbitMQ. RabbitMQ wird als eigenständiger Server auf einem System installiert und fungiert dabei als nachrichtenorientierte Middleware. Die Message Queues können manuel oder über die Clients angelegt und konfiguriert werden. Das Pub-Sub-System wird durch Exchanges und Routingkeys realisiert, wobei die Exchanges die Nachrichten an Queues weiterleiten und die Routingkeys angeben, welche Queue addressiert werden soll. RabbitMQ bietet für sämtliche Programmiersprachen und Clients eine Anbindung. Darunter sind auch Java, Javascript und NodeJS. Die Nachrichten werden über das Protokoll "AMQP" übermittelt.

Aufgrund der beschriebenen Aspekte von, fällt die Wahl auf RabbitMQ. Es ist aber zu beachten, dass keine Erfahrungen mit dieser Technologie im Projektteam vorhanden sind, wodurch ein Risiko entsteht. Daher sollte RabbitMQ und ein Publish-Subscribe mit dieser Technologie in einem "Proof-of-Concepts" getestet werden.

Datenbank

Eine der Anforderungen ist es, Daten persistent zu speichern, wodurch es notwendig ist ein DBMS zu implementieren. Aus dem gewonnen Wissen des Studiums sind uns zwei verschiedene Datenbanktypen bekannt. Im folgenden sollte deshalb abgewägt werden welcher Typ für das System und den Nutzungskontext besser geeignet ist.

realtional vs dokumentenorientiert

Relationale Datenbanken organisieren die Daten in Tabellen und Feldern, die in Beziehung miteinander stehen. Daher entstehen Referenzen in den Tabellen untereinander. Eine raltionale Datenbank folgt einem festegelgten Standard und die Abfragesprache ist einheitlich.
Dokumentenorientierte Datenbanken speichern die Daten in Containerobjekten mit bleibigen Feldern. Dieses Modell fokussiert sich vorallem auf Performance und Verfügbarkeit.
Aus der Domänenrecherche ist bereits ersichltich, dass im zu entwickelnden System besonders viele Abhängigkeiten in den Datenmodellen bestehen. So setzt sich eine Patientenakte aus Verordnungen, Patientendaten, Indikationen und

Untersuchungen zusammen. Diese Abhängigkeiten lassen sich idealerweise durch Beziehungen in relationalen Datenbank realisieren. Aus diesem Grunde fällt die Entscheidung auf ein realtionales DBMS. Die Realiesierung eignet sich vorallem mit MySQL, da die Implementation einfach ist und Anbindungen in sämtlichen Programmiersprachen durch die weite Verbreitung von MySQL möglich sind.

Es gilt zu testen, in wie weit NodeJS eine Implementierung von MySQl ermöglicht. Daher ist es notwendig ein "Proof of Concepts" für diese Technologie einzuplanen.

Webservice

Um Fehler bei der Medikation zu vermeiden, sind zusätzliche Informationen zu Medikamenten notwendig. Es muss festgestellt werden, ob verschriebene Medikamente kompatibel mit den bisherigen gestellten Verordnungen sind. Des weiteren muss geprüft werden, welche Wirkstoffe oder Speisen die Wirkung der Medikamente aufheben. Diese Informationen werden in der Regel von Fachkräften der Pharmazie auf Grundlage von Nebenwirkungsprüfungen und Studien erarbeitet und sind nicht frei im Internet zugänglich. Deshalb muss ein externer Webservice in das MDKS eingebunden werden, wodurch es möglich ist, Wechselwirkungsdaten und Daten über Risikofaktoren zu bestimmten Medikamenten zu erhalten. Die Daten können dann im NodeJS-Server für eine Medikationskontrolle verwendet werden.

Die Firma "ifap GmbH" bietet einen solchen Webservice an. Allerdings wurde uns ein Testzugang zu diesem verwehrt. Daher ist es notwendig einen Webservice zu simulieren, um die Anwendungslogik der Medikationskontrolle zu realisieren.

Insbesondere ist bei der Anbindung eines externen Webservice der Datenschutz zu beachten. Deshalb müssen die Anfragen auf den Webservice ohne die Weitergabe persönlicher Daten der Patienten realisiert werden und nur Medikamentennamen enthalten. Dadurch kann der Datenaustausch mit einem Drittanbieter mit Beachtung des Datenschutzes realisiert werden.

Synchrone und asynchrone Nachrichten

Innerhalb des MDKS müssen die Daten asynchron und synchron ausgetauscht werden. Nach den ermittelten Anforderungen müssen Erinnerungen und Benachrichtigungen über Änderungen in Form von Nachrichten an die Empfänger-Clients verschickt werden. Für solche Anwendungsfälle soll ein

Publish-Subscribe-System mit RabbitMQ genutzt werden. Die Nachrichten werden dadurch asynchron per AMQP-Protokoll verschickt.
Neben einer asynchronen Kommunikation sollen Daten auch synchron ausgetauscht werden. Ein Anwendungsfall ist z.B. ein Datenabruf von Verordnungen für die Erstellung von Medikationsplänen. Da bereits ein asynchrones Kommunikations-Paradigma mit RabbitMQ gewählt worden ist, soll die synchrone Kommunikation ebenfalls über RabbitMQ und Warteschlangen stattfinden. Daher muss von der Entscheidung den synchronen Datenaustausch mit dem http-Protokoll zu realisieren abgesehen werden. Gründe dafür sind vor allem eine Realisierung einer minimalistischen Architektur. Dafür müssen entsprechende Warteschlangen eingerichtet werden, die die Anfragen an den Server weiterleiten. Der Anwendungsserver soll die Anfragen erhalten und die gewünschten Daten aus der Datenbank holen und an die jeweilige Queue publishen, an die der Client als Consumer verbunden ist. Der Client wartet dabei nach dem senden der Anfrage auf die Antwort.

Anwendungslogik

Im folgenden soll die Anwendungslogik innerhalb der Komponenten konkret beschrieben werden. Dabei werden zunächst einzelne Komponenten betrachtet.

Android-Client

Die Primäre Aufgabe des Android-Clients soll es sein, Medikationspläne mithilfe der Daten aus den Patientenakten zu erstellen und eingehende Nachrichten der Middleware sowie ausgehende Nachrichten zur Middleware zu Verwalten und zu verarbeiten.

Für die Erstellung der Medikationspläne sind die Verordnungsdaten aus der Patientenakte notwendig. Der Medikationsplan eines Patienten unterscheidet sich vom Medikationsplan eines Krankenpflegers in der Zusammensetzung der benötigten Daten. Für den einzelnen Patienten sind nur die Daten der jeweiligen Patientenakte notwendig wohingegen ein Krankenpfleger für alle Patienten einer Station zuständig ist und daher sämtliche Verordnungsdaten aller Patienten der Station benötigt, um seinen Aufgaben nachzugehen. Der Medikationsplan muss chronologisch in Abhängigkeit von dem Applikationszeitpunkt der Verordnungen sortiert sein. Außerdem muss es für das Pflegepersonal möglich sein mit dem auf dem Device dargestellten Plan zu interagieren, um Verabreichungen zu dokumentieren. Die Dokumentation der Verabreichungen müssen persistent gespeichert werden.

Die Speicherung findet aber nicht auf dem Device statt, sondern auf der zentralen Datenbank des MDKS. Dadurch müssen Nachrichten zur Middleware geschickt werden, die dann vom NodeJS-Server abgegriffen werden. Da der Server die Daten in die Datenbank schreibt, müssen die Nachrichten zwingend die notwendigen Daten zur Speicherung der Verabreichung enthalten (siehe Artefakt Datenstrukuren).
Um Änderungen an Verordnungen für das Personal sowie für den Patienten kenntlich gemacht werden können muss ein Hintergrundservice in der Clientanwendung integriert werden, der einegehende Nachrichten der Middleware als eine Systembenachrichtigung auf der Statusleiste des Device ausgibt. Abhängig von dem Inhalt der Nachricht müssen die Medikationspläne angepasst werden. Daher soll der Hintergrundservice auch das Aktualisieren des Medikationsplans triggern, damit keine veralteten Daten auf dem Device angezeigt werden. Da die Medikationspläne individuell für den jeweiligen Benutzer erstellt werden müssen auf dem Device individuelle Queues angelegt und an die Middlware angebunden werden.

NodeJS-Server

Die Aufgabe des Servers ist es zum Einen, gestellte Verordungen auf Wechselwirkungen und Risiken zu prüfen bevor diese in der Datenbank abgelegt werden. Dabei werden die Daten aus dem Webservice verwendet, um sicherzustellen, dass keine Risiken oder Wechselwirkungen innerhalb der Medikation eines Patienten auftreten. Außerdem wird die gestellte Verordnung mit der Medikamentenanamnese des Patrienten verglichen. Der Server fungiert an dieser Stelle als überwachende Instanz. Wird also eine Verordnung vom Arzt über den Javascript-Client gestellt, wird zunächst eine Nachricht mit den Verordnungsdaten an den server versendet, der die Medikationsprüfung vornimmt und anschließend eine Antwort mit dem Status der Verordnung und zusätzlichen Informationen über Risiken und Wechselwirkungen sowie Vorschlägen zu alternativen Medikamenten an den Javascript-Client sendet.

Zum Anderen muss der Server alle Datenanfragen sowie Datenspeicherungen verwalten. Da die synchrone Kommunikation nicht über Http-Get und -Post stattfindet muss eine Anwendungslogik die Anfragen verarbeiten. Die Nachrichten der Middleware müssen daher eine eindeutige Struktur aufweisen und auf verschiedene Queues abgelegt werden. Des weiteren muss eine Rollenverteilung und die damit einhergehende Berechtigung auf Datenzugriffe verwaltet werden, um Verletzungen des Datenschutzes zu eleminieren.

Eine weitere Aufgabe des Servers ist es, Benachrichtigungen an die Android-Clients zu verschicken. Dies wird über das verschicken von Nachrichten an die Middleware realisiert. Zum Einen sollen Erinnerungen zu Verabreichungen versendet werden. Dafür muss die Systemzeit mit den Applikationszeiten der Verordnungen verglichen werden. Zum Anderen müssen Nachrichten über Änderungen an Verordnungen verschickt werden. Diese Benachrichtigungen werden verschickt, wenn ein Arzt über den Javascript-Client eine Verordnung bearbeitet hat.

Ressourcen

Um ein umfassendes Medikationsystem zu entwickeln, ist es notwendig sämtliche Infromationen zut Medikation eines Patienten als Ressource zu modellieren (siehe Artefakt WBA Modellierung) . Im Rahmen des Projektes soll der Fokus allerdings auf die Ressourcen des Patienten und der Verordnungen fallen und den dadurch aggregierten Medikationsplan. Die Ressourcen sollen als Grundlage für die Anwendungslogik und für die Präsentation der Informationen eines Patienten dienen.

Topics

Da eine nachrichtenorientierte Middleware im MDKS eingesetzt wird, müssen Queues verwendet werden, die bestimmte Nachrichten an die Consumer weiterleiten. Durch die Topics soll die synchrone und die asynchrone Kommunikation zwischen den Komponenten verwaltet werden. Die Nachrichten auf einen Exchange der RabbitMQ-Middleware werden durch die Angabe eines Topics auf die jeweiligen Queues verteilt.
Da ein Krankenhaus auf Stationen aufgeteilt ist, ist es sinnvoll ein Topic zu verwenden, dass sich auf eine bestimmte Station bezieht. Des weiteren müssen Topics für alle Patienten erstellt werden, damit jeder Patient nur seine persönlichen medikationsbezogenen Nachrichten erhalten kann. Da der Fokus in diesem Projekt auf den Medikationsplan fällt, wird zunächst davon abgesehen weitere Topics wie Verordnungen zu modellieren.
Damit die synchrone Kommunikation ebenfalls mit RabbitMQ umgesetzt werden kann, müssen Topics modelliert werden, die GET- und POST-Requests als Nachrichten verwalten können. Zu dem müssen individuelle Queues für die Clients angelegt werden, die die Daten nach einer Anfrage entgegen nehmen.

Datenrepräsentation

Interaktionen zwischen den Komponenten

todo begründung beschreibung sequenzdiagramm

Risiken

Die Risiken sollten bereits in der frühen Entwicklungphase des Projekts identifiziert und priorisiert werden, damit mögliche Probleme, die in der späteren Entwicklungsphase auftreten können vermieden werden können.
Nach der Betrachtung des Nutzungskontext konnten bereits einige Risiken bei der Behandlung des Datenschutzes bei der Entwicklung herauskristallisiert werden. Hinsichtloic

Asynchrone Kommunikation über das System nicht möglich

Die Anbindung des Androidclienten an den Publish-Subscribe Service mittels RabbitMQ könnte nicht gelingen, da das Projektteam mit dem Framework nicht vertraut ist und die benötigte Einarbeitungszeit nur sehr ungenau eingeschätzt werden kann. Sollte die asynchrone Kommunikation zwischen den Clients mittels RabbitMQ nicht möglich sein, so wäre eines der Alleinstellungsmerkmale des Systems gefährdet. Daher müsste ein alternativer Publish-Subscribe Service ermittelt werden, der die gleiche Funktionalität unterstützt.

Benachrichtigungen zu Ereignissen im System nicht möglich

Ein Notification-Service des Androidclienten ist für die Realisierung der in den Alleinstellungsmerkmalen genannten Errinnerungs- und Benachrichtigungsfunktion für Patienten und Pflegepersonal über Verabrechungen unerlässlich. Daher muss eine entsprechende Funktion recherchiert und in das System integriert werden. Sollte dies nicht gelingen, wäre die erfolgreiche Erstellung eines funktionalen Prototypen gefährdet.

Implementierung eines redundanten Serversystems

Zur Vermeidung von Serverausfällen und Sicherstellung der Erreichbarkeit des Systems soll eine redundante verteilte Systemarchitektur realisiert werden. Falls ein System ausfällt, werden die Aufgaben sofort von einem zweiten baugleichen System übernommen. Das Team ist mit der Implementation eines solchen Systems noch nicht vertraut. Zeitliche Engpässe können entstehen oder die Implementation gar scheitern. Es empfiehlt sich daher einen Prototypen in Form eines Proof of Concept zu erstellen und gegebenenfalls alternativen abzuwägen.

Datenaustausch zwischen Client und Server ist nicht möglich

Die Benutzer des Systems müssen über die Clients Daten mit dem Server austauschen können. Das Pflegepersonal muss in der Lage sein Daten über Verordnungen der Patienten vom Server zu erhalten, damit ein Medikationsplan erstellt werden kann. Ebenso müssen die Dokumentationen der Verabreichungen auf einer Datenbank persistent abgelegt werden können. Sollte dies nicht möglich sein, so sind die Alleinstellungsmerkmale und Kernfunktionen des Systems gefährdet.

MCI-Vorgehensmodell sieht Iterationen vor

Das Vorgehensmodell sieht für die geeignete Gestaltung eines gebrauchstauglichen interaktiven Systems nach DIN EN ISO 9241-210 Iterationen vor. Um zeitliche Engpässe zu minimieren werden diese Iterationen im Projektplan berücksichtigt.

Datenschutz

Der Umgang mit Patientendaten unterliegt strengen Datenschutzverordnungen. Diese sollten noch vor der Ermittlung der Benutzeranforderungen recherchiert werden um mögliche Widersprüche mit diesen frühzeitig zu erkennen.

Datensicherheit

Um das Ziel der Risikominimierung des Medikationsprozesses zu erfüllen, ist eine fehlerfreie Datenspeicherung unabdinglich. Sollten empfindliche Patienten- bzw. Medikationsdaten fehlerhaft angelegt/übertragen werden, so wäre das gesammte System nicht marktfähig. Daher empfiehlt sich die Nutzung eines Datenbankmanagementsystems nach ACID-Konsistenzmodell.

Umsetzung der Medikationsüberprüfung

Wenn ein Arzt Medikamente verordnet müssen die Wirkstoffe der Medikamente hinsichtlich Wechselwirkungen und Patienteneigenschaften wie Alter oder Nierenwerte überprüft werden. Der Vorgang der Überprüfung benötigt besondere Fachkenntnisse in dem Gebiet der Medizin sowie der Pharmazeutik. Die technische Umsetzung dieser Funktion muss deshalb mit fachkompetenten Menschen erarbeitet werden. Durch den eingeschränkten Zeitrahmen des Projekts ist dies leider nicht möglich. Eine mögliche Alternative ist die Anbindung des Systems an einen externen Webservice, der diese Aufgabe durchführt.

Ausfall eines Teammitglieds

Die Durchführung des Projekts ist auf den “work load” von zwei Teammitgliedern abgestimmt. Der Ausfall eines Mitglieds könnte das Projektziel gefährden.

Proof of Concepts