Dokumentation

Betreffende Software: Entscheidungsunterstützungssystem zur Diagnose von PIMS und Kawasaki

Eingereicht von: Jannis Gumz, Levin Kantelberg, Noah Kaspereit, Leon Röscher

Eingereicht am: 27.01.2022

Inhalt

Beschreibung der Software 1

Risikoanalyse 3

Qualitätsmanagement 6

Software-Lebenszyklus-Prozesse 8

Gebrauchstauglichkeit 12

Quellen 14

Beschreibung der Software

In Folge des Ausbruchs des Coronavirus SARS-Cov-2 und der damit einhergehenden COVID-19-Pandemie wurde bei Kindern im Alter von 0-14 Jahren gehäuft ein neuartiges Inflammationssyndrom mit dem Namen „Pediatric Inflammatory Multisystem Syndrome“ (PIMS) entdeckt. Die Symptome von PIMS ähneln dabei denen des bekannten Kawasaki-Syndroms (Kawasaki), welches vorrangig bei Kindern im Alter von 0-5 Jahren auftritt. Bei beiden Erkrankungen wird davon ausgegangen, dass eine vorherige Virusinfektion der Auslöser einer Immunreaktion ist, die zum jeweiligen Krankheitsbild führt. Während die Ursache von Kawasaki weiterhin unbekannt ist, gehen Wissenschaftler bei PIMS davon aus, dass eine vorausgegangene Infektion mit SARS-Cov-2 die Erkrankung auslöst.
Entwickelt wurde ein Entscheidungsunterstützungssystem, welches im medizinischen Kontext bei der Diagnose von PIMS und Kawasaki unterstützen soll. Die Software soll Ärzte und medizinische Fachangestellte bei Diagnosen unterstützen. Anhand von bereitgestellten Patientendaten wird eine Klassifikation der Patienten durchgeführt. Die Benutzeroberfläche ermöglicht es dem Anwender, auf einen Blick zu sehen, bei welchen Patienten potenziell eine Infektion mit PIMS oder Kawasaki vorliegt. Zudem werden Patienten, bei denen die vorhandenen Daten unzureichend sind, aber Anzeichen auf eine Erkrankung vorliegen, besonders gekennzeichnet. Wichtig zu beachten hierbei ist, dass der Anwender sich nie vollständig auf die Software verlassen sollte. Sie dient lediglich der Entscheidungsunterstützung und eine getroffene Klassifikation sollte immer hinterfragt und analysiert werden. Um dies zu erleichtern, bietet die Software neben der Klassifikation auch eine genauere Analyse des Patienten an. Hierbei kann der Anwender sich einen Patienten im Detail anschauen und sehen welche Aspekte aus den bereitgestellten Patientendaten die Software zu der getroffenen Klassifikation geführt haben.
Datensätze für mehreren Patienten können ebenso verarbeitet werden wie auch einzelne Patientendaten. Um größtmögliche Datenintegrität zu gewährleisten, wird die Datenbank von Bestandsdaten gesäubert, sollte der Nutzer neue Datensätze hinzufügen. Eine persistente Speicherung der Daten ermöglicht es, diese auch zu einem späteren Zeitpunkt erneut abzufragen. Eine Installationsanleitung liegt der Anwendung bei.

Die Software gliedert sich im Wesentlichen in die folgenden drei Hauptkomponenten:

ETL Prozess:
Um die bereitgestellten Patientendaten in ein einheitliches Format zu überführen, wurde ein Exchange, Transform, Load (ETL) Prozess entwickelt. Dieser wandelt die in Form von CSV-Dateien bereitgestellten Patientendaten in das OMOP Common Data Model (OMOP) um und speichert sie in eine Datenbank mit dem entsprechenden OMOP Schema. Vorteil der Transformation in das OMOP-Modell ist die leichte Wiederverwendbarkeit der Daten und des zugehörigen Analyseprozesses. Dadurch dass man ein standardisiertes Datenmodell verwendet, können die bereitgestellten Daten ohne erneuten Transformationsaufwand auch von anderen Programmen

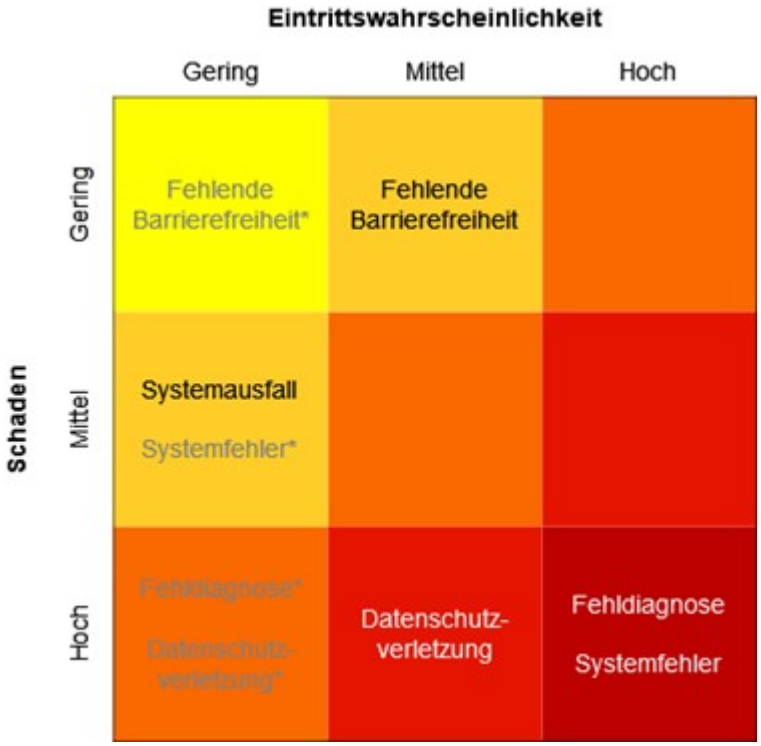
analysiert und ausgewertet werden. Andersrum kann auch die entwickelte Software fremde Datensätze analysieren, solange diese im selben Datenmodell vorliegen.

Backend:
Im Backend der Hauptanwendung findet die tatsächliche Analyse und Klassifikation der Patienten statt. Das Backend greift dabei auf die transformierten Daten aus der Datenbank zu und überprüft die Patienten auf vorher definierte Symptome und Parameter. Die Daten werden anhand dieser Kriterien durch einen Entscheidungsbaum klassifiziert. Die Klassifikation für PIMS erfolgt dabei nach der WHO-Leitlinie [4]. Im Fall von Kawasaki wird sich hingegen hauptsächlich an den Kriterien der AWMF orientiert [3]. Ergänzt werden diese um Klassifikationen für ein unvollständiges Krankheitsbild, welches seinen Ursprung in unvollständigen Daten haben kann.

Frontend:
Das Frontend besteht aus einer Webanwendung, welche der Anwender in seinem Browser öffnen kann. Mittels eines persönlichen Logins gelangt man auf die Startseite, von der aus man die verschiedenen Funktionen der Software ansteuern kann. Der Anwender hat die Möglichkeit, neue Patientendaten hochzuladen und somit den vorher erwähnten ETL Prozess zu starten. Hierfür kann er zusätzlich eine Konfigurationsdatei angeben, welche die benötigten Daten zur Verbindung mit der Datenbank enthält. Des Weiteren kann man einen einzelnen Patienten anlegen und entsprechende Symptome und Parameter eintragen. Lädt der Anwender einen neuen Satz an Patientendaten hoch, so wird die Datenbank zurückgesetzt, um die Datenintegrität zu wahren. Dies reduziert Fehlklassifikationen, welche durch missgedeutete Korrelationen zwischen den Datensätzen entstehen könnten. Auf der linken Seite befindet sich ein Reiter, der sich automatisch ausklappt und als Menü dient. Dieser beinhaltet alle Funktionalitäten der Software und kann von jeder Seite aus bedient werden. Über den Button „Analyse“ gelangt man auf die Übersichtsseite aller klassifizierten Patienten. Der Anwender kann diese je nach Bedarf sortieren und einzelne Patienten zur genaueren Analyse auswählen. Hierbei werden ihm die Aspekte der Patientendaten angezeigt, welche zur jeweiligen Klassifizierung geführt haben sowie solche, die zu einer höheren Klassifizierung gefehlt haben. Außerdem können auch im Nachhinein noch Symptome oder Parameter von einzelnen Patienten bearbeitet werden, um die Aussagekraft der Ergebnisse zu korrigieren. Über das Menü sind darüber hinaus noch die Einstellungen mittels des Zahnrades zu erreichen. Hier kann der Nutzer die aktuelle Konfigurationsdatei herunterladen, die Analyse der Patientendaten neu starten, die Datenbank zurücksetzen und die für die Farbcodierung der Analyseergebnisse verwendete Farbe einstellen.

Risikoanalyse

Da das System als Medizinprodukt Einsatz finden soll, ist es zwingend erforderlich, das Risikomanagement konform zur ISO 14971 umzusetzen, um so Schäden zu vermeiden und vorzubeugen [2].
Aus der Zweck- bzw. Anforderungsbestimmung des Systems, gegeben durch die Beschreibung der Software, wurden fünf relevante Gefährdungsquellen identifiziert, die im Folgenden einzeln beschrieben werden. Dabei werden Ursachen und Folgen erläutert und abgeschätzt, wie wahrscheinlich ein Risiko eintritt und wie Schwer dabei die Folgen wiegen. Die Einschätzung bezieht sich auf das Risiko **vor** der Umsetzung von Gegenmaßnahmen und wird in einer Risikomatrix in der nebenstehenden Abbildung dargestellt.
Im darauffolgenden Abschnitt werden Gegenmaßnahmen beschrieben, die in den Entwicklungsprozess einbezogen werden, um den festgelegten Akzeptanzkriterien zu entsprechen. Dabei wird angestrebt, die zuvor bestimmten Risiken auf Restrisiken zu reduzieren. Diese Ziele werden in der Risikomatrix in grauer Schrift und mit * dargestellt.



Fehldiagnosen:
Das System soll medizinisch geschultes Personal bei der Diagnose von Krankheiten unterstützen. Fehler im Algorithmus und der Visualisierung, die ein größeres Vertrauen in die Ergebnisse hervorrufen, als zu begründen wäre, können zu Fehldiagnosen führen. So ist einerseits eine falsch positive Diagnose für den Patienten schädlich, da dies zu unnötigen und potenziell schädlichen Behandlungen führen kann, andererseits ist auch eine unterlassene Diagnose schädlich, da hier die Krankheit unbehandelt weiter voranschreiten kann.
Im schlimmsten Fall unterstützt das System ausschließlich falsche Diagnosen und ist damit an sich schadhaft. Der hervorgerufene Schaden ist somit als *hoch* einzuschätzen. Da Entwickler vor der Implementierung des Systems über kein Wissen zu Kawasaki bzw. PIMS verfügen müssen, ist die Eintrittswahrscheinlichkeit dieses Risikos a priori *hoch*.

Zugriff auf Patientendaten durch unberechtigte Dritte (Datenschutzverletzung):
Da es sich um ein System handelt, dass sowohl persönliche Daten (Name, Wohnort, Geburtstag), als auch medizinisch relevante Daten (z.B. Krankheiten, Untersuchungen, Versicherungsdaten) verwaltet, könnten diese persönlichen Daten durch unberechtigte Dritte entwendet, bzw. veröffentlicht werden. Der Missbrauch dieser Daten könnte private und berufliche Folgen für die betroffenen Patienten haben und das öffentliche Vertrauen in die Gesundheitsversorgung gefährden. Weiterhin ist mit rechtlichen Konsequenzen (beispielsweise einer Klage gegen die Gesundheitseinrichtung) zu rechnen. Der potentielle Schaden ist sowohl für Patienten als auch für Anwender hoch. Der Zugriff auf Patientendaten im System kann ausschließlich durch medizinisch geschultes Personal oder Dritte, die in der Informatik geschult sind, erfolgen. Es ist also von einer mittleren Eintrittswahrscheinlichkeit auszugehen.

Systemausfall:
Durch unvorhergesehene Ereignisse wie Stromausfällen, Naturkatastrophen oder mutwilligen Schädigungen kann es zu Ausfällen des Systems kommen. Dadurch können Entscheidungen des medizinischen Personals nicht mehr unterstützt werden. Es ist also damit zu rechnen, dass das Stellen einer Diagnose mehr Zeit in Anspruch nimmt und bereits gestellte Diagnosen nicht mehr abrufbar sind. In akuten Fällen ist die Diagnose PIMS bzw. Kawasaki kontraindizierend für eine Notfallbehandlung. Der Ausfall des Systems kann zu einer verspäteten und im schlimmsten Fall falschen Behandlung eines Patienten führen. Da Diagnosen von medizinisch geschultem Personal gestellt werden müssen, ist der Schaden nur als *mittel* anzusehen. Da die Wahrscheinlichkeit eines unvorhergesehenen Ereignisses gering ist, ist auch die Wahrscheinlichkeit eines Systemausfalls *gering*, muss jedoch im Kontext von Systemfehlern beurteilt werden.

Systemfehler:
Im Gegensatz zu Systemausfällen bleibt das System bei einem Systemfehler komplett oder in Teilen nutzbar. Ein solcher Fehler ist von Anwendern nicht immer erkennbar. Diagnosen, die unter Einwirkung eines Fehlers gestellt wurden, können unvollständig oder falsch sein und stellen somit Fehldiagnosen dar. Fehler können weiterhin dazu führen, dass gestellte Diagnosen nicht im Datenbanksystem persistiert werden. Da das System den Anschein erweckt, korrekt zu funktionieren, werden solche Fehler erst spät erkennbar. Systemfehler können in bestimmten Fällen auch zu kompletten Systemausfällen führen, wodurch die Schwere des Schadens als *hoch* anzusehen ist. Wurde die Software nicht rigoros getestet, so ist auch die Wahrscheinlichkeit von Systemfehlern *hoch*.

Fehlende Barrierefreiheit:
Bei der Entwicklung der Benutzerschnittstelle des Systems können Annahmen getroffen werden, die Nutzergruppen von der Nutzung des Systems ausschließen. Eine unrealistische aber häufig getroffene Annahme ist zum Beispiel, dass Farben von allen Nutzern gleich wahrgenommen werden. Wenn nun die Darstellung einer Diagnose allein auf Farben und speziell Farbtonunterschieden beruht, ist sie nicht für alle Nutzer zugänglich. Behandlungen werden dadurch behindert. Außerdem können Diagnosen unterschiedlich interpretiert werden, wodurch sie uneindeutig werden und zu falschen Behandlungen führen. Die Wahrscheinlichkeit Nutzergruppen

auszuschließen ist a priori hoch, da es sich jedoch um kleine Gruppen handelt und die visuelle Schnittstelle dem medizinischen Personal bekannt ist, ist von einer insgesamt *mittleren* Eintrittswahrscheinlichkeit für einen Schaden auszugehen. Es ist auch davon auszugehen, dass diagnoserelevante Daten eindeutig für einen großen Teil der Anwender dargestellt werden. Der Schaden ist daher als *gering* anzusehen.

Gegenmaßnahmen:
Um die Eintrittswahrscheinlichkeiten und Schäden der zuvor beschriebenen Risiken zu verringern, wurden Gegenmaßnahmen ermittelt, welche im Entwicklungsprozess berücksichtigt werden sollen.

Der genaue Zweck des Systems muss allen Anwendern vor dessen Benutzung klar sein. Außerdem müssen Anwender, bevor die Nutzung erlaubt wird, eine Schulung im Umgang mit dem System erhalten. Dadurch wird sichergestellt, dass alle Nutzer etwaige Barrieren kennen und mit diesen umgehen können. Der Schaden durch fehlende Barrierefreiheit wird damit verringert.

Durch Feedback in der Entwicklung kann Barrierefreiheit kontinuierlich sichergestellt werden, wodurch die Eintrittswahrscheinlichkeit minimiert wird (User Centered Design).

Nutzern muss klar sein, dass das System ausschließlich zur Unterstützung von Entscheidungen konzipiert ist. Das setzt eine transparente Entscheidungsunterstützung voraus. Das Risiko von Fehldiagnosen wird dadurch idealerweise unabhängig vom System. Der potentielle Schaden bleibt derselbe, aber die Eintrittswahrscheinlichkeit kann -relativ zum System- minimiert werden.

Um Datenschutzverletzungen zu vermeiden, dürfen persönliche Daten ausschließlich für die betreffende Person selbst und für medizinisches Personal, das die Behandlung der Person sicherstellt, zugänglich sein. In keinem Fall dürfen die Daten für Dritte lesbar bzw. schreibbar sein. Eine Verschlüsselung der Daten, passwortgeschützter Zugang, Definition von Lese/Schreibrechten und automatischer Logout minimieren das Risiko für Datenschutzverletzungen. Davon wurde bis jetzt der automatische Logout und der passwortgeschützte Zugang umgesetzt. Die Anonymisierung der Daten würde Schäden von Datenschutzverletzungen minimieren, könnte jedoch zu Verwechslungen führen.

Das Auftreten von Systemfehlern kann durch rigoroses Testen des Systems nahezu ausgeschlossen werden. Im Weiteren wird durch Kommunikation von Systemfehlern der Nutzer beständig über mögliche Angriffsvektoren informiert. Dadurch werden sowohl Schaden als auch Eintrittswahrscheinlichkeit eines Systemfehlers verringert.

Insgesamt ist zu erkennen, dass Eintrittswahrscheinlichkeiten im Allgemeinen auf ein akzeptables Maß verringert werden können. Schäden betreffen jedoch im schlimmsten Fall die Gesundheit von Menschen und dürfen daher bei der Entwicklung nie unterschätzt werden.

Qualitätsmanagement

Dokumentenhistorie

Version	Datum	Änderungen	Autoren
0.1	2021-11-01	Erstellung der Vorlage	Sven Helfer
0.2	2021-11-25	Ziele & Geltungsbereich festhalten	Noah Kaspereit
0.3	2021-12-02	Rollen und Zuständigkeiten definiert	Noah Kaspereit
1.0	2021-12-10	Beschreibung der Durchführung und Dokumentation	Noah Kaspereit

Gegenstand dieser SOP

Diese SOP beschreibt das Vorgehen bei der Entwicklung von Software im Rahmen des Komplexpraktikums „Medizinische Informatik 1“.
Ziel ist es eine geordnete, agile Entwicklung zu gewährleisten. Diese beinhaltet unter anderem den richtigen Umgang mit auftretenden Problemen während der Entwicklung. Fehler und Probleme sollen schnell erkannt und klassifiziert werden. Durch einen agilen Entwicklungsprozess wollen wir flexibel auf diese Probleme und sich ändernde Anforderungen eingehen. Zudem werden mittels dieser SOP klare Rollen und Zuständigkeiten innerhalb des Teams verteilt

Geltungsbereich

Diese SOP gilt während des gesamten Entwicklungsprozesses der Software im Rahmen des Komplexpraktikums „Medizinische Informatik 1“ und für alle daran beteiligten Entwickler. Verantwortlich für die Einhaltung und Aktualisierung der SOP ist der Qualitätsmanager des Projekts.

Rollen und Zuständigkeiten

Frontend Entwickler:
Verantwortlich für die Entwicklung des User Interfaces sowie der Schnittstelle zwischen dem Front- und Backend. Probleme die in diesen Bereichen auftreten sind vom Frontend Entwickler nach dessen Schwere und Relevanz zu klassifizieren und im Laufe der agilen Entwicklung zu bearbeiten.

Backend Entwickler:
Verantwortlich für die Entwicklung eines ETL-Prozesses, einer geeigneten Analyse der Daten sowie der Schnittstelle zwischen Front- und Backend. Probleme die in diesen Bereichen auftreten sind vom Backend Entwickler nach dessen Schwere und Relevanz zu klassifizieren und im Laufe der agilen Entwicklung zu bearbeiten.

Qualitätsmanager:
Verantwortlich für die Erstellung und Einhaltung einer SOP. Der Qualitätsmanager ist weiterhin hauptverantwortlich für das Führen einer Dokumentation.

Durchführung

Zur initialen Strukturierung der Arbeit werden ein gemeinsames Git-Repository und ein Kanban Board angelegt. Alle Entwickler erarbeiten zusammen die Meilensteine des Projekts und tragen diese in das Kanban Board ein.
Entsprechend eines agilen Entwicklungsprozesses findet jede Woche ein Treffen mit allen

Entwicklern statt. Hierbei werden die einzelnen Arbeiten der letzten Woche vorgestellt und Probleme erläutert. Die vom Front- oder Backend Entwickler vorklassifizierten Probleme werden im Team besprochen und final nach Schwere und Relevanz eingeordnet. Anschließend werden entsprechende Tickets in das Kanban Board eingetragen und einem entsprechenden Entwickler zugeordnet. Zusätzlich können Termine mit dem Betreuerteam des Komplexpraktikums stattfinden, bei denen sich die Anforderungen verfeinern oder Fragen und Probleme geklärt werden. Sich aus diesen Treffen ändernde Anforderungen werden dabei ebenso als Ticket festgehalten. Das interne Ticketsystem befolgt dabei folgende Struktur. Neu erstellte Tickets landen im „Backlog“, einer Sammlung aller noch zu erledigenden Tickets. Aus diesem werden bei den wöchentlichen Treffen der Entwickler Tickets entnommen, welche für die kommende Woche eingeplant werden. Diese Tickets werden entsprechend in die Kategorie „in progress“ verschoben. Sobald ein Entwickler sein Ticket seiner Meinung nach bestmöglich bearbeitet hat, verschiebt er es nach „Ready for review“. In dieser Kategorie warten die Tickets darauf, von einem anderen Entwickler auf mögliche Fehler überprüft zu werden. Der Reviewer kann nun entweder dem Ticket eine Review-Nachricht anhängen und es dem Entwickler zurück in die Kategorie „in progress“ geben oder er akzeptiert das Ticket und legt es im Anschluss in die Kategorie „done“. Mehrfach im Entwicklungsprozess werden dem Betreuerteam die aktuellen Zwischenstände der Software präsentiert, um den Fortschritt zu validieren und frühzeitig Anpassungen vornehmen zu können.

Dokumentation

Die Dokumentation wird hauptsächlich vom Qualitätsmanager geführt. Einzelne Bereiche wie die Software-Lebenszyklus-Prozesse werden auch von den Entwicklern mit dokumentiert, welche durch die Nähe zum Quellcode ein tieferes Verständnis in diesen Bereichen haben. Die Dokumentation wird parallel zur Entwicklung geführt und regelmäßig entsprechend den Änderungen an der Software angepasst. Während der Entwicklung ist der Qualitätsmanager angehalten, die Qualität und den Zwischenstand der Dokumentation regelmäßig zu kontrollieren und gegebenenfalls Anpassungen vorzunehmen. Am Ende der Entwicklung wird die Dokumentation von allen Personen Korrektur gelesen, um die Vollständigkeit und Korrektheit zu gewährleisten.

Freigabe

Version 1.0	Geprüft am	Freigegeben am
vom 10.12.2021	Von (Unterschrift)	Von (Unterschrift)

Software-Lebenszyklus-Prozesse

Damit Qualität und Verfügbarkeit des Systems kontinuierlich sichergestellt werden können, ist es nötig, den gesamten Lebenszyklus der Software zu betrachten, siehe [1]. Dazu orientiert sich der Entwicklungsprozess an der DIN EN 62304 und setzt das V-Modell um [5].
In den vorigen Abschnitten wurde die Softwarearchitektur beschrieben und analysiert, welche funktionalen Anforderungen vom System erfüllt werden müssen, damit es als zweckbestimmtes und sicheres Medizinprodukt eingesetzt werden kann.
In diesem Abschnitt wird nun zunächst betrachtet, wie diese Anforderungen im Entwicklungsprozess sichergestellt wurden. Der Aspekt Sicherheit durchdringt alle Bereiche eines Medizinproduktes. Daher wird im weiteren Abschnitt beschrieben, welche Tests verwendet werden, um Fehler innerhalb der Anwendung zu minimieren. Der letzte Abschnitt gibt einen Ausblick darauf, wie zukünftig die kontinuierliche Verfügbarkeit des Systems sichergestellt werden kann und welche Grundlagen der Systemarchitektur dafür eingebunden wurden. Das V-Modell wird also ausgehend vom zentralen Punkt des Software-Entwurfs beschrieben.

Analyse und Entwicklung
Die Aspekte Zweckbestimmtheit und Sicherheit werden als kritische Dimensionen der Qualität eines Medizinproduktes betrachtet und sind zentral für die MDR. Daher wurden sie als kritische Anforderungen bei der Analyse und Entwicklung umgesetzt. Zweckbestimmtheit bedeutet, dass die Software ausschließlich den Zweck der Entscheidungsunterstützung für PIMS und Kawasaki umsetzt. Ein konkrete Gegenbeispiel wäre ein System, das Entscheidungen zu anderen seltenen Krankheiten unterstützt. Die Software erfüllt den Aspekt der Sicherheit, wenn sie keine Risiken birgt, die größer sind, als in den Akzeptanzkriterien festgelegt, vgl. Risikoanalyse.
Die Zweckbestimmung wurde kontinuierlich sichergestellt, indem bei der Entwicklung Datensätze zum Kawasaki Syndrom und PIMS einbezogen wurden. Dabei wurden auch Ergebnisse aus Konsultationen mit Experten berücksichtigt. Es wurde klar, dass es besonders wichtig ist, die Zweifel der Experten zu fokussieren und nicht ausschließlich die Perspektive des Entwicklungsteams zu bestärken. Letzteres führt zu Voreingenommenheit oder Bias im System, siehe [6, 7]. Dieser Bias kann zu Fehlentscheidungen führen, wodurch schlussendlich wieder die Sicherheit des Systems gefährdet wird.
Eine grundlegende Quelle dieses Bias ist gegeben durch Abstraktionen, die zum Zweck der Usability getroffen werden [6]. Ein konkretes Beispiel ist folgender Anwendungsfall:

Eine Spezialistin für seltene Krankheiten möchte für alle Patienten einer Einrichtung prüfen, ob diese am Kawasaki- oder PIMS erkrankt sind. Der manuelle Aufwand wäre viel zu hoch, weswegen Sie das hier beschriebene System nutzt.
Damit das System für diesen Anwendungsfall nutzbar ist, muss es eine Seite geben, auf der alle Patienten und zugehörige Daten dargestellt werden. Das allein reduziert den Arbeitsaufwand jedoch nicht Maßgeblich, denn alle Einträge müssten weiterhin manuell überprüft werden. Das System muss also zwangsläufig Kohorten für Kawasaki und PIMS filtern können, welche auf rigorosen Leitlinien wie [3] und [4] basieren. Diese Leitlinien sind Grundlage der Zweckbestimmung.

Durch das Filtern nach Kohorten wird allerdings - entgegen der Zweckbestimmung - eine Entscheidung impliziert, was jedoch der Spezialistin obliegen sollte. Liegen zu Patienten unvollständige Daten vor, so wird darüber hinaus eine falsche Entscheidung getroffen, was die Sicherheit des Systems gefährdet.
Aus diesem Grund wurde bei der Implementierung zunächst entschieden, statt rigorosen Charakteristiken, Ordnungen zu verwenden. Die Idee dahinter ist zusammengefasst, dass die Spezialistin eine Sortierung sehen kann, bei der Patienten mit vielen spezifischen Symptomen früher sichtbar sind, da sie relevanter für die Entscheidung der Spezialisten sind. Es wurde angestrebt, eine Wahrscheinlichkeit für beide Syndrome zu berechnen. Für diesen Zugewinn an Usability ging jedoch die Zweckbestimmung verloren. Statt rigoroser Leitlinien wurde eine intransparente Gewichtung für jedes Symptom verwendet.
Schlussendlich wurde ein Kompromiss aus Filter und Ordnung verwendet. Kohorten, die den Leitlinien entsprechen, bekommen die wichtigste Ordnungszahl. Geringere Ordnungen haben Leitlinien, wie etwa in [3] - für das inkomplette Kawasaki Syndrom. Darauf folgend werden dann Patienten aufgeführt, deren Symptome auf eine Erkrankung hindeuten und bei denen vermutet werden kann, dass Daten unvollständig sind. Diese letzten Annahmen wurden auf Basis der Usability getroffen. Die Spezialistin kann nun entscheiden, ob beispielsweise weitere Untersuchungen durchgeführt werden müssen. Die Kohorte mit der unwichtigsten Ordnung umfasst Patienten, die kein charakteristisches Symptom aufweisen.
Diese Definitionen können in einer Legende nachvollzogen werden. Insgesamt wurden also Zweckbestimmung und Sicherheit im Kontext der Usability umgesetzt. Die Software hat darüber hinaus einen Nutzen für die Spezialisten und unterstützt ihre Entscheidungen.
Im Sinne der Usability wurden also beim Softwareentwicklungsprozess Kompromisse getroffen, die auf Expertengesprächen basieren, wodurch die Sicherheit und Zweckbestimmtheit des Systems sichergestellt wurden. Diese Expertengespräche stellen einen fundamentalen Input für den Softwareentwicklungs- und Analyseprozess dar.

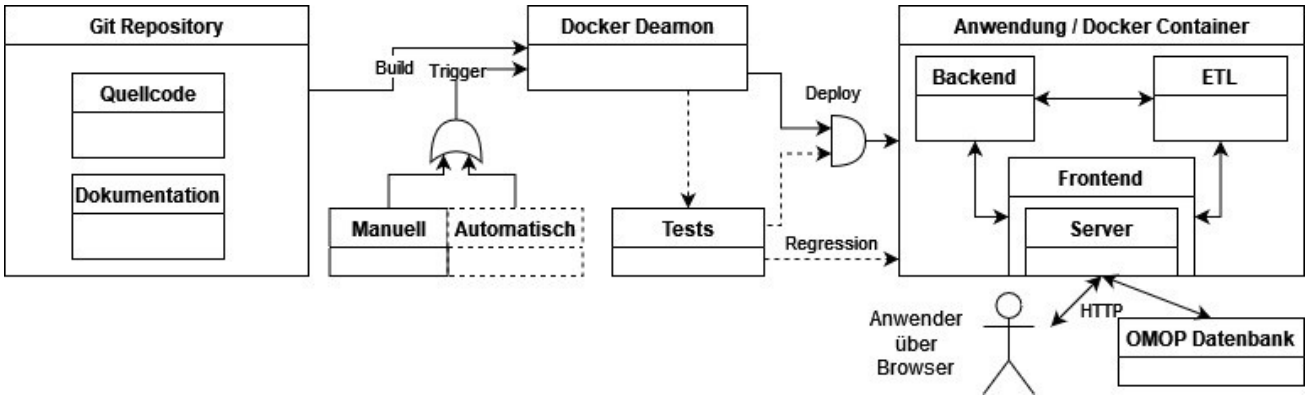
Verifizierung
In der Risikoanalyse wurde beschrieben, dass Systemfehler im schlimmsten Fall zu Fehldiagnosen führen können. Der hervorgerufene Schaden wäre daher hoch und gefährdet die Sicherheit des Systems.
Im vorigen Abschnitt wurde dargelegt, wie sichergestellt wurde, dass der Bias im System möglichst gering ist. Die Entscheidung obliegt schließlich den Anwendern, das System unterstützt und ist transparent. Es ist also an sich nicht schadhaft und kann keinen schweren Schaden hervorrufen.

Wenn die Eintrittswahrscheinlichkeit von Systemfehlern auf ein geringes Maß reduziert werden kann, so ist es teil der MDR Klasse IIa.
Aus diesem Grund wurden Systemtests definiert, die das Ziel haben, Systemausfälle auszuschließen und weiterhin die valide Ein- und Ausgabe von Daten zu prüfen. So wurden beispielsweise für das Backend Modul Unit- und Integrationstests implementiert.

Besonders relevant ist der Integrationstest. Dieser deckt den kompletten Prozess der Entscheidungsunterstützung ab. Es wird dabei ein Patient angelegt, überprüft ob die Daten korrekt in der Datenbank persistiert werden und schließlich eine korrekte Ordnungszahl bekommt. Dadurch werden die kritischen Anforderungen an das System verifiziert.
Folgender Test muss noch vor der Freigabe der Software implementiert werden. Da die Interaktion der Anwender mit dem System über ein User Interface geschieht, muss auch dieses getestet werden. Konkret muss der komplette Weg vom ersten Öffnen der Anwendung, über Login und das Anlegen eines Patienten, bis zur Entscheidungsunterstützung getestet werden. Dabei soll auch geprüft werden, ob die Legende korrekt dargestellt wird. Wird dieser Integrationstest regressiv ausgeführt, so ist auch die Verfügbarkeit der Anwendung sichergestellt. Das Selenium Framework ist Basis dafür.

Die beschriebenen Systemtests verifizieren die kritische Anforderung der Sicherheit. Die Anwendung wird dabei als Gesamtes getestet und lässt sich somit - als Standalone Software - der Sicherheitsklasse A zuordnen. Die Hardware, auf der das System läuft, ist nicht Teil des Systems selbst. Wird Hardware benutzt, so müssen sich Nutzer klar sein über damit zusammenhängende Risiken und die Dokumentation der Hardware konsultieren.

System Architektur, Wartung und Kontinuität
Damit die Software freigegeben und veröffentlicht werden kann, wurde das zuvor beschriebene System in eine Architektur eingebettet. Diese Systemarchitektur ist in der folgenden Abbildung dargestellt. Der Grund für eine solche Architektur ist, dass das System zu jeder Zeit anpassbar sein muss. Das ist z.B. der Fall, wenn sich Leitlinien zu PIMS oder Kawasaki ändern. In diesem Fall müsste das Backend Modul angepasst werden. Auch Änderungen am Frontend könnten erforderlich werden, wenn etwa Umfragen zur Usability ergeben, dass das Design angepasst werden muss.



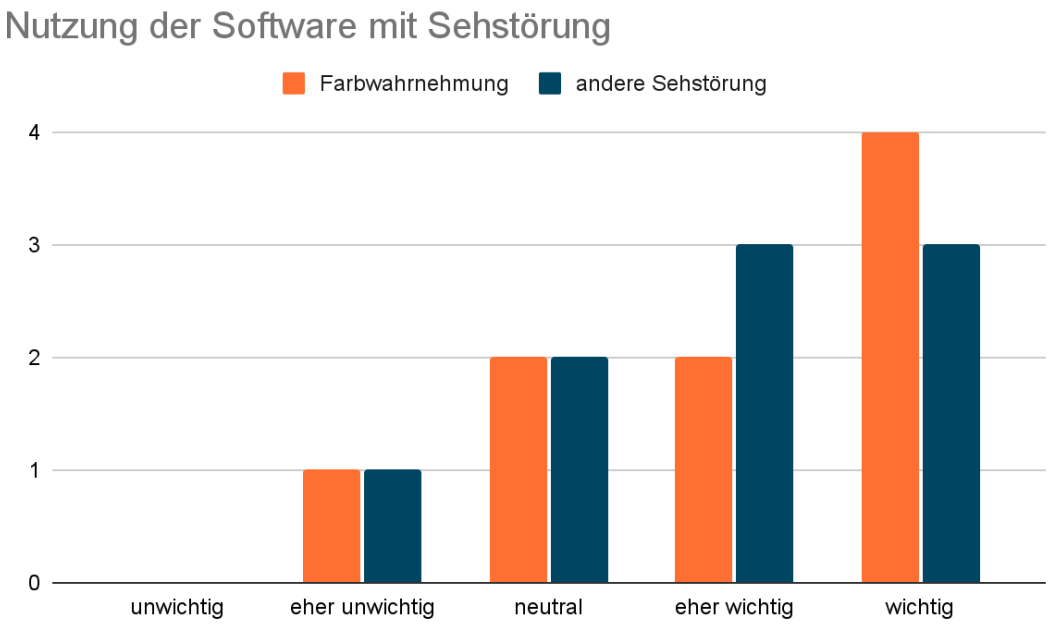
Dazu liegt der Quellcode der Implementierung in einem Git Repository. Änderungsanfragen werden vom Entwicklerteam aufgenommen und, wie in der SOP beschrieben, vom Entwicklerteam als

Problem behandelt. In regelmäßigen Abständen und nach der Lösung von Problemen mit hoher Priorität werden geprüfte Versionen des Quellcodes auf dem Main Branch veröffentlicht.
Damit diese Änderungen für Nutzer zugänglich werden, wird ein dazugehöriges Docker Image vom Docker Deamon gebaut. Dieses Image wird durch die zuvor beschriebenen Tests verifiziert und daraufhin freigegeben.
Dieser Prozess stellt sicher, dass Anwender ausschließlich verifizierte Images benutzen können. Manipulationen durch Entwickler oder Dritte können jedoch so nicht ausgeschlossen werden. Auch stellt dieser Prozess nicht sicher, dass alle Nutzer immer im Besitz der aktuellsten Version der Anwendung sind und so die Analysen auf dem neuesten Stand der medizinischen Forschung durchgeführt werden können.
Daher wird für die Zukunft angestrebt, den Deployment-Prozess vollständig zu automatisieren und Images an einem zentralen Punkt, in Form von Releases, freizugeben. Dazu wird es erforderlich, einen Build-Server einzurichten, der auf dem Main Branch des Git Repositories lauscht. Nach Definition eines Releases wird ein automatischer Build angestoßen, der die Unit- und Integrationstests durchläuft und beim Bestehen validiert. Jedes valide Image wird mit einem Zertifikat versehen. Das Leben der Software endet, wenn das Zertifikat ungültig ist. Die Invalidierung geschieht automatisch mit dem Fehlschlag der Regressionstest und kann auch manuell erfolgen. Eine zentrale Validierungsstelle würde die Sicherheit erhöhen, benötigt jedoch einen gesonderten Zugang.

Gebrauchstauglichkeit

Um die Nutzer gemäß des User-Centered Designs schon früh in den Entwicklungsprozess einzubeziehen, wurde in den ersten Phasen der Entwicklung ein Fragebogen entworfen. Ziel des Fragebogens war es, die Anforderungen der Nutzer an das Entscheidungsunterstützungssystem herauszuarbeiten und gleichzeitig ein erstes Feedback für bis dahin erarbeitete Designvorschläge zu erhalten. Der Fragebogen bestand aus insgesamt 32 Fragen, verteilt auf die vier Kategorien „Zu Ihrer Person“, „Ihre Aufgaben und Ziele“, „Darstellung“ und „Arbeitsweise und Performance“. Im Folgenden werden die wichtigsten Erkenntnisse der Auswertung vorgestellt.

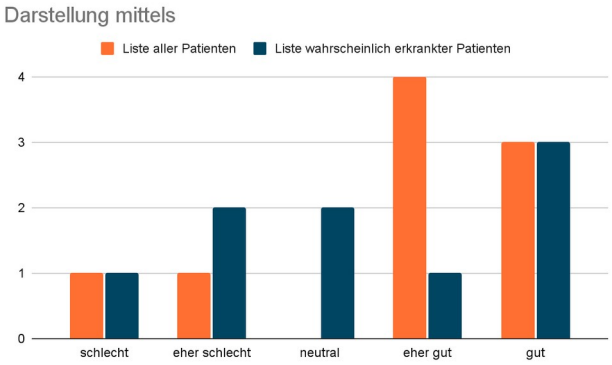
Barrierefreiheit:
Um zu ermitteln, welche Punkte der Barrierefreiheit bei den Nutzern die höchste Priorisierung haben, wurden einige Fragen zu diesem Thema gestellt. Es stellte sich heraus, dass zwar keiner der Teilnehmenden an einer Sehstörungen leidet, das Thema aber nichtsdestotrotz vom Großteil als wichtig erachtet wird, wie der folgenden Abbildung zu entnehmen ist.

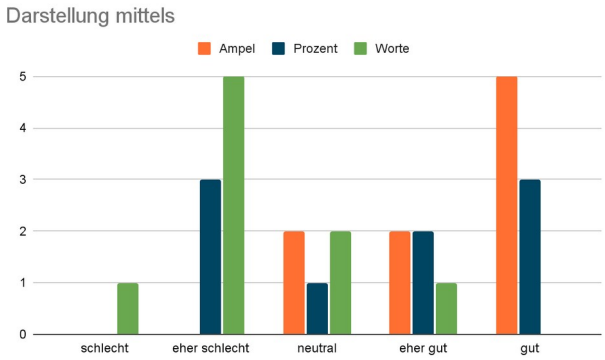


Als Konsequenz aus den Ergebnissen wurde ein besonderer Fokus auf die optische Gestaltung der Anwendung gelegt. Wichtig hierbei war, dass Informationen nicht ausschließlich farblich kodiert werden, dass ein Augenmerk auf die ausgewählten Farben gelegt wird (Rot-Grün Sehschwäche) und dass zwischen verschiedenen Farben immer ein ausreichend hoher Kontrast besteht.

Darstellung der Ergebnisse:
Ein wichtiger Aspekt der Anwendung ist die Darstellung der Ergebnisse der Entscheidungsunterstützung. Hierfür gab es verschiedene Ansätze, um sowohl eine Übersicht über mehrere Patienten zu ermöglich als auch eine genauere Analyse eines einzelnen Patienten zu

bieten. Die Ergebnisse des Fragebogens zu diesem Thema zeigen die beiden folgenden Abbildungen.





Es zeigte sich, dass die Teilnehmenden sich mehrheitlich für die Darstellung einer Liste von allen Patienten aussprachen und eine einzelne Analyse sich sowohl als Farbcodierung ähnlich zu den bekannten Ampelfarben oder durch eine Prozentangabe vorstellen könnten. Auch eine Kombination aus einer Farbcodierung mit einer zusätzlichen Prozentangabe wurde vorgeschlagen. Schlussendlich wurde sich nichtsdestotrotz gegen eine Angabe in Prozent entschieden. Hauptgrund dafür war, dass es keine klare Zuordnung zwischen Symptomen und Prozentwerten gibt. Dadurch können die ermittelten Prozentwerte zu falschen Annahmen führen, welche Behandlungsfehler nach sich ziehen könnten. Stattdessen gibt es eine Klassifizierung in klar vorgeschriebene Kategorien in Kombination mit einer Farbcodierung. Zusätzlich kam durch die Teilnehmer die Idee auf, die Liste der Patienten nach der Wahrscheinlichkeit einer Erkrankung für PIMS oder Kawasaki zu sortieren, wodurch man auf einen Blick die gefährdeten Patienten erkennen kann.

Designvorschlag:
Am Ende des Fragebogens wurde den Teilnehmern ein erstes Mock-Up des User Interfaces präsentiert. Die Haupterkenntnisse dieses Bereichs waren, dass eine klare Definition der Farbcodierung nötig ist. Zudem muss die verwendete Sprache der Anwendung dem Sprachgebrauch der späteren Anwendergruppe entsprechen. Ein Beispiel hierfür wäre der Begriff „Batchvorgang“, welcher durch passendere Terminologie ersetzt werden muss. Vom Designvorschlag bestätigt wurden hingegen die explizite Begründung einer Entscheidung, die Darstellung einer Entscheidung und die Kombination einen einzelnen Patienten analysieren zu können oder direkt eine ganze Gruppe an Patienten zu analysieren.

Quellen

[1] Balzert, Helmut. *Lehrbuch der Softwaretechnik: Entwurf, Implementierung, Installation und Betrieb*. Spektrum Akademischer Verlag, 2011.

[2] "ISO 14971:2019 - Medical devices — Application of risk management to medical devices." ISO, https://www.iso.org/standard/72704.html. Accessed 24 January 2022.

[3] "Leitlinie Kawasaki Syndrom (S2k Leitlinie)." AWMF, https://www.awmf.org/uploads/tx_szleitlinien/185-003l_S2k_Kawasaki-Syndrom_2021-01.pdf. Accessed 24 January 2022.

[4] "Multisystem inflammatory syndrome in children and adolescents temporally related to COVID-19." *WHO | World Health Organization*, 15 May 2020, https://www.who.int/news-room/commentaries/detail/multisystem-inflammatory-syndrome-in-children-and-adolescents-with-covid-19. Accessed 24 January 2022.

[5] „DIN EN 62304 - 2016-10". https://www.beuth.de/de/norm/din-en-62304/256450057 (zugegriffen 26. Januar 2022).

[6] B. Friedman und H. Nissenbaum, „Bias in computer systems", *ACM Trans. Inf. Syst.*, Bd. 14, Nr. 3, S. 330–347, Juli 1996, doi: 10.1145/230538.230561.

[7] S. Corbett-Davies, E. Pierson, A. Feller, S. Goel, und A. Huq, „Algorithmic decision making and the cost of fairness", *arXiv:1701.08230 [cs, stat]*, Juni 2017, doi: 10.1145/3097983.309809.